Oracle® Communications Calendar Server

System Administrator's Guide

Release 7.0.4

July 2015

ORACLE

Oracle Communications Calendar Server System Administrator's Guide, Release 7.0.4

Contents

Chapter 1. Calendar Server 7 Administration Guide

Oracle Communications Calendar Server Administration Guide

This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

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

Installing Calendar Server
If you need to install and perform an initial configuration of Calendar Server 7, see Installation Scenario - Calendar Server 7.0.4.15.0.

Topics:

Setting up Your Initial Calendar Server 7 Deployment

The following tasks are meant to be of use in planning and setting up your initial deployment. After your deployment has gone live, you might also find it useful to revisit these tasks to make additional changes.

Provisioning Calendar Server Users

Topics in this section:

Provisioning Calendar Server Overview

Calendar Server uses Directory Server to store and retrieve user and resource information and to perform authentication. Before Calendar Server can access information in Directory Server, you need to prepare Directory Server by running the `comm_dssetup` script, as described in the installation documentation. The `comm_dssetup` script installs the necessary additional schema elements to support Communications Suite components, including Calendar Server.

> **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.

Calendar Server does not add or modify LDAP data. Calendar Server data is stored in an SQL database,

which can be either MySQL Server or Oracle Database. By default, Calendar Server automatically creates the database entries for users upon their initial Calendar Server login or invite. However, you must first perform some basic LDAP user provisioning for automatic calendar creation to work. You can provision Calendar Server users and resources in the Directory Server LDAP either by using Delegated Administrator or LDAP tools.

For users to be able to access Calendar Server services, you must provision them with the following functionality in LDAP:

- An email attribute, such as `mail`.
- A unique ID attribute corresponding to the value for the server configuration parameter `davcore.uriinfo.permanentuniqueid` (see Calendar Server Unique Identifier for more information). 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 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 the following:

- Messaging Server, Calendar Server, and Contacts Server LDAP Object Classes and Attributes
- Calendar Server and Directory Server Integration
- LDAP Schema Changes Between Calendar Server 6 and Calendar Server 7

To Define Valid Calendar Users

> The `davcore.ldapattr.userobject` configuration parameter was introduced in **Calendar Server 7 Update 2 Patch 5.**

The `davcore.ldapattr.userobject` configuration parameter defines what LDAP object class is required to consider an entry as a valid calendar user entry. 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.

To Create Calendar Account with Default Calendar Automatically Upon Login

1. Create users by provisioning them in LDAP.
2. Provide users instructions for logging in to Calendar Server.
   The auto-creation of the calendar accounts happens when users log in to Calendar Server. By default, the `davcore.autocreate.enableautocreate` parameter, which permits the auto-creation, is enabled. Disable this parameter if you do not want to automatically create calendar accounts upon login.
3. Use the `davadmin` command to set any of the `davcore.autocreate.*` parameters to customize your deployment:
   For more information on these parameters, see Calendar Server 7 Configuration Parameters.

To Enable and Disable Account Autocreation

> This feature is available starting in **Calendar Server 7 Update 3**.

You can enable or disable, on a system-wide basis, calendar account autocreation, either on login or invite.

- To enable calendar accounts autocreation:

```
davadmin config modify -o davcore.autocreate.enableautocreate -v true
```

- To disable calendar accounts autocreation:

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

For more information, see the `davcore.autocreate.enableautocreate` parameter in Calendar Server 7 Configuration Parameters.

To Provision 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 `icsdwphost` (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, which is `defaultbackend`.

When a user or group is assigned a calendar service, the `davEntity` objectclass is added along with `icscalendaruser` or `icscalendargroup` objectclass, 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.x 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 hostname.

To Provision 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 Installation Scenario - Calendar Server 7.0.4.15.0 and Installation Scenario - Delegated Administrator 7.0.0.9.0.
2. Using Delegated Administrator to create the hosted domain, and users, resources, and groups in that hosted domain.
   For more information, see Delegated Administrator Administration Guide.
3. Setting domain ACLs
   For more information, see Managing Domain Access Controls.

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

To Prevent a User or Resource From Accessing Calendar Server

To prevent a particular user from accessing Calendar Server 7, set the `icsStatus` attribute to **`inactive`**. As of Calendar Server 7 Update 2, this can be done on both a per user or resource, or domain basis. For more information, see icsStatus.

To Check for Active Calendar Users

Starting with **Calendar Server 7 Update 2 Patch 5**, you can use the `commands` log to check for which users have been querying their calendars. Thus, you can use that information to 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.

To Remove Calendar Users

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

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 need to do this so that the last step of deleting from LDAP by using the `commadmin` command is successful. Even when you delete the user account in Step 2, the LDAP entry for the user remains with the `icsStatus` attribute unchanged. Thus, you need to 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: Maintaining Referential Integrity in the *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**
> Beginning with **Calendar Server 7 Update 2 Patch 5**, the `search_calprops.wcap` command does not return calendars belonging to accounts which have a status of 'inactive,' 'removed,' or 'deleted.'

To Remove 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:

```
# <cal-svr-base>/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:

```
# <cal-svr-base>/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:

```
# <da-base>/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:

```
# <da-base>/bin/commadmin user delete -l jsmith -S mail
# <msg-svr-base>/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:

```
# <da-base>/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.>
```

To Move Calendar Users to a New Back-End Database

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.

```
# ./asadmin stop-domain domain1;./asadmin start-domain domain1
```

To Change a User's Email Address in the Calendar Server Database

> The `davadmin` command was enhanced in **Calendar Server 7 Update 2 Patch 5** to perform the `repair` operation.

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 for the old email address.

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 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.
   For more information on this command, see davadmin account.
4. (Optional) Remove the previous email value from the `mailAlternateAddress` values.

To Subscribe or Unsubscribe Calendars

> This feature is available starting in **Calendar Server 7.0.4.14.0**.

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/cal1/
     /home/userd@example.com/personal/
     ```

  2. Run the following command:

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

For more information, see `account Operation`.

Setting up CalDAV and CardDAV Autodiscovery

> This feature is available starting in **Calendar Server 7 Update 1**.

Calendar Server supports the `.well-known` URI concept as defined in Locating CalDAV and CardDAV services for CalDAV clients 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, administrators should configure a corresponding DNS record as described in Locating CalDAV and CardDAV services.

Configuring External Authentication

> This feature is available starting in **Calendar Server 7 Update 3**.

Calendar Server enables 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 would use the external directory for 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 Communications Suite directory user entry

Topics in this section:

To Configure 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 <cal-svr-base>/sbin
# ./davadmin ldappool create -n myldap -y
'ldaphost=ldap.example.com,ldapport=389,ldapusessl=true,binddn="cn=Di
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 `davadmin ldappool` command 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. See `externalAuthPreUrlTemplate` for more information.

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 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 Communications Suite directory by using the `externalAuthPostUrlTemplate` domain entry attribute. As with the the `externalAuthPreUrlTemplate` attribute, the `externalAuthPostUrlTemplate` value is an LDAP URL of the following form:

     ```
     ldap://<server name>/<search base
     DN>?<attributes>?<scope>?<search filter>
     ```

     See `externalAuthPostUrlTemplate` for more information. 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 Communications Suite directory `uid` attribute value (in the 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 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 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.

Creating Additional Back-end Calendar Server Stores

If you need to create a back-end calendar store after initial installation, see Configuring Multiple Calendar Server Back-end Hosts.

Making Calendar Server Secure

This information describes how to implement security for Calendar Server 7. It also provides links to security topics that provide more indepth information for configuring and administering Calendar Server security.

Topics in this section:

- Securing Communications to Calendar Server Back Ends
- Securing iSchedule Port
- Creating Secure Communications Between davadmin and Calendar Server
- Configuring GlassFish Server 3 Denial of Service Prevention
- Blocking CalDAV and WCAP Clients
- Configuring a Secure LDAP Connection
- Configuring Calendar Server for Virus Scanning
- Where to Find Additional Security Related Information

Securing Communications to Calendar Server Back Ends

This capability is available starting in **Calendar Server 7.0.4.14.0**.

You can enhance your security by configuring SSL communication between the Calendar Server front ends and back ends. First enable the back-end database servers for SSL, setting up either the required `trustStore` files or wallets. Then configure the Calendar Server front ends to connect over SSL by making use of the stored certificates. To configure a MySQL back end, see Configuring SSL for the Calendar Server MySQL Back End. To configure an OracleDB back end, see Configuring SSL for the Calendar Server OracleDB Back End.

You can also configure SSL communication between the Calendar Server front ends and the remote document stores. See To Configure Remote Document Store SSL.

Securing iSchedule Port

This capability is available starting in **Calendar Server 7.0.4.14.0**.

You can use the `service.dav.ischedulewhitelist` configuration parameter to specify the hosts that are allowed to send iSchedule POST requests. You can even limit the hosts to just the SMTP server that uses iSchedule for automatic iMIP handling. Additionally, you can change the maximum content length of requests POSTed to the iSchedule port by setting the value for the `davcore.serverlimits.maxischedulecontentlength` configuration parameter (the default is set to 10000000 bytes). This blocks very large freebusy or invitation requests at the iSchedule port itself.

- To specify a list of hosts that are allowed to send iSchedule POST requests, use the `davadmin config` command:

  ```
  davadmin config modify -o service.dav.ischedulewhitelist -v <hosts>
  ```

  where *hosts* can be:

- A space-separated list of single host IP addresses and/or Classless Inter-Domain Routing (CIDR) entries (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 to 10.20.30.255.
- 0.0.0.0/0, to allow all requests.
- An empty list, which denies all requests except for those from "localhost."
- To change the maximum size of iSchedule data that a client can POST, use the `davadmin config` command:

  ```
  davadmin config modify -o
  davcore.serverlimits.maxischedulecontentlength -v <maximum content
  length in bytes>
  ```

Creating Secure Communications Between davadmin and Calendar Server

This capability is available starting in **Calendar Server 7.0.4.14.0**.

The `davadmin` command uses the JMX protocol to connect to GlassFish Server. This section describes how to create secure communications between the `davadmin` command and the Calendar Server over SSL. To do so, you need to create a `trustStore` file for `davadmin`. If you are using SSL for

communicating with GlassFish Server, it is mandatory to also configure JMX to use SSL.

To Create Secure Communications Between davadmin and Calendar Server

1. Export the server certificate that GlassFish Server is using for SSL.
   Depending on the GlassFish Server version, you might use the Java `keytool` command or the NSS `certutil` command to export the certificate.
   - Example `keytool` command:

     ```
     keytool -exportcert -keystore keystore.jks -storetype JKS
     -alias s1as -rfc -file /tmp/s1as.txt
     ```

   - Example `certutil` command:

     ```
     /usr/sfw/bin/certutil -L -d . -n s1as -a > /tmp/s1as.txt
     ```

     In these examples, the current directory is the GlassFish Server configuration directory and the certificate is named `s1as`.

2. Import the GlassFish Server certificate into a Java `keystore` for use by the `davadmin` command with the `-s` option.
   For example:

   ```
   keytool -importcert -alias s1as -file /tmp/s1as.txt -keystore
   /var/opt/sun/comms/davserver/config/davtruststore -storetype JKS
   ```

3. Modify the `/var/opt/sun/comms/davserver/config/davadmin.properties` file to reflect the new `trustStore` file created in the previous step.
   Add the following line:

   ```
   secure=/var/opt/sun/comms/davserver/config/davtruststore
   ```

   Alternately, you could specify the explicit path to the `trustStore` file in the `davadmin` command with the `-s` option.
   For example:

   ```
   davadmin db backup -k /tmp/backup_file -O -A
   docstore_host.example.com:8008 -s /my_home/my_truststore -u mysql
   ```

4. In the GlassFish Server 3 Administration Console, enable secure JMX.
   a. Navigate to Configurations.
   b. Navigate to server-config.
   c. Navigate to Admin Service.
   d. On the JMX Connector tab, select the Enabled box for Security.

Configuring GlassFish Server 3 Denial of Service Prevention

This capability is available starting in **Calendar Server 7.0.4.14.0**.

GlassFish Server 3 provides the following ways to help prevent a Denial of Service (DoS) attack:

- Limit the size of a POST request
- Specify a request timeout value
- Create a blacklist of host names and/or IP addresses

You can configure additional DoS prevention at the Calendar Server level by using configuration parameters such as `davcore.serverlimits.maxcontentlength`, `service.dav.blacklist`, `service.wcap.blacklist`, `service.dav.ischedulewhitelist`, and `davcore.serverlimits.maxischedulecontentlength`. For a complete list of configuration parameters, see Calendar Server 7 Configuration Parameters.

To Limit the Size of a POST Request

To limit the size of a POST request, set the Max Post Size field in the GlassFish Server Administration Console:

1. Choose Configurations.
2. Choose server-config.
3. Choose Network Config.
4. Choose Protocols.
5. Choose http-listener*.
6. Choose HTTP.
7. Edit the Max Post Size field.

On this same page, you can also set the Request Timeout field so that a request fails if it takes longer than the specified time to complete.

Alternately, you can use the `asadmin` command to set the Max Post Size, for example:

```
asadmin set
server-config.network-config.protocols.protocol.http-listener-1.http.max-pc
```

To Create a Blacklist

To create a blacklist of host names and/or IP address, use the GlassFish Server Administration Console to navigate to the server page:

1. Choose Configurations.
2. Choose server-config.
3. Choose Virtual Servers.
4. Choose server.
   Add properties to indicate which hosts or IP addresses you want to blocked. Use the `allowRemoteHost` or `allowRemoteAddress` property names to whitelist hosts, and use the `denyRemoteHost` or `denyRemoteAddress` property names to block a host or IP address. Separate multiple host names or IP addresses by commas.

Alternately, you can use the `assadmin` command to create a whitelist or blacklist of hosts.

- For example, to create a whitelist:

  ```
  asadmin set
  "server-config.http-service.virtual-server.server.property.allowRemot
  ```

- For example, to create a blacklist:

```
asadmin set
"server-config.http-service.virtual-server.server.property.denyRemote
```

Blocking CalDAV and WCAP Clients

> This feature is available starting in **Calendar Server 7 Update 2**.

Calendar Server can block requests from certain versions of CalDAV clients and WCAP clients that are known to be problematic. Blocked clients receive an error, HTTP 403 status, which indicates that the server can be reached, but the server declined to allow access to the page. By using this capability, you can help to prevent denial of service attacks by blocking rogue clients that can inundate the server with requests.

Topics in this section:

To Block CalDAV Client Requests

1. Look at the HTTP user-agent request header that contains the client information to determine which client to add to the reject list.
   Here are some user-agent examples:

```
user-agent: DAVKit/3.0.6 (661); CalendarStore/3.0.8 (860); iCal/3.0.8
(1287); Mac OS X/10.5.8 (9L31a)
user-agent: DAVKit/4.0.1 (730); CalendarStore/4.0.1 (976); iCal/4.0.2
(1379); Mac OS X/10.6.3 (10D573)
user-agent: DAVKit/4.0.1 (730); CalendarStore/4.0.1 (976); iCal/4.0.2
(1379); Mac OS X/10.6.3 (10D578)
user-agent: DAVKit/4.0.3 (732); CalendarStore/4.0.3 (991); iCal/4.0.3
(1388); Mac OS X/10.6.4 (10F569)
user-agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-US;
rv:1.9.2.11) Gecko/20101012 Firefox/3.6.11
user-agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-US;
rv:1.9.2.5pre) Gecko/20100430 Lightning/1.0b2pre
OracleBeehiveExtension/1.0.0.0pre6 Thunderbird/3.1b2
user-agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US;
rv:1.9.2.10) Gecko/20100914 Firefox/3.6.10
user-agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US;
rv:1.9.2.11) Gecko/20101012 Firefox/3.6.11
user-agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US;
rv:1.9.2.11) Gecko/20101013 Lightning/1.0b2 Thunderbird/3.1.5
user-agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US;
rv:1.9.2.12) Gecko/20101027 Lightning/1.0b2 Thunderbird/3.1.6
user-agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.24)
Gecko/20100228 Lightning/0.9.7.4-Oracle Thunderbird/2.0.0.24
user-agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.10)
Gecko/20100512 Lightning/1.0b1 Thunderbird/3.0.5
user-agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.9)
Gecko/20100317 Lightning/1.0b1 Thunderbird/3.0.4
user-agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.11)
Gecko/20101012 Firefox/3.6.11 ( .NET CLR 3.5.30729)
user-agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.11)
Gecko/20101013 Lightning/1.0b2
OracleBeehiveExtension/1.0.0.0-OracleInternal Thunderbird/3.1.5
```

```
user-agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.11)
Gecko/20101013 Lightning/1.0b2 Thunderbird/3.1.5
user-agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.12)
Gecko/20101026 Firefox/3.6.12 ( .NET CLR 3.5.30729)
user-agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.12)
Gecko/20101027 Lightning/1.0b2
OracleBeehiveExtension/1.0.0.0-OracleInternal Thunderbird/3.1.6
user-agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.12)
Gecko/20101027 Lightning/1.0b2 Thunderbird/3.1.6
user-agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.8)
Gecko/20100722 Firefox/3.6.8
user-agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.9)
Gecko/20100825 Lightning/1.0b2 Thunderbird/3.1.3
user-agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US)
AppleWebKit/534.0 (KHTML, like Gecko) Chrome/6.0.408.1 Safari/534.0
user-agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.11)
Gecko/20101013 Lightning/1.0b2 Thunderbird/3.1.5
user-agent: Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.2.12)
Gecko/20101027 Lightning/1.0b2 Thunderbird/3.1.6
user-agent: Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.2.12)
Gecko/20101027 Ubuntu/10.10 (maverick) Firefox/3.6.12
user-agent: Mozilla/5.0 (X11; U; SunOS i86pc; en-US; rv:1.8.1.21)
Gecko/20090323 Lightning/0.9 Thunderbird/2.0.0.21
user-agent: Mozilla/5.0 (X11; U; SunOS i86pc; en-US; rv:1.9.1.8)
Gecko/20100226 Firefox/3.5.8
user-agent: Mozilla/5.0 (X11; U; SunOS i86pc; en-US; rv:1.9.1.8)
Gecko/20100302 Lightning/1.0b1 Thunderbird/3.0.3
user-agent: Mozilla/5.0 (X11; U; SunOS i86pc; en-US; rv:1.9.2.4)
Gecko/20100610 Lightning/1.0b2 Thunderbird/3.1
user-agent: Mozilla/5.0 (X11; U; SunOS sun4u; en-US; rv:1.9.0.17)
Gecko/2009122317 Firefox/3.0.17
```

```
user-agent: Mozilla/5.0 (X11; U; SunOS sun4v; en-US; rv:1.8.1.23)
Gecko/20090910 Lightning/0.9 Thunderbird/2.0.0.23
```

2. Specify a client or list of clients to be rejected for the DAV protocol by using the `service.dav.blacklist` parameter.
   This parameter takes a single client/version pair, or list of space-delimited client/version pairs. For example, to cause Calendar Server to reject client requests from Lightning 1.0 Beta 1 and iCal version 3.0.8, use the following command:

   ```
   davadmin config modify -o service.dav.blacklist -v Lightning/1.0b1
   iCal/3.0.8
   ```

   This example causes Calendar Server to reject requests from a client when the exact string "Lignting/1.0b1" or "iCal/3.0.8" appears in the client's user-agent header. A variation of the client/version pair is when the version part is expressed as a regular expression.

See Calendar Server 7 Configuration Parameters for more information on the `service.dav.blacklist` parameter.

To Block WCAPbis Clients

1. Look at a WCAP client's HTTP user-agent request header that contains the client information (regular expression) to use to deny client access.
2. Use the `service.wcap.blacklist` configuration parameter to specify a space separated list of regular expressions.

See Calendar Server 7 Configuration Parameters for more information on the `service.wcap.blacklist` parameter.

Configuring a Secure LDAP Connection

Like other applications, Calendar Server and GlassFish Server use the LDAP protocol to read from and write to Directory Server. The default is to communicate LDAP data by "unsecured" transmission. You can configure your Calendar Server deployment to use Secure Sockets Layer (SSL) for secured transmission of data. You do so by installing a Certificate Authority (CA) certificate on the GlassFish Server then configuring Calendar Server to use LDAP for SSL.

To Configure Calendar Server for Secure LDAP

1. Configure GlassFish Enterprise Server to use a CA signed certificate for SSL.
2. When performing this task, note the following.
   - Not all GlassFish Server domains use the same security key store.
     GlassFish Server may end up using either the NSS security library and associated key store for certificates, or it may use the Java Key Store instead. The keystore that GlassFish Server uses is initially determined by the GlassFish Server "Profile:" Enterprise, Developer, or Cluster. The Enterprise profile uses NSS, while the Developer and Cluster use Java Key Store.

     You can specify the profile when you create a new domain. However, the initial domain defaults to a profile depending on the version of GlassFish Server installed along with possible configuration specifications.

     The To Configure GlassFish Enterprise Server to Use a CA Signed Certificate for SSL procedure assumes the Enterprise version of GlassFish Server, which uses the NSS security library and includes the `certutil` utility.

     If you want to use a profile with Java Key Store, review the Java Key Store documentation

on the tool "keytool." Use a newer version of keytool from Java 6.

The key store is configured during creation of the domain. Although both sets of key store files may be present in the `config` directory, only the chosen key store is configured with common CA certificates and the GlassFish Server password. Thus, for example, if Java Key Store is configured, and you try to generate a certificate request with `certutil`, you may get an I/O error because the NSS database is not initialized with the GlassFish Server password.

Though it should not be necessary, if you need to change the key store password, use the `asadmin` command, not the `certutil` utility.

- Certificate request generation depends on generating the request with the key store database in the domain's `config` directory.
- GlassFish Server installs with a default self-signed certficate named `s1as`. This should already be configured for SSL, and so SSL can be tested before adding a valid certificate. However, many clients warn that the SSL certificate cannot be trusted, and other clients may simply not work.

  Some older versions of FireFox report a "connection was interrupted" error when the certificate is self-signed or otherwise not valid.

3. Configure Calendar Server to use LDAP for SSL.
   - In step #2, while running `davadmin config` to process the `davadmin` commands: Processing the configuration file can take time and can result in multiple messages to restart GlassFish Server. Wait until all the commands are finished processing before restarting GlassFish Server. Otherwise, some configuration parameters might not be processed, leaving SSL partially configured.

Configuring Calendar Server for Virus Scanning

> This feature is available starting in **Calendar Server 7 Update 2**.

Calendar Server 7 enables attachments virus scanning by making use of the Messaging Server MTA and its capability to interface to many different AVS systems. Calendar Server packages the calendar event and attachment and sends it by using SMTP to the Messaging Server MTA, where the scanning takes place on the configured AVS system. Messaging Server relays the result of the scan to Calendar Server. Calendar Server can be configured to test and optionally reject incoming data, as well as scan existing data (and optionally delete) by using the `davadmin vscan` utility.

See Configuring and Administering Virus Scanning in Calendar Server for details.

Where to Find Additional Security Related Information

For more information on making your Calendar Server deployment secure, see Setting Up and Managing Calendar Server Security.

Enabling Calendar Server 7 Advanced Features

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 Attachments

Starting with **Calendar Server 7 Update 2**, the `davcore.attachment.enable` configuration parameter determines whether a deployment is using 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`**.
  For example:

```
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?

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 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 Using the

iSchedule Channel to Handle iMIP Messages for instructions on how to set up Messaging Server. No special setup is required for Calendar server.

Starting with **Calendar Server 7.0.4.14.0**, you can 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 Securing iSchedule Port.

Stopping and Starting Calendar Server 7 Services

Topics in this section:

- Stopping and Starting Calendar Server
- Starting and Stopping the Document Store

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:

```
/opt/SUNWappserver/bin/asadmin stop-domain domain1
```

- To start Calendar Server:

```
/opt/SUNWappserver/bin/asadmin start-domain domain1
```

Starting and Stopping the Document Store

> The document store was introduced in **Calendar Server 7 Update 1.** For more information, see Planning the Document Store.

- To start the document store server:

```
cd <cal-svr-base>/sbin
./start-as
```

- To stop the document store server:

```
cd <cal-svr-base>/sbin
./stop-as
```

Administering a Calendar Server 7 Deployment

Topics in this section:

Administering GlassFish Server

Calendar Server 7 depends on Sun GlassFish Server as a web container. See the Oracle GlassFish Server Product Documentation for details on administering Glassfish Server. Starting with version 7.0.4.14.0, Calendar Server supports GlassFish Server 3.

The following Glassfish Server documentation links are also provided to help you get started:

- Introduction to Certificates and SSL
- HTTP and IIOP Commands
- `asadmin` Command Usage
- High Availability in GlassFish Enterprise Server
- To Configure GlassFish Enterprise Server to Use a CA Signed Certificate for SSL

Administering the Document Store

> The document store was introduced in **Calendar Server 7 Update 1.**

Topics in this section:

Document Store Overview

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.`**`.dbdir`** **configuration parameter, if the store is local, and the** **`store.dav.`**`.attachstorehost` configuration parameter, if the store is remote. For more information, see Configuring the Calendar Server 7.0.4.15.0 Document Store.

To start and stop the document store process, see Starting and Stopping the Document Store.

To Change the Password Used for Remote Document Store Authentication

This feature is available starting with **Calendar Server 7 Update 3.**

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 in sync.

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

   ```
   cd <cal-svr-base>/sbin
   ./davadmin passfile modify -O
   ```

   Respond to the prompts.

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

   ```
   cd <cal-svr-base>/sbin
   ./stop-as
   ./start-as
   ```

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

   ```
   cd <cal-svr-base>/sbin
   ./davadmin modify passfile
   ```

   Respond to the prompts.

Note on Backing up the Document Store

The `davadmin db backup` command prompts for the document store password starting with **Calendar Server 7 Update 3.**

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, you can create a password file by running the `davadmin passfile` command.

Using Calendar Server 7 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 *cal-svr-base*`/sbin` directory with user or group `bin/bin` permissions. See Calendar Server Command-Line Utilities.

Administering Calendar Server 7 Logging

Topics in this section:

- Calendar Server 7 Logging Overview
- To Log Calendar Server Information to Application Server Log File
- To Configure Calendar 7 Log Files

Calendar Server 7 Logging Overview

Calendar Server 7 maintains four 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. For more information on the `commands` log, see Using 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` - **Beginning with Calendar Server 7 Update 2**, stores information on virus scanning actions.

Each of these files has a suffix of `.#`, which is a rollover number indicating that the log file was filled to its maximum capacity and was rolled over to another file. The lowest numbered log file is the newest.

Each log file has its own configuration attribute that controls the log file location, maximum size, and log level.

The Calendar Server log files are kept separate from the GlassFish Enterprise Server log files. The GlassFish Server log files are maintained in the `${APPSERVER_DOMAIN}/logs` directory, where `${APPSERVER_DOMAIN}` is the path to the respective container's base directory, for example, `/opt/SUNWappserver/domains/domain1/`. 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.

To Log Calendar Server Information to Application Server Log File

Calendar Server sets its `logToParent` flag to `false` by default, preventing logging of information to the Application Server log file.

To log calendar information to both the Application Server log file (`server.log`) and the Calendar Server log file (`calendar.0`)):

- Set the `log.dav.commands.logtoparent` parameter to `true`:

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

To Configure Calendar 7 Log Files

- Use the davadmin utility to configure the Calendar Server 7 log file parameters as shown in the following table.
  For more information on each of the configuration parameters and their default values, see

Calendar Server 7 Configuration Parameters.

The following table shows the log file parameters.

Calendar Server Log File Parameters

Parameter	Description
`log.dav.`*name*`.logdir`	Specifies the log file directory path
`log.dav.`*name*`.loglevel`	Specifies the log level: • `SEVERE`: Logs catastrophic errors. • `WARNING`: Logs major errors or exceptions with the system. • `INFO`: Logs general informational messages. This is the default level. Any information logged at this level is placed in the log file. • `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 finer grain details than `FINE`. • `FINEST`: Logs the finest grain details of code flow or problem information. Turning on logging at this level could cause massive amounts of data to be put into the log file, making it hard to parse.
`log.dav.`*name*`.logtoparent`	Sets a flag to enable logging to the GlassFish Enterprise Server log file, in addition to the Calendar Server logs
`log.dav.`*name*`.maxlogfiles`	Specifies the maximum number of log files
`log.dav.`*name*`.maxlogfilesize`	Specifies the log file's maximum size

Where *name* can be `commands`, `errors`, `scheduling`, `telemetry`, or `scan`, depending on which log file type you want to configure. The *name* to use to get the Calendar Server logs is `errors`.

To View the Document Store Logs

- The document store logs are located in the *datadir*`/logs` directory. Change to this directory to view the log files.

To Set the Logging Level for Calendar Server 7 Logs in GlassFish Enterprise Server

You can set the logging level for Calendar Server logs in the GlassFish Enterprise Server logs, using the Admin Console.

1. In a browser, access the Admin Console, for example:

   ```
   http://example.com:4848
   ```

2. Type the administration user name and password, specified when you installed the product, and click Log In.
3. In the Admin Console, click the Logging tab.
4. Click the Logging Settings tab.
5. At the bottom of the page add a logger and set its level by clicking the Add Properties button. For the Name, the logger should have the root logger name, `com.sun.comms`, or a package name, such as `com.sun.comms.davserver.core`. For the Value, type any of the following

values (listed from highest to lowest):

- SEVERE
- WARNING
- INFO
- CONFIG
- FINE
- FINER
- FINEST
- OFF

  Messages at the selected level or higher appear in the log. The default level is INFO, so the log will contain messages of INFO, WARNING, or SEVERE levels.

6. When done, click Save.

Using the scheduling Log

The following table explains the scheduling codes in the scheduling log file.

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 need to 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-13385
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-13385
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-13385
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-13385
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 following functionality documented in this section was introduced in **Calendar Server 7 Update 2 Patch 5**.

The Calendar Server `commands` log contains per servlet entries that are designed to help monitor requests to the server and help diagnose problems. Starting with Calendar Server 7 Update 2 Patch 5, the `commands` log is enhanced to pass additional information, including the principal account that logged in and what operations were done from that account.

The `commands` log records contain three set fields and one variable field:

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.
Variable	Logs information about the start and end of specific internal server operations.. 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.

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-cf21-4d68-90b6-35095ad51ccb8
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-cf21-4d68-90b6-35095ad51ccb8
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. Starting in **Calendar Server 7 Update 2 Patch 6**, 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.

> The following features documented in this section were introduced in **Calendar Server 7 Update 1.**

Topics in this section:

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 in the case of 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 7 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 as follows:

- 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 need to 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.

Configuration Parameters for Access Control

The following table describes the configuration parameters that Calendar Server uses for access control.

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

See Calendar Server 7 Configuration Parameters for more information on these access control configuration parameters.

Command-Line Utilities for Access Control

Use the davadmin calendar to get or set calendar ACLs for calendars and the davadmin account command to get or set scheduling ACLs for access control.

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).

Managing Domain ACLs

Starting with Calendar Server 7 Update 2, domain ACLs control calendar operations that span multiple domains. Calendar Server 7 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 Managing Domain Access Controls.

Managing Dynamic Group ACLs

Starting with Calendar Server 7 Update 2, the group ACL feature now 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.

Topics in this section:

- Configuring Scheduling Options
- Modifying Calendar Booking Window
- Modifying Calendar Double Booking
- 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)

Modifying Calendar Booking Window

The `davcore.scheduling.minbookingwindow` configuration parameter was introduced in **Calendar Server 7 Update 3.**

Topics in this section:

- Calendar Booking Window Overview
- To Configure a Booking Window

Calendar Booking Window Overview

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 minbooking window 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 the `davcore.scheduling.minbookingwindow` parameter (introduced in Calendar Server 7 Update 3) and the `davcore.scheduling.maxbookingwindow` parameter. Starting with Calendar Server 7 Update 3, 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.

To Configure 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)&minbookingw
```

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

> This feature is available starting in **Calendar Server 7 Update 3**.

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 autocreation to automatically assign the double-booking property flag. Additionally, you can control the value assigned during autocreation 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.

Topics in this section:

- Controlling Double Booking When Creating Accounts Automatically
- To Modify Configuration Parameters That Control Double Booking
- To Override the Account Autocreation Through LDAP

- Manually Creating Accounts
- To Modify Double Booking on Existing Accounts

Controlling Double Booking When Creating Accounts Automatically

Because autocreation 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 To Provision Calendar Users 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)

Both configuration parameters take the options described in the following table. Double booking is allowed on calendars when the value of these
attendant flag options is `0`, `1`, `4`, or `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
`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.

To Modify Configuration Parameters That Control Double Booking

- Use the `davadmin config modify` command to change double booking behavior.
  For example, this 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
```

  This 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
```

To Override 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 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:
   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.

To Modify 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

The following features documented in this section were introduced in **Calendar Server 7 Update 1.**

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 recipients' 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 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*, and `davcore.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

> The following features documented in this section were introduced in **Calendar Server 7 Update 1.**

Topics in this section:

- About Resource Calendars
- To Provision Resource Calendars (commadmin)
- To Provision Resource Calendars (Delegated Administrator Console)
- To Manage 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 with Delegated Administrator or LDAP tools. See Messaging Server, Calendar Server, and Contacts Server LDAP Object Classes and Attributes 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.

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.

The `mail` LDAP 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 To Manage 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.

To Provision Resource Calendars (commadmin)

- See commadmin resource create. 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 Calendar Server 7 coexistent deployments

Example for one back-end host:

```
<da-base>/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:

```
<da-base>/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:

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

To Provision Resource Calendars (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.

To Manage 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=exaample.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 Timezones Support

> Support for WCAP was introduced in **Calendar Server 7 Update 1.**

Topics in this section:

About Timezones

Timezones 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 timezone

information. The timezone information is compiled and shipped along with Calendar Server. Each Calendar Server patch is updated to the latest available Time zone Database.

Adding New WCAP Timezones

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

The list of WCAP timezones is derived from the list provided in the *cal-svr-base* `/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 timezone.

For more information on how this works with Convergence, see Customization Example - Adding and Modifying Calendar 7 Timezones in Convergence 2.

To Add an Alias to an Existing Timezone

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

1. Edit the *cal-svr-base*`/config/timezoneids.txt` file.
2. Find the corresponding timezone 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` timezone 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.

To Add a New Timezone

1. Find the timezone or its equivalent in the list of Time Zone Database supported by the server.
2. Edit the *cal-svr-base*`/config/timezoneids.txt` file.
3. Add an entry for that timezone 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 timezone ID entry.
5. Restart Calendar Server.
   See Stopping and Starting Calendar Server.

Customizing Calendar Notifications

Calendar Server 7 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 7 Notifications for details.

Administering the Calendar Server Back End Databases

Topics in this section:

- Administering the MySQL Database
- Administering the Oracle Database

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
- MySQL Server and Server-Startup Programs
- MySQL Administrative and Utility Programs
- Best Practices for Backing Up and Restoring Databases in Calendar Server Deployments

> **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.

- Starting and Stopping Oracle Database
- Administering Oracle Database
- Best Practices for Backing Up and Restoring Databases in Calendar Server Deployments

> **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 Best Practices for Backing Up and Restoring Databases in Calendar Server Deployments.

Removing Unwanted Calendar Data to Reclaim Space

Topics in this section:

- Purging Deleted Calendar Entries
- Purging Messages from the Calendar 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. For more information on this parameter, see Calendar Server 7 Configuration Parameters.

Purging Messages From the Scheduling Inbox and Outbox

The ability to purge messages from the scheduling inbox and outbox was introduced in **Calendar Server 7 Update 1 Patch 3.**

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. Prior to Calendar Server 7 Update 2 Patch3, these messages were deleted only if the end user or the end user's client explicitly deleted them. This could have lead to those collections growing indefinitely. Starting with Calendar Server 7 Update 1 Patch 3, 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. For more information on these options, see the Calendar Server 7 Configuration Parameters. 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/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).

Troubleshooting Calendar Server 7

See Calendar Server 7 Troubleshooting.

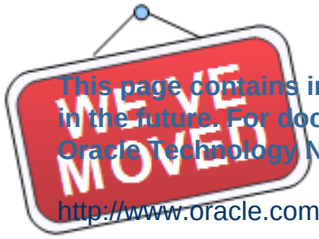
Where to Go for More Information

See the following Calendar Server 7 reference information:

- Calendar Server 7 Command-Line Utilities
- Calendar Server 7 Configuration Parameters
- Calendar Server Supported Standards
- Communications Suite Schema Reference
- Calendar Server 7 Configuration Reference
- Calendar Server and Directory Server Integration
- Calendar Server Common Topics
- Calendar Server Clients

Chapter 2. Calendar Server 7 Performance Tuning

Oracle Communications Calendar Server Performance Tuning



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This information describes how to tune your Calendar Server deployment.

Topics:

Calendar Server 7 Tuning

The Calendar Server logging function is I/O intensive. For optimal performance, decrease the log level to `INFO`. 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 -o log.dav.errors.loglevel -v INFO
$ davadmin config -o log.dav.commands.loglevel -v INFO
```

Sun GlassFish Enterprise Server Tuning

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

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
```

JDBC Pool

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

HTTP

http-service

keep-alive	max-connections	250
	thread-count	25
	timeout-in-seconds	30
request-processing	header-buffer-length-in-bytes	16384
	initial-thread-count	10
	request-timeout-in-seconds	20
	thread-count	50
	thread-increment	10
connection-pool	max-pending-count	4096
	queue-size-in-bytes	4096
	receive-buffer-size-in-bytes	4096
	send-buffer-size-in-bytes	8192

http-listener

acceptor-threads	1
accessLoggingEnabled	false
xpowered-by	false

MySQL Server Tuning

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 (Calendar Server 7 Update 1 and
prior releases)
transaction_isolation = READ-COMMITTED (starting with Calendar Server 7
Update 2)
tmp_table_size = 64M
log-bin=mysql-bin
expire_logs_days=1
binlog_format=row (Calendar Server 7 Update 1 and prior releases)
binlog_format=mixed (starting with Calendar Server 7 Update 2)
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
default-storage-engine = InnoDB
character-set-server = utf8 (Calendar Server 7 Update 3 and prior
releases)
character-set-server = utf8mb4 (starting with Calendar Server
7.0.4.14.0)
```

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

Oracle Solaris CMT Server Tuning

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:

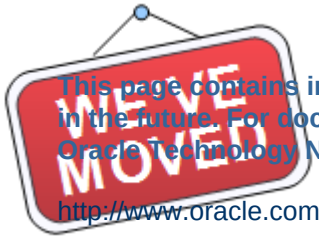
```
zfs create rpool/data
zfs set recordsize=16K rpool/data
```

Reference

- MySQL: http://www.solarisinternals.com/wiki/index.php/Application_Specific_Tuning
- Network: http://www.solarisinternals.com/wiki/index.php/Networks
- Application 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

Chapter 3. Calendar Server 7 Troubleshooting

Calendar Server 7 Troubleshooting



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

Troubleshooting Calendar Server Initial Configuration

If you experience trouble configuring Calendar Server (running the initial configurator program, `init-config`), and you are receiving an error from GlassFish Server, make sure that you running at least Java 1.6 and that your environment is correctly configured. For more information, see Installation Scenario - GlassFish Server or Oracle 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 need to 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 Enterprise Server web container is running and that Calendar Server is deployed. You can use either the GlassFish Server Admin Console or the `asadmin` command-line utility.

Topics in this section:

To Use the asadmin Command-line Utility 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.

To Use the GlassFish Server Admin Console to Check davserver Status

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

To Use the asadmin Command-line Utility to Check davserver Status

- Run the following command:

```
# 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.
```

To Troubleshoot davserver

1. If davserver is not enabled, check the GlassFish Server log, `server.log`, in the `/opt/SUNWappserver/domains/domain1/logs` or equivalent 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 -o log.dav.errors.loglevel -v FINE
```

To Troubleshoot 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 `.asadmintrustore` 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.

To Troubleshoot Back-end Errors on MySQL Server

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

1. Start the GlassFish Server Admin 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:

```
/opt/SUNWappserver/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=mysqlpa
caldavPool
% asadmin create-jdbc-resource -p <admin-port> --user admin
--connectionpoolid caldavPool jdbc/defaultbackend
```

   If you recreate the JBDC 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.

```
% asadmin stop-appserv; asadmin start-appserv
```

To Import a Convergence `ics` File

You might see the following error when importing an `ics` file that was created in Convergence. The following example is for Calendar Server 7. Starting with Calendar Server 7 Update 2, the `davadmin` command has been made more secure by the removal of the capability to "pass in" passwords by using a password file. In Calendar Server 7 Update 2, all `davadmin` passwords must now be entered by typing in to a no-echo prompt.

```
# ./davadmin calresource import -u admin -W passwordfile -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
```

Starting with **Calendar Server 7 Update 1**, the `calcomponent` argument replaces the `calresource` argument.

This is likely a Convergence problem, since it creates the todo by setting `DTSTART` and `DUE` to the same time. Note that this is due to the restrictions as 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`.

To Use SSL With iPhone, iPod, iPad, or iCal

To enable clients such as the iPhone and iCal to communicate by Secure Sockets Layer (SSL) with Calendar Server, you need to complete the steps as described in [Calendar Server 7.0.4.15.0 Post Configuration].

Disabling Elliptic Curve Encryption is required for SSL to work with these clients.

To Refresh 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.

To Troubleshoot 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 do the following:

   ```
   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 7 log files.

Topics in this section:

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 Filename

```
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.sun.com:8080
INFO    [2009-08-24T19:35:10.418-0700] <...DavServerServlet.service>
[RES] [501]    Command execution time: 0.0020 secs
```

For more information, see Administering Calendar Server 7 Logging.

Using the Browser Servlet

You can use a web-based user interface 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 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.

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
- 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:

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 OS needs to be reloaded twice to get the newly created Todo.
- Inviting users on Thunderbird for Solaris OS 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 OS. 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 a 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 cancelled.
- 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.x 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 and Event Time Modifications

See Calendar Server Clients for more information.

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

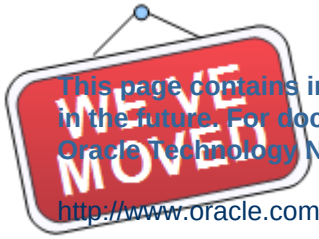
See To Check for Active Calendar Users.

Troubleshooting Calendar Server Agent Alerts in Instant Messaging

You can configure Instant Messaging for Java Message Service (JMS) to support Calendar Server 7 Agent alerts, as described in Configuring Instant Messaging 9.0.1.4.0 Calendar Agent with Calendar Server 7. 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.

Chapter 4. Calendar Server Supported Standards

Calendar Server Supported Standards



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This information lists national, international, industry and de-facto standards related to electronic calendaring and for which support is claimed by Calendar Server 7. Most of these are Internet standards, published by the RFC Editor and approved by the Internet Engineering Task Force (IETF). Standards for documents from other sources are noted.

Calendaring

The following documents are relevant to national and international standards for calendaring.

RFC2616	Hypertext Transfer Protocol HTTP/1.1
RFC4791	Calendaring Extensions to WebDAV (CalDAV)
RFC5545	Internet Calendaring and Scheduling Core Object Specification
RFC5546	iCalendar Transport-Independent Interoperability Protocol (iTIP)
RFC6047	iCalendar Message-Based Interoperability Protocol
RFC6578	Collection Synchronization for WebDAV
RFC6638	Scheduling Extensions to CalDAV

The following documents are in draft state.

caldav-ctag-02	Calendar Collection Entity Tag (CTag) in CalDAV
draft-daboo-srv-caldav-10	Locating CalDAV and CardDAV services

Chapter 5. Configuring and Administering Virus Scanning in Calendar Server

Configuring and Administering Virus Scanning in Oracle Communications Calendar Server

This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

Virus scanning is available starting in **Calendar Server 7 Update 2**.

Topics:

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."

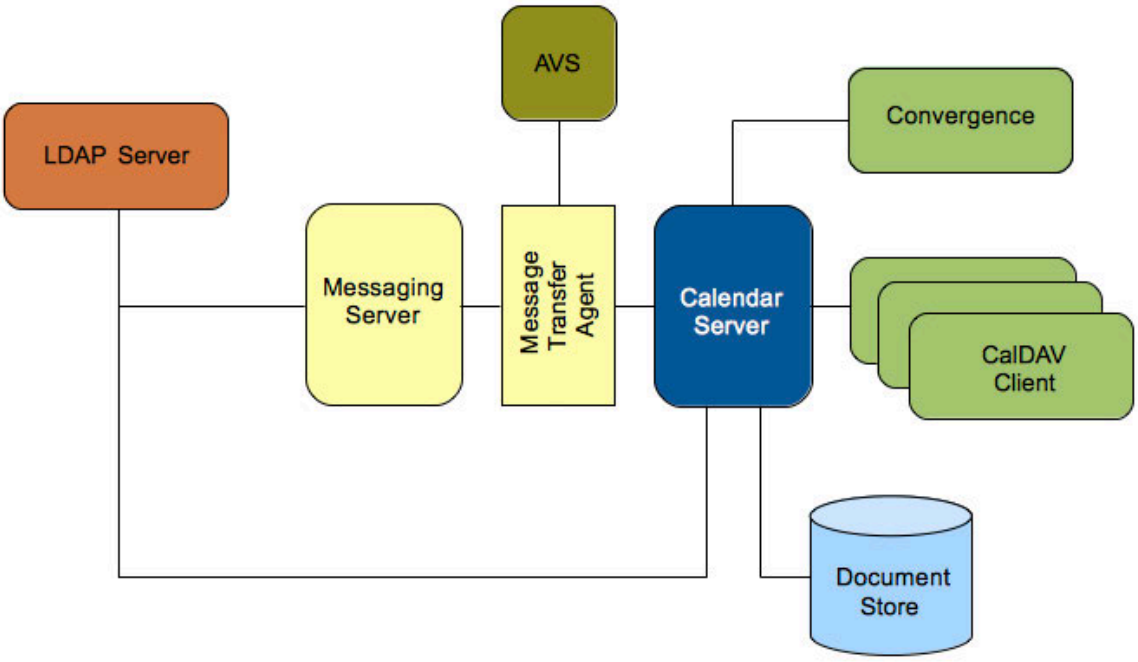
Virus scanning is not performed by Calendar Server itself. Instead, you configure an Oracle Communications Messaging Server's Message Transfer Agent (MTA) to filter the calendar data. You can configure Calendar Server to share an existing MTA that has already been configured for Messaging Server virus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

The following figure depicts the Calendar Server virus scanning logical architecture.

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.virusscan.onlinevirusaction` parameter remains unset (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.virusscan.onlinevirusaction` is set to "keep," the attachment is accepted. The

`davcore.virusscan.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

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.

To Install 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 Installation Scenario - Message Transfer Agent.

When configuring Messaging Server, the "configure" step requires a valid LDAP server that is used to include configuration data such as the default mail domain and messaging administrator account. The LDAP server that you specify needs to be available during virus scanning operations. However, due to MTA caching of LDAP data, this server 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 vendor-specific sections in Integrating Spam and Virus Filtering Programs Into Messaging Server.

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 should return `FilterVerdictPositive`. Calendar Server

checks SMTP return values for this exact string, which is defined in the `option.dat` file. See Example MTA Configuration 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.

To Create 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 `sourcespamfilter`*n* that is configured with the desired virus scan software, so that incoming calendar data is tested. For instructions, refer to the following documentation:

- Configuring Rewrite Rules
- MTA Concepts
- Messaging Server Administration Guide

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

The 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 MTA documentation on rewrite rules and channels for more information:

- Configuring Rewrite Rules
- MTA Concepts
- Messaging Server Administration Guide

Configuring Calendar Server

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.viruscan.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 Virus Scanning 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 Installation Scenario - Message Transfer Agent for instructions.
   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. For more information, see Is ClamAV Integrated With Messaging Server? and Deploying ClamAV. To download ClamAV, see ClamAV Binaries.
   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 Linus: `/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
["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 as described in the MTA documentation.

```
!
! 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's virus scan email address to be something 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.

Calendar Server Virus Scanning Configuration Parameters

The following table describes the Calendar Server virus scanning configuration parameters.

Calendar Server Virus Scanning Parameters

Parameter	Description
`davcore.virusscan.host`	Specifies the host name of the MTA that performs virus scanning for the Calendar Server.
`davcore.virusscan.port`	Specifies the SMTP port of the MTA that performs virus scanning for the Calendar Server.
`davcore.virusscan.emailaddress`	Specifies the email address for the user that the MTA recognizes for Calendar Server virus scans. Use the MTA's correct mail domain so the MTA does not try to find information about an unknown domain.
`davcore.virusscan.timeout`	Specifies the timeout for the SMTP connection. Default is 10000 milliseconds.
`davcore.virusscan.onlineenable`	Specifies to enable virus scanning on incoming data to the Calendar Server. Default is `false`, meaning do not scan incoming data.
`davcore.virusscan.onlinevirusaction`	Specifies the action to take if a virus is detected on incoming data. The default is an empty string, which logs a warning message to the Calendar Server's "scan" log. A value of `reject` causes either an HTTP error 403 to be returned to the client, if using a CALDAV command, and a WCAP error 100 to be returned, if using a WCAP command, notifying the client that the data has been rejected.
`davcore.virusscan.onlinefailureaction`	Specifies the action to take if the virus scan service is experiencing a failure (such as the scanning service is down). Default is the empty string, which logs a warning message to the Calendar Server's "scan" log. A value of `reject` causes either an HTTP error 403 to be returned to the client, if using a CALDAV command, and a WCAP error 100 to be returned, if using a WCAP command, notifying the client that the data has been rejected.
`davcore.virusscan.clivirusaction`	Specifies the action to take if a virus is detected during a command-line virus scan by using the `davadmin` utility. Default is an empty string, which logs a warning message to the Calendar Server's "scan" log file. A value of `delete` means that the data is automatically deleted. Use this option with care.

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 by using the the `davadmin` utility.

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

Calendar Server Virus Scan Command-line Utility

davadmin Class of Operation

Argument	Description
`vscan`	Performs virus scanning operations.

The `davadmin vscan` command must be followed by the `scan` action.

Options for vscan Operation

Short Option	Long Option	Description
`-u` *adminuserid*	`--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` *file*	--clifile	File with bootstrap information that you use to specify command-line options so that they don't have to be entered at the command line. Each line in the bootstrap file is in the form `property=value`. 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 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` *host*	`--hostname`	Host name of the server. Optional, defaults to localhost.
`-p` *port*	`--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` *path*	`--secure`	Path to the truststore file used for a secure connection (HTTPS). Optional. Required if GlassFish is running in secure mode.
`-a` *account*	`--account`	The account information (email address) of the user to be scanned.
`-n` *name*	`--name`	The name of the target backendID.
`-B` *uri*	`--ldapbaseuri`	Base URI in LDAP.
`-R` *filter*	`--ldapfilter`	User search filter in LDAP. Default is `(objectClass=icsCalendarUser)`.
`-T` *time*	`--starttime`	Scan data entered into the server after this time. Format: *yyyymmddThhmmssZ*"
`-r`		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 varaiables.
`-h`	`--help`	Help for that particular operation. Optional.

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 attachements 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 -B "o=dav" -R "uid=caluser12"
davadmin vscan scan -u admin -B "o=dav" -R
"(|(uid=caluser222)(uid=caluser111))"
```

In this example, using just a uid filter might not be specific enough in the case of 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. Note that 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.

Virus Scan Logging

Virus scan activity for both online and CLI is printed 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. Owning 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.

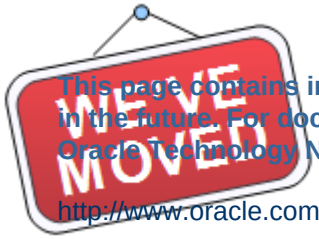
MTA Logging

See the MTA documentation 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.

Chapter 6. Best Practices for Backing Up and Restoring Databases in Calendar Server Deployments

Best Practices for Backing Up and Restoring Databases in Calendar Server Deployments



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

Topics:

Overview

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 need to 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 need to 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 Datbase 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 Calendar Server Unique Identifier.

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
- Using the davadmin db restore 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.

Using the davadmin db restore Command

Calendar Server 7 provides the `davadmin db restore` command to restore calendar server data. See db examples for more information.

ZFS Snapshots

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

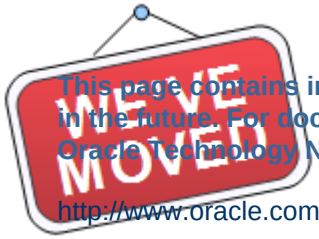
- MySQL Async Replication
  - Use MySQL replication to replicate the databases. See http://dev.mysql.com/doc/refman/5.1/en/replication.html.
- MySQL database dump
  - Use `mysqldump` to dump the databases for backup or transfer to another SQL server. See http://dev.mysql.com/doc/refman/5.1/en/mysqldump.html
- Point-in-time binlog backup and recovery.
  - The binary log files provide you with the information you need to replicate changes to the database. See 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 Backup and Recovery.

Chapter 7. Best Practices for Calendar Server

Best Practices for Oracle Communications Calendar Server



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This information describes best practice guidelines for deploying and administering Oracle Communications Calendar Server.

- Use root (`/`) as the default context URI for deployment.
  The purpose of this recommendation is for better client compatibility and ease of client configuration. If the base URI is not `/`, users are more likely to have to use their advanced client configuration and type a long cumbersome URL versus just being able to enter a host name. If you do deploy Calendar Server by using a root (`/`) URI context, you cannot deploy Convergence in the same instance of Glass Fish without moving the `/iwc_static/` files somewhere. The Convergence static files are deployed under the default root context and if another application is deployed to `/`, GlassFish is not be able to find the `/iwc_static/*` URLs, resulting in a 404 error immediately after after login. There are potential workarounds for this issue, but at this time create an additional domain in GlassFish that listens on a different port so that Convergence and Calendar Server 7 run in separate instances. See here for documentation on creating a new GlassFish domain. Additionally, for assistance in configuring clients for non-root URL configurations, see To Configure a CalDAV Account by Using Nonstandard or Demo Settings.
- Use the standard ports for HTTP and HTTPS protocols.
- Make sure to perform the post-installation steps to secure your deployment.
  See Calendar Server 7.0.4.15.0 Post Configuration.
- Create one document store per back-end host in a multiple back-end deployment.

Chapter 8. Managing Calendar Server JMS Destinations

Managing Calendar Server Java Messaging Server Destinations

This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This information 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 the Sun Java System Message Queue 4.1 Administration Guide.

Topics:

Calendar Server JMS Destinations Overview

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. For more information, see Using Calendar Server 7 Notifications.

Using the imqcmd Command with Calendar Server

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

To List a JMS Destination's Metrics

1. Change directories to the GlassFish Server `imq/bin` directory.
   For example:

```
cd /opt/SUNWappserver/imq/bin
```

2. List and display the metrics of the JMS topic used by the Calendar Server, `DavNotificationTopic`.
   For example:

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

To Purge 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.

- Run the following commands.

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

To Monitor Disk Utilization

- To monitor a physical destination's disk utilization, use the `imqcmd metrics` command with the `dsk` option.
  For example:

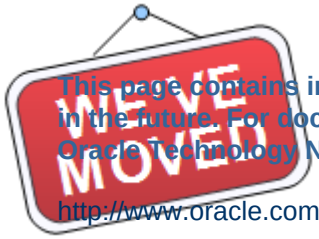
```
$ ./imqcmd metrics dst -t t -n DavNotificationTopic -m dsk -u admin
```

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`.

Chapter 9. Using Calendar Server 7 Notifications

Using Calendar Server 7 Notifications



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

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 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.

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

Note
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 the following information. 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.

Topics:

- More About Reminders (Alarms)
- More About Server Email Notifications
- Managing Notification Templates
- Writing a Java Messaging Service Consumer
- Configuring Presence Notifications

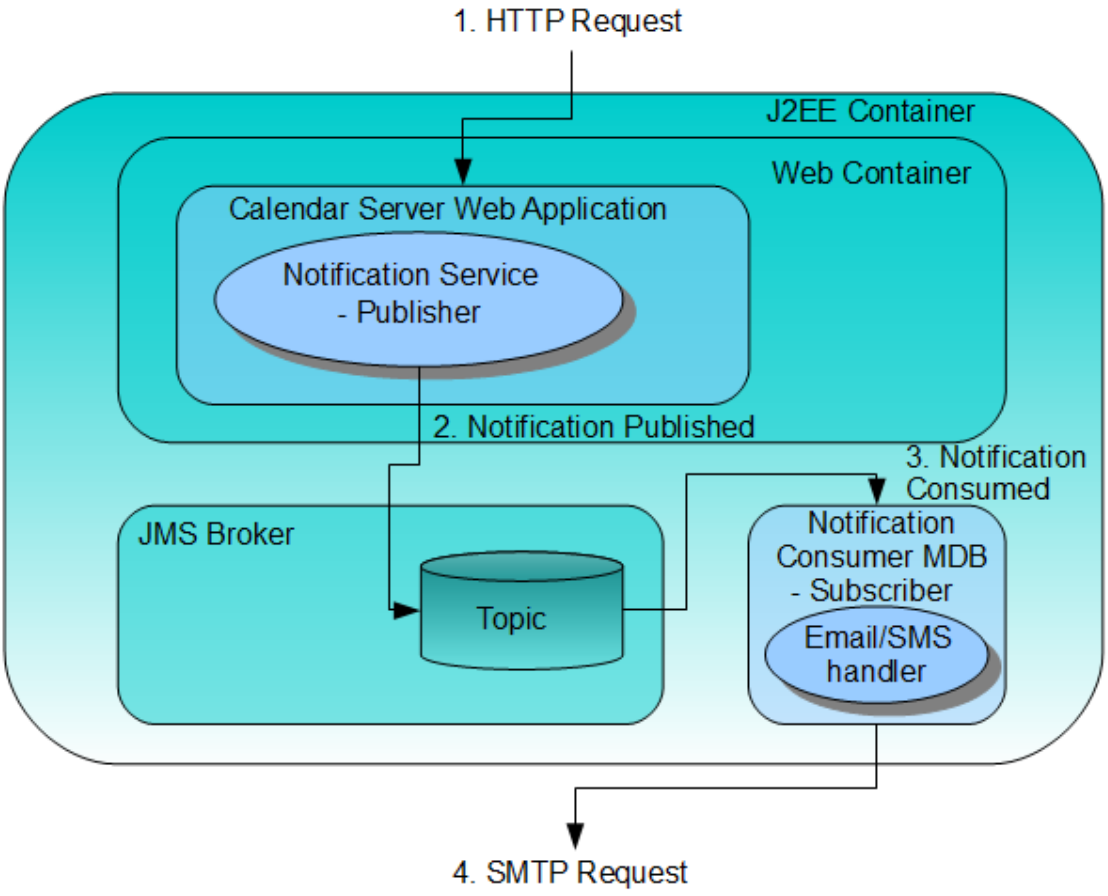
Notification Architecture Overview

Calendar Server 7 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 7 utilizes the built-in Java Messaging Service (JMS) in Oracle GlassFish Enterprise Server to communicate calendar data changes and calendar alarm triggers. Calendar Server 7 bundles a consumer program that "consumes" this information and sends email for certain subset of the notifications as detailed in Notification Types.

The following figure 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.

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.

More About Reminders (Alarms)

Calendar Server sends out emails 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.

Starting with **version 7.0.4.14.0**, 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.

For details on how to set alarms by using the store commands in the WCAP protocol, see WCAP 7.0 Server Alarms

More 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`).

Enabling Calendar Server Notifications

Calendar Server notifications are controlled by the configuration parameters described in the following table.

Notification Configuration Parameters

Name	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 has to 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 on Calendar Server configuration parameters, see Calendar Server 7 Configuration Parameters.

To Enable Notifications on an Account

1. To enable notifications for all accounts, use the davadmin command to set the davcore.autocreate.enableemailnotification to **true**.
   For example:

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

To Modify 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, see set_accountprops.wcap and davadmin account.

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.

The following table lists and describes 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.

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
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 **Introduced in Calendar Server 7.0.4.14.0.**	Event start for presence integration	UID, DTSTART, DTEND	Notification email is triggered if presence notification is enabled (`davcore.presence.enable=true`).
EVENT_END **Introduced in Calendar Server 7.0.4.14.0.**	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 **Introduced in Calendar Server 7 Update 2.**	An account was shared	None	An email is sent if additional permission was granted.
SHARE_CAL_COLLECTION **Introduced in Calendar Server 7 Update 2.**	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 **Introduced in Calendar Server 7 Update 3.**	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 7 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.

The following table summarizes 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.

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 upcomir event or todo.
Alarm	ALARM	sms.fmt	User's scheduling address	Recipients listed in alarm	SMS reminder for an upcomin event or todo. The SMS messa is a more concise message bu still sent by email.
Auto creation	AUTOCREATE	autocreate.fmt	User's scheduling address.	User's scheduling address	Notifies of auto creation of use home collection due to the arrival of the very first invit
Event Request Notification	SCHEDULE_ RECEIVE	request.fmt request_ recur.fmt	Organizer	Notification recipients	Notifies attendee of a new eve invitation or significant change to an invitation.
Todo Request Notification **Introduced in Calendar Server 7 Update 3.**	SCHEDULE_ RECEIVE	request_ todo.fmt request_recur_ todo.fmt	Organizer	Notification recipients	Notifies attendee of a new tod significant change to a todo.
Event reply Notification	SCHEDULE_ RECEIVE	reply_ request status.fmt	Attendee	Organizer	Notifies the organizer of the st of an invitation, when the status is of value 3.x and 4 which indicates some issues with the scheduling.

Todo Reply Notification **Introduced in Calendar Server 7 Update 3.**	SCHEDULE_ RECEIVE	reply_ request status_ todo.fmt	Attendee	Organizer	Notifies the organizer of the st of a todo, when the status is of value 3.x and 4.x, indicates some issues with the scheduling.
Event Cancel Notification	SCHEDULE_ RECEIVE	cancel.fmt cancel_ recur.fmt cancel_imip_ todo.fmt cancel_recur_ imip_todo.fmt	Organizer	Notification recipients	Notifies of a canceled event (t attendee).
Todo Cancel Notification **Introduced in Calendar Server 7 Update 3.**	SCHEDULE_ RECEIVE	cancel_ todo.fmt cancel_recur _todo.fmt	Organizer	Notification recipients	Notifies of a cancelled todo (to attendee).
Event Reply Notification	SCHEDULE_ RECEIVE	reply.fmt reply_ recur.fmt reply_imip_ todo.fmt reply_recur_ imip_todo.fmt	Attendee	Organizer	Notifies of the following reply scenarios: Notifies the event organizer th attendee accepted the invitation. Notifies the event organizer th attendee declined the invitation. Notifies the event organizer th attendee tentatively accepted the invitat
Todo Reply Notification **Introduced in Calendar Server 7 Update 3.**	SCHEDULE_ RECEIVE	reply_ todo.fmt reply_recur_ todo.fmt	Attendee	Organizer	Notifies of the following reply scenarios: Notifies the todo organizer tha attendee accepted the todo. Notifies the todo organizer tha attendee declined the todo. Notifies the todo organizer tha attendee tentatively accepted the todo.
Event Request Notification with iMIP Data	SCHEDULE_ 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 eve significant change to the event. The notification contains iCal information because the notification.dav.enableimipem configuration parameter has been set to true

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 (t attendee). The notification contains iCal information because the notification.dav.enableimipema 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: Notifies the event organizer th attendee accepted the invitation. Notifies the event organizer th attendee declined the invitation. Notifies the event organizer th attendee tentatively accepted the invitat The notification contains iCal information because the notification.dav.enableimipema configuration parameter has b set to true. |
iTIP Event Request	SCHEDULE_ ITIP	itip_ event request.fmt	Organizer	External attendee	iTIP scheduling: Announces a scheduling event or a significant change t event to an external attendee.
iTIP Todo Request **Introduced in Calendar Server 7 Update 3.**	SCHEDULE_ ITIP	itip_ todo request.fmt	Organizer	External attendee	iTIP scheduling: Announces a todo or a significant change to a todo external attendee.
iTIP Event Cancel	SCHEDULE_ ITIP	itip_ event cancel.fmt	Organizer	External attendee	iTIP scheduling: Notifies of a cancellation of an iTIP scheduling event to external attendee.
iTIP Todo Cancel **Introduced in Calendar Server 7 Update 3.**	SCHEDULE_ ITIP	itip_ todo cancel.fmt	Organizer	External attendee	iTIP scheduling: Notifies of a cancellation of an iTIP todo to an external attendee.
iTIP Event Reply	SCHEDULE_ ITIP	itip_ event reply.fmt	External attendee	Organizer	iTIP scheduling: Replies to an scheduling event. Attendee accepted the invitatio Attendee declined the invitatio Attendee +tentatively accepte invitation.

iTIP Todo Reply **Introduced in Calendar Server 7 Update 3.**	SCHEDULE_ ITIP	itip_ todo reply.fmt	External attendee	Organizer	iTIP scheduling: Replies to an todo. Attendee accepted the todo. Attendee declined the todo. Attendee tentatively accepted todo.
Share calendar account	SHARE_ ACCOUNT	share_ account .fmt	Sharer's email address	Sharee's email address	Notifies of a calendar account shared.
Share calendar collection	SHARE_CAL_ COLLECTION	share_ cal.fmt	Sharer's email address	Sharee's email address	Notifies of a calendar collectio being shared.

Notes:

- Notification recipients: A recipient list stored in the property, `SUN_NOTIFIRECIPIENT`. 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.

The following table shows all notification values and trinkets.

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
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 reply_requeststatus 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, as described in Notifications Architecture. The information in this section describes how to create your own consumer program.

All Calendar Server 7 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 Notifications 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 now 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 (JMSException 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();
}
} .........
```

Where to Go From Here
See Managing Calendar Server JMS Destinations to learn how to manage Java Messaging Server (JMS) destinations in Calendar Server by using the `imqcmd` command.

Configuring Presence Notifications

Starting with **version 7.0.4.14.0**, 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 infrastucture, which is also used to publish alarms for database changes.

To Configure 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 To Enable 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.

Chapter 10. Deploying Calendar Server on a GlassFish Server Cluster

Deploying Calendar Server on a GlassFish Server Cluster

This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This information describes how to deploy and configure Oracle Communications Calendar Server 7 on an Oracle GlassFish Server 3.1.2 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

This information assumes that you are familiar with the following tasks:

- Installing and configuring Calendar Server. See Communications Suite 7.0.6 Installation Guide for information about installing and configuring Calendar Server.
- Setting up clusters using Oracle GlassFish Server 3.1.2. See the following documents for more information:
  - Oracle GlassFish Server High Availability Administration Guide
  - Oracle GlassFish Server Quick Start Guide Chapter 2 Use Cases for Production Deployments

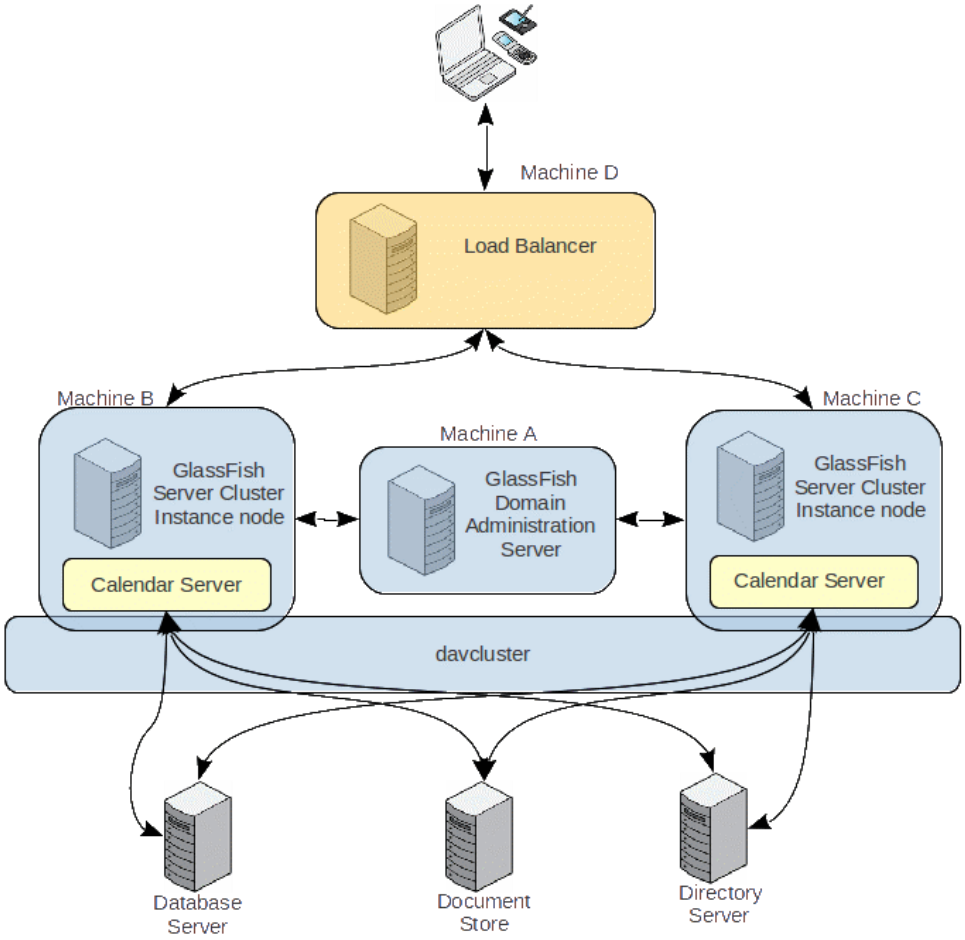
The hardware and software requirements for configuring Calendar Server in a GlassFish Server cluster are the same as configuring a non-cluster Calendar Server deployment. See Requirements for Communications Suite 7.0.6 for more information. You also need to install a load balancer for the cluster.

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

The following figure shows an example cluster architecture consisting of two GlassFish Server nodes.

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 to set up a GlassFish Server 3 cluster and load balancer.
  Instead of deploying 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, To Deploy 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 3 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 3 installation path, use the network share path of the GlassFish Server 3 DAS installation.
- If you plan to install and configure GlassFish 3 as the `root` user, you need to 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 3 on a network share location.

To Deploy Calendar Server on a Two-Instance GlassFish Server Cluster

1. Following the deployment example described previously, install Calendar Server on cluster node Machine B.
   You need to install the Unified Communications Suite distribution on the machine, run the `commpkg install` command, then select Calendar Server. For more information, see Installation Scenario - Calendar Server.
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 are running GlassFish 3 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 7.0.4.15.0 Initial Configuration.
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_p
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.

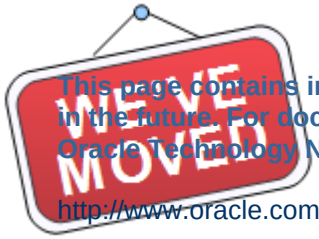
> **Note**
> If you install and configure Calendar Server on the DAS machine, when you run the `davadmin command`, you need the `-H` *cluster-node* option for the command to succeed, where *cluster-node* can be any one of the nodes in the cluster.

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.

Chapter 11. Making Calendar Server Highly Available

Making Oracle Communications Calendar Server Highly Available



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

Choices for making Calendar Server highly available include 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

This section describes how to deploy and configure Oracle Communications Calendar Server 7 on Sun GlassFish Enterprise Server 2.1.1 cluster. The GlassFish Enterprise Server cluster feature enables you to create highly available and scalable deployment architectures.

About This Deployment Example

This example assumes that you are familiar with the following tasks:

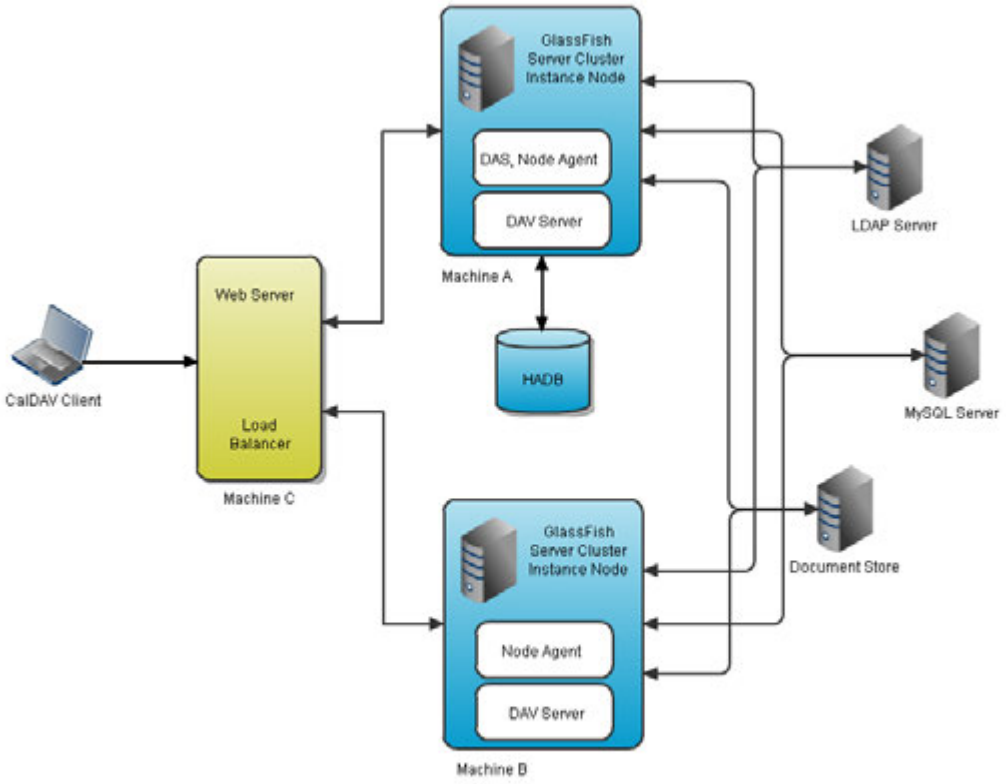
- Installing and configuring Calendar Server. See Communications Suite 7.0.6 Installation Guide for information about installing and configuring Calendar Server.
- Setting up clusters using Sun GlassFish Enterprise Server v2.1.1. See the following document for more information:
  - Sun GlassFish Enterprise Server v2.1.1 High Availability Administration Guide

Prerequisites and Deployment Example Architecture

The hardware and software requirements for configuring Calendar Server in a GlassFish Server cluster is the same as configuring a default Calendar Server deployment. However, additionally you need to 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.

The following figure shows the deployment architecture.

Calendar Server Cluster Deployment Architecture



Make sure that you have the following software:

- Sun GlassFish Enterprise Server 2.1.1 Patch05 (v2.1 Patch11) (9.1_02 Patch17) with HADB, Solaris: File based patch: http://sunsolve.sun.com/search/document.do?assetkey=1-21-128643-19-1
- Oracle iPlanet Web Server

Also, ensure that clocks are synchronized on the HADB hosts. See Synchronizing System Clocks for details.

To Configure a Calendar Server Cluster Deployment

For the following steps, refer to the Calendar Server Cluster Deployment Architecture figure.

On 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_semmni=10
set semsys:seminfo_semmns=60
set semsys:seminfo_semmnu=600
```

2. Reboot the system.

```
bash # reboot
```

   Consult Sun GlassFish Enterprise Server v2.1.1 High Availability Administration Guide Chapter 2 for more shared memory configuration information.

3. Run the Glassfish installer, `sges_ee-2_1_1-p05-solaris-sparc-ml.bin`, and select the "High Availability Database Server," and "Domain Administration Server and Administration Tool" in the Component Selection screen.
4. Start the HADB agent.

```
/opt/SUNWappserver/hadb/4/bin/ma-initd start
```

5. Create a HADB Management Domain.

```
hadbm createdomain --adminpassword=<password>
<machine-A-IP-address>
export HADBM_AGENT=machineA:1862
```

6. Give Node Supervisor Processes Root Privileges.

```
chmod u+s /opt/SUNWappserver/hadb/4/lib/server/clu_nsup_srv
chown root /opt/SUNWappserver/hadb/4/lib/server/clu_nsup_srv
```

7. Create a nodeagent 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 setup machine B and C, then return here for next step.
9. Create a 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 Admin 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 Installation Scenario - Calendar Server 7.0.4.15.0. 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.
    Edit the `/opt/SUNWappserver/domains/domain1/applications/j2ee-modules/davserver/WE` file to add `<distributable/>` under `<web-app>`. Create `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 content 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
asasmin start-cluster dav-cluster
```

On machine B where nodeagent resides

1. Run the GlassFish installer, `sges_ee-2_1_1-p05-solaris-sparc-ml.bin`, and select Command Line Administration Tool Only in the Component Selection screen.
2. Create a nodeagent 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
```

On machine C where Web Server 7 is used as the load balancer

1. Install Web Server 7 and create a symlink in the *websrv-base-dir* 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 installer, `sges_ee-2_1_1-p05-solaris-sparc-ml.bin`, 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 config 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 a SSL http listener for use with load balancer config 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
```

6. From machine A, export the server certificate and copy the `s1as.rfc` file to `/var/tmp` on machine C.

```
/opt/SUNWappserver/lib/certutil -L -a -n s1as -d
/opt/SUNWappserver/domains/domain1/config > /var/tmp/s1as.rfc
```

7. 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
```

8. Add the following lines to the `default.acl` in the `/var/opt/SUNWwbsvr7/admin-server/config-store/machineC/config/default.acl` file. Please 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_p
user = "anyone";
```

9. 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"
```

10. Finally, deploy the config to the instance and restart the server instance.

```
wadm deploy-config machineC
wadm restart-instance --config=machineC
```

Limitations

The GlassFish 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 async replication. You need to configure GlassFish Server to use the replication driver, as explained in Driver/Datasource Class Names, URL Syntax and Configuration Properties for Connector/J. See MySQL High Availability and Replication Information For Calendar Server 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 async replication to achieve a redundant, highly available Calendar Server deployment.

Back End High Availability: Oracle Data Guard

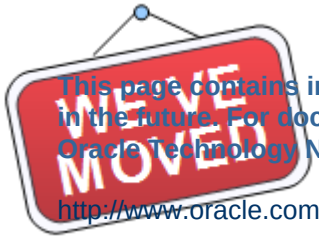
For information on maximizing Oracle Database availability by using Oracle Data Guard and Advanced Replication, see High Availability.

Document Store High Availability

Starting with Calendar Server 7 Update 2, you can deploy a highly available document store. See To Configure the Document Store for High Availability for details.

Chapter 12. MySQL High Availability and Replication Information For Calendar Server

MySQL High Availability and Replication for Calendar Server 7 Update 2 with MySQL 5.5



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

Oracle Communications Calendar Server (Calendar Server 7) relies on native capabilities offered by MySQL Server and GlassFish Server for a high-availability solution. This information 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, Chapter 14. High Availability and Scalability.

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:

- MySQL Server Asynchronous Replication Overview
- MySQL Server Asynchronous Replication Example
- MySQL Server Two-Way Replication Example
- Replication Synchronization Issues
- Using the Multi-Host Failover Feature of JDBC Connector/J in GlassFish for Some Automatic HA Functionality
- Test for MySQL Server Asynchronous Replication (Manual)
- Test for MySQL Server Two-Way Replication with Connector/J Failover

MySQL Server Asynchronous Replication Overview

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, resyncing nodes, and adding a node are all done manually. See Chapter 15. Replication, in the MySQL Server documentation, 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 is 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 7 Update 2 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 Chapter 15. Replication to complete the configuration.
   In this example, both nodes have a `log-bin` and do not have `log-slave-updates`. Some procedures in "Chapter 15. Replication" 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 be able to get all data from the other node's logs anyhow.

MySQL Server Two-Way Replication Example

This example is also 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 resync 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 in GlassFish for Some Automatic HA Functionality

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 resync and failback. 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 configuration parameters documentation: Chapter 21.3.5.1. Driver/Datasource Class Names, URL Syntax and Configuration Properties for Connector/J.
   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 need to address the situation quickly to control when `master1` comes back up and is resynced and new connections are made to it.
2. Recover `master1`.
   If `master1`'s data is damaged or lost, you need to reload `master1` from `master2` as 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 resync with `master2`. Other procedures are possible, but `master1` needs to resync before it receives new GlassFish Server connections.
4. Bring up `master1`.
   Verify it has resynced 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 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 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 in sync. 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 in sync 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. Servers

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 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 in sync. 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;
      ```

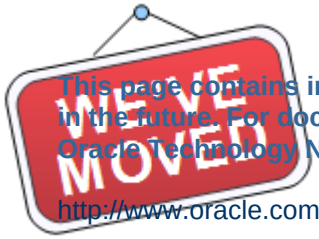
      Verify that the position is not increasing.

    - On both `Master1` and `Master2`:

      ```
      select count(*) from Resources;
      ```

Chapter 13. To Configure GlassFish Enterprise Server to Use a CA Signed Certificate for SSL

To Configure GlassFish Enterprise Server to Use a CA Signed Certificate for SSL



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

These instructions describe how to configure GlassFish Server to use a certificate issued by a Certification Authority (CA) to establish secure sessions through secure sockets layer (SSL) technology.

> **Note**
> For complete instructions on how to request and install a certificate for GlassFish Enterprise Server, refer to the official GlassFish Security documentation. The Mozilla certutil page also provides more information on the `certutil` command.

To configure GlassFish Server to use a CA signed certificate:

1. Change to the *app-svr-base*`/lib` directory and run the `certutil` command to generate the certificate request.
   In the following example:
   - The Organization (O) field is required and must be filled in with organization name exactly (without commas or periods).
   - The Country (C) field is required and must be filled in with the two-letter code for the country in which the server resides, for example,**US**.
     VeriSign does not accept CSR's with a country code of `U`. Instead you must use the country code `GB`.
   - The State (ST) field is required and should be filled in with the full name, not the abbreviated name, of the state or province in which the server resides.
   - The Common name (CN) field is required and should be filled in with the fully qualified domain name (FQDN) of the server. A fully qualified domain name means the full name of the server host. For example, `demoserver.central.example.com` is a fully qualified domain name, while `demoserver.central` is not. DNS must be able to resolve the FQDN.
   - The Locality (L) field is optional, but if filled in appears in the certificate. Use the name of the city in which the server resides.
   - The Organization Unit (OU) field is optional, but if filled in appears in the certificate. You can use it to differentiate between multiple SSL server instances running on the same host.

If you don't need to use it then leave it blank.

- `-o` specifies the file to be created.
- `-d` specifies the `config` directory of the GlassFish Server domain for which the certificate is being requested.
- `-a` specifies ASCII output.

```
Sample command to create a request:

./certutil -R -s "CN=demoserver.central.example.com,OU=Demo,
O=Example Inc,L=San Jose,ST=California,C=US" -o
/export/tmp/democert-app-server.req -d
/opt/SUNWappserver/domains/domain1/config -a

Enter Password or Pin for "NSS Certificate DB": <This is the
GlassFish Enterprise Server's administrative password.>

A random seed must be generated that will be used in the
creation of your key. One of the easiest ways to create a
random seed is to use the timing of keystrokes on a keyboard.

To begin, type keys on the keyboard until this progress meter
is full. DO NOT USE THE AUTOREPEAT FUNCTION ON YOUR KEYBOARD!


Continue typing until the progress meter is full:

|************************************************************|

Finished. Press enter to continue:


Generating key. This may take a few moments...
```

The Certificate Request resembles the following:

```
Certificate request generated by Netscape certutil
Phone: (not specified)

Common Name: <>
Email: (not specified)
Organization: <>
State: <>
Country: <>

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBzDCCATUCAQAwgYsxCzAJBgNVBAYTAklOMRIwEAYDVQQIEwlLYXJuYXRha2Ex
NEW CERTIFICATE REQUEST-----
```

2. Submit the certificate request and get the certificate approved by the Certificate Authority (CA). As there are many ways to have your certificate request approved, this step is left up to you. The approved certificate, in `pem` format, resembles the following:

```
Certificate:
```

```
Data:
Version: 3 (0x2)
Serial Number: 3 (0x3)
Signature Algorithm: md5WithRSAEncryption
Issuer: C=<>, ST=<>, L=<>, O=<>, OU=<>, CN=<>
Validity
Not Before: Sep 24 11:47:45 2009 GMT
Not After : Sep 24 11:47:45 2010 GMT
Subject: C=<>, ST=<>, O=<>, OU=<>, CN=<>
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
Modulus (1024 bit):
00:9f:f7:91:78:44:65:09:60:b1:e0:1c:60:04:3f:
88:36:4f:8e:56:95:fe:ec:a1:77:ce:06:a5:5b:6a:
78:b6:e4:d6:57:8d:03:4a:18:ab:0f:aa:fa:fc:7f:
a6:ad:6b:a3:8b:14:c8:18:cd:3c:c8:92:92:39:aa:
44:56:63:f3:88:a4:40:1f:9b:77:f1:52:eb:fe:3d:
60:f1:99:d9:d0:62:a4:9c:fe:de:fe:39:d0:2a:79:
e3:64:e5:59:70:d0:c2:36:3e:77:8e:b1:7b:66:e6:
18:9a:84:e4:ca:30:06:4f:ad:06:e7:6e:d7:9f:c2:
22:a9:e7:cc:ca:73:03:05:6f
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Basic Constraints:
CA:FALSE
Netscape Cert Type:
SSL Server
X509v3 Key Usage:
Digital Signature, Non Repudiation, Key Encipherment
Netscape Comment:
OpenSSL Generated Certificate
X509v3 Subject Key Identifier:
A0:0B:AC:87:D6:29:DA:AD:1C:EC:82:85:33:C3:BC:09:E0:25:B4:2B
X509v3 Authority Key Identifier:
keyid:10:C9:2C:31:AC:4A:A5:F1:08:0B:28:15:96:3F:1D:1A:71:33:E7:47
DirName:/C=<>/ST=<>/L=<>/O=<>/OU=<>/CN=<>
serial:CA:0F:64:A4:89:4F:2C:21

Signature Algorithm: md5WithRSAEncryption
51:5e:8b:08:bc:fa:9d:21:be:c6:1e:6b:30:d4:7d:a7:ef:86:
28:1b:6f:e4:66:c0:69:64:14:19:07:e9:5d:ad:a0:bb:ce:1a:
3c:27:81:30:3e:65:46:57:60:4c:a6:8c:76:a2:2e:14:4f:12:
35:a2:04:e9:36:31:2b:4e:c1:63:be:89:db:30:b8:01:78:c8:
39:0d:7d:2c:87:c9:cb:72:5d:1e:88:87:e7:ce:0f:b8:45:8a:
d7:66:a1:5a:d0:bf:3a:67:bd:a2:b9:65:21:f5:e5:db:8b:cf:
c0:39:18:66:96:79:7e:96:b3:21:00:c5:4a:24:bb:42:ad:52:
d4:f1
-----BEGIN CERTIFICATE-----
MIID7jCCA1egAwIBAgIBAzANBgkqhkiG9w0BAQQFADCBpTELMAkGA1UEBhMCSU4x
EjAQBgNVBAgTCUthcmFudGFrYTESMBAGA1UEBxMJQmFuZ2Fsb3JlMR8wHQYDVQQK
ExZTdW4gTWljcm9zeXN0ZW1zIEluZGlhMRQwEgYDVQQLEwtDb21tc1FBLUlFQzET
MBEGA1UEAxMKQ0EtQ29tbXNRQTEiMCAGCSqGSIb3DQEJARYTYWRtaW5AaW5kaWEu
c3VuLmNvbTAeFw0wOTA5MjQxMTQ3NDVaFw0xMDA5MjQxMTQ3NDVaMHcxCzAJBgNV
BAYTAklOMRIwEAYDVQQIEwlLYXJuYXRha2ExHjAcBgNVBAoTFVN1biBNaWNyb3N5
c3RlbXMgSW5jLjEOMAwGA1UECxMFQ29tbXMxJDAiBgNVBAMTG2NvbXMtMTUyeC0x
```

```
NjguaW5kaWEuc3VuLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAn/eR
eERlCWCx4BxgBD+INk+OVpX+7KF3zgalW2p4tuTWV40DShirD6r6/H+mrWujixTI
GM08yJKSOapEVmPziKRAH5t38VLr/j1g8ZnZ0GKknP7e/jnQKnnjZOVZcNDCNj53
jrF7ZuYYmoTkyjAGT60G527Xn8IiqefMynMDBW8CAwEAAaOCAVkwggFVMAkGA1Ud
EwQCMAAwEQYJYIZIAYb4QgEBBAQDAgZAMAsGA1UdDwQEAwIF4DAsBglghkgBhvhC
AQ0EHxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYDVR0OBBYEFKAL
rIfWKdqtHOyChTPDvAngJbQrMIHaBgNVHSMEgdIwgc+AFBDJLDGsSqXxCAsoFZY/
HRpxM+dHoYGrpIGoMIGlMQswCQYDVQQGEwJJTjESMBAGA1UECBMJS2FyYW50YWth
MRIwEAYDVQQHEwlCYW5nYWxvcmUxHzAdBgNVBAoTFlN1biBNaWNyb3N5c3RlbXMg
SW5kaWExFDASBgNVBAsTC0NvbW1zUUEtSUVDMRMwEQYDVQQDEwpDQS1Db21tc1FB
MSIwIAYJKoZIhvcNAQkBFhNhZG1pbkBpbmRpYS5zdW4uY29tggkAyg9kpIlPLCEw
DQYJKoZIhvcNAQEEBQADgYEAUV6LCLz6nSG+xh5rMNR9p++GKBtv5GbAaWQUGQfp
Xa2gu84aPCeBMD5lRldgTKaMdqIuFE8SNaIE6TYxK07BY76J2zC4AXjIOQ19LIfJ
y3JdHoiH584PuEWK12ahWtC/Ome9orllIfXl24vPwDkYZpZ5fpazIQDFSiS7Qq1S
1PE=
-----END CERTIFICATE-----
```

3. Once you have obtained your CA signed and approved SSL server certificate, install it by using the `certutil` command.

```
./certutil -A -n TestSSLCert -t "P,u,u" -d
<app-svr-base>/domain-config directory -i
/space/smime/ssl-certs/certs/<>.pem
```

   `TestSSLCert` is an example of the certificate nickname that you need to provide in the GlassFish Enterprise Server configuration.

4. Verify that the certificate was installed.

```
./certutil -L -d <app-svr-base>/domain-config

(The output is similar to the following.)

verisignclass1ca T,c,c
thawtepersonalpremiumca T,c,c
baltimorecodesigningca T,c,c
TestSSLCert T,c,c
verisignclass2g2ca T,c,c
verisignclass3g3ca T,c,c
entrustglobalclientca T,c,c
entrustsslca T,c,c
verisignclass3g2ca T,c,c
thawtepremiumserverca T,c,c
entrust2048ca T,c,c
valicertclass2ca T,c,c
gtecybertrust5ca T,c,c
equifaxsecureebusinessca1 T,c,c
verisignclass1g3ca T,c,c
godaddyclass2ca T,c,c
thawtepersonalbasicca T,c,c
verisignclass1g2ca T,c,c
verisignclass2g3ca T,c,c
equifaxsecureca T,c,c
entrustclientca T,c,c
verisignserverca T,c,c
geotrustglobalca T,c,c
equifaxsecureebusinessca2 T,c,c
s1as u,u,u
sslCACert T,c,c
verisignclass3ca T,c,c
verisignclass2ca T,c,c
sslcert1 pu,pu,pu
gtecybertrustglobalca T,c,c
entrustgsslca T,c,c
thawtepersonalfreemailca T,c,c
thawteserverca T,c,c
baltimorecybertrustca T,c,c
starfieldclass2ca T,c,c
equifaxsecureglobalebusinessca1 T,c,c
TestSSLCert P,u,u
```

5. Log in to the GlassFish Server administration console and change the SSL certificate nickname. (This example uses `TestSSLCert`.)
   If you want the JMX connector to also use the new certificate, go to configuration->server-config->admin service->system, select the SSL tab, and change the SSL certificate nickname to be the new one you want to use.
6. Run any `asadmin` command to prompt you to accept the new certificate, for example:

```
cd <app-svr-base>/bin
./asadmin list-jms-hosts
Do you trust the above certificate y? yes
```

Accepting the certificate updates your `.asadmintruststore` file.

7. Restart the GlassFish Enterprise Server domain, for example:

```
cd <app-svr-base>/bin
./asadmin stop-domain domain1;./asadmin start-domain domain1
```

8. If you are installed the CA certificate after running the `init-config` command, copy the `.asadmintruststore` file under the root directory to Calendar Server's `config` directory, by default `/var/opt/sun/comms/davserver/config`.

Chapter 14. Calendar Server and Directory Server Integration

Calendar Server and Directory Server Integration

This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This information describes how Calendar Server uses Directory Server, what LDAP schema is necessary, and what specific object classes and attributes are required. Where appropriate, differences between Calendar Server 7 and Calendar Server 6 are described.

Topics:

How Does Calendar Server Use Directory Server?

Calendar Server uses Directory Server to store and access LDAP data for individual users, groups, and domains. The LDAP data is used in a variety of ways, including:

- Authenticating users
- Determining a user's, group's, or domain's status
- Administering domain level access control
- Retrieving important information, such as the user's unique ID, email address, calendar store back-end ID, and so on
- Retrieving default values for various user properties

You need to install and provision Directory Server prior to installing and configuring Calendar Server (and for that matter, prior to installing and configuring most Unified Communications Suite component products). You also need to prepare the Directory Server LDAP schema by running the `comm_dssetup.pl` script, which is provided as part of the Unified Communications Suite installer. This script prepares the LDAP directory by adding the necessary Communications Suite schema. For more information, see Communications Suite 7.0.6 Directory Server Setup Script (comm_dssetup.pl).

For information on installing and configuring Calendar Server, see Installation Scenario - Calendar Server 7.0.4.15.0. In addition, you should perform post-configuration steps to enable certain levels of LDAP searches required by for calendar searches and subscription, and to enable a secure connection between Calendar Server and Directory Server. See Calendar Server 7.0.4.15.0 Post Configuration for more information.

LDAP Schema Used by Calendar Server

To understand the schema that is used by Calendar Server, refer to the Communications Suite Schema Reference. Some additional LDAP object classes were added specifically to support Calendar Server 7.

The `davcore.ldapattr.*` configuration parameters govern the default attributes and object classes used by the Calendar Server 7. Default values are set based on the Communications Suite schema. See Calendar Server 7 Configuration Parameters for details.

Calendar Server 6 Versus Calendar Server 7 LDAP Schema Usage

This section contains the following topics:

- Similarities and Differences Between Calendar Server 6 and Calendar Server 7 Schema
- Deprecated Attributes and Object Classes in Calendar Server 7

Similarities and Differences Between Calendar Server 6 and Calendar Server 7 Schema

In Calendar Server 6, a user's LDAP entry requires the `icsCalendarUser` object class for calendaring to work for that user. Unless disallowed by the server configuration, Calendar Server 6 automatically provisions users by adding the `icsCalendarUser` object class to LDAP, creating the database entries for the user, and thus enables calendaring for the user upon first login or invite. Basic provisioning of the user entry in the LDAP directory is required for this to work.

Unlike Calendar Server 6, Calendar Server 7 does not add or modify LDAP data nor does it require the presence of a particular LDAP object class like `icsCalendarUser`. Unless disallowed by the server configuration, Calendar Server 7 automatically creates the database entries for a user upon initial login or invite, thus enabling the user. Basic provisioning of the user entry in the LDAP directory is required for this to work.

In Calendar Server 6, both the `icsStatus` or `icsAllowedService` attributes can be used to enable or disable calendaring service. In Calendar Server 7, only the `icsStatus` attribute is used to enable or disable the service. For more information, see icsStatus.

In Calendar Server 6, the `icsDWPHost` attribute specifies the user's back-end data store. In Calendar Server 7, it is the `davstore` attribute.

Deprecated Attributes and Object Classes in Calendar Server 7

The following Communications Suite LDAP attributes used by Calendar Server 6 are no longer used by Calendar Server 7. These attributes are from the `icsCalendarUser`, `icsCalendarResource`, and `icsCalendarDomain` object classes:

- `icsAllowedServiceAccess`
- `icsCalendar`
- `icsCalendarOwned`
- `icsDefaultSet`
- `icsDWPHost`
- `icsExtended`

- icsExtendedUserPrefs (but still used by Convergence)
- icsFirstDay (but still used by Convergence)
- icsFreeBusy
- icsGeo
- icsPartition
- icsPreferredHost
- icsQuota
- icsSet
- icsSubscribed
- icsTimezone
- nswcalDisallowAccess
- aclGroupAddr
- icsAlias
- icsCapacity
- icsContact
- icsExtended
- icsExtendedResourcePrefs
- icsSecondaryowners
- icsDefaultacl
- icsAllowRights
- icsAnonymousAllowWrite
- icsAnonymousCalendar
- icsAnonymousDefaultSet
- icsAnonymousLogin
- icsAnonymousSet
- icsDWPBackEndHosts
- icsExtendedDomainPrefs
- icsDefaultAccess
- icsDomainAllowed
- icsDomainNotAllowed
- icsMandatorySubscribed
- icsMandatoryView
- icsRecurrenceBound
- icsRecurrenceDate
- icsSessionTimeout
- icsSourceHtml

The icsCalendarGroup, icsAdministrator, and icsCalendarDWPHost object classes are also no longer used by Calendar Server 7.

This information can also be found in the Schema Reference.

Special LDAP Object Classes for Calendar Server

This section describes the following object classes:

- icsCalendarUser
- icsCalendarResource
- icsCalendarDomain
- davEntity
- groupofUniqueNames

icsCalendarUser

This object class defines a user entry with calendar service. While addition of this object class for calendar users is recommended, it is not mandatory. By default, a user entry that contains the deployment's unique ID (a unique identifier in the form of an LDAP attribute whose value is used to map each calendar account to a unique account in the Calendar Server database), and uid, password, and

mail attributes, works as a valid calendar user entry. The icsCalendarUser object class gives you more flexibility in enabling and disabling calendaring for provisioned users by using the icsStatus attribute. To require addition of this object class for a user entry to be considered as a valid calendar user entry, set the value of the Calendar Server configuration parameter davcore.ldapattr.userobject to **icsCalendarUser**.

See Defining Valid Calendar Users for more information on using the davcore.ldapattr.userobject configuration parameter.

icsCalendarResource

This object class is required to define a resource entry with calendar service. A resource in the scheduling context is any shared entity that can be scheduled by a calendar user, but does not control its own attendance status.

To use a different object class instead, set the custom value for the davcore.ldapattr.resourceobject configuration parameter. The icsCalendarResource object class provides the owner attribute that by default specifies the owner of the resource.

For more on resource accounts see Administering Resource Calendars.

icsCalendarDomain

This object class defines a domain entry with calendar-enabled users. It enables setting domain level access control for calendar users' cross-domain access by using its icsDomainNames and icsDomainAcl attributes. See Managing Domain Access Controls for details. You can use the icsStatus attribute in this object class to enable or disable calendaring service for an entire domain.

davEntity

This is a common object class that you can add to any of the others such as icsCalendarUser, icsCalendarResource, or icsCalendarDomain, to include some commonly needed attributes. This object class includes the davStore attribute that defines the back-end calendar store host for a user, or an entire domain in a multiple back-end setup. It also includes the davUniqueID attribute, introduced in Calendar Server 7 Update 3, used to specify a globally unique ID for any LDAP entry.

groupofUniqueNames

The Communications Suite schema does not define any specific group object classes. By default, Calendar Server considers entries with groupofuniquenames, groupofurls, or inetmailgroup object classes as groups. This is defined by the Calendar Server davcore.ldapattr.groupobject configuration parameter.

Group entries make it easy to invite a whole set of users, or to set access control for a whole set of users. Groups are identified by using their mail attribute value. To determine members of a group, Calendar Server uses LDAP attributes defined by the following configuration parameters:

- davcore.ldapattr.dngroupmember: Defines members by using their distinguished name (dn); default value is uniquemember
- davcore.ldapattr.mailgroupmember: Defines members by using their email address; default value is mgrprfc822mailmember
- davcore.ldapattr.urlgroupmember: Defines members by using a URL; default value is memberurl

Note
Both `uniquemember` and `memberurl` are considered for checking ACLs while `mgrprfc822mailmember` is not. Thus, members of a `uniquemember` or `memberurl` group can read, write, subscribe, and so forth, to a calendar depending on the ACL. In the majority of cases, use `uniqueMember` when configuring a calendar group. `mgrprfc822mailmember` is application specific and useful to invite users that are not part of your deployment.

For more information, see Inviting LDAP Groups and Dynamic Group ACLs.

Important Attributes for Calendar Server

This section contains the following topics:

- Unique ID Attribute
- Mail Attribute
- Status Attribute
- Store Id Attribute
- External Authentication Attributes
- Indexing the Directory Server

Unique ID Attribute

This attribute defines the unique value used as the database identifier for each account. This value is used internally to identify a user in other user's access control entries, subscription entries, and so on. The attribute chosen as the unique ID attribute must be present in all user, group, and resource LDAP entries for the deployment.

The `nsUniqueId` attribute was initially chosen to be used by Calendar Server for a unique ID. However, the `nsUniqueId` attribute is no longer recommended because the value cannot be preserved if the LDAP entry needs to be moved or recreated. The `nsUniqueId` attribute does guarantee that the value is unique throughout the Directory Server instance but unfortunately that is not enough. The recommendation is now to choose an attribute whose value can be guaranteed to be unique. In addition, you should add checking to your provisioning process and tools to make sure that the chosen unique ID attribute is defined for every entry and is indeed unique.

Starting in **Calendar Server 7 Update 3**, you can use the `davUniqueId` attribute to define a unique ID for any LDAP entry. It is recommended that it be used as the value of the `davcore.uriinfo.permanentuniqueid` parameter. This parameter defines which attribute Calendar Server uses as the unique identifier for users, groups, and resources. Calendar Server 7 Update 3 provides the `populate-davuniqueid` tool to set this attribute's value for all existing LDAP entries.

In the Calendar Server data base, the unique identifier value is case sensitive. If you need to move or recreate the corresponding LDAP entry, make sure to retain the case of the value as is. However, because the value is considered as case insensitive for LDAP comparisons, do not create a unique identifier value for another user or resource entry by just changing the case of the value.

Mail Attribute

By default, Calendar Server uses the `mail` attribute. This attribute is used for a number of important functions, such as the default account identifier for the `davadmin` tool, the address to be used for scheduling, the default address for email notifications, and so on. The Calendar Server database also stores the value of this attribute. If scheduling is performed by using other mail attributes, such as `mailAlternateAddresses`, the values are canonicalized to the mail value before storing in the

database. If you change the value of this attribute, you need to perform additional steps to changing it in the LDAP directory. See To Change a User's Email Address in the Calendar Server Database command for more information.

This attribute is also required for resource entries. Though resources do not receive email, Calendar Server uses this address value to identify and schedule the resource. Hence this value should be unique to the resource, and other values such as the owner's email address should not be used. For more information, see To Manage a Resource Calendar's Mailbox.

Status Attribute

By default, Calendar Server uses the `icsStatus` attribute. This attribute value can be used to enable or disable a user for calendaring services. Absence of this attribute or a value of `active` indicates active status. Values of `removed`, `deleted`, or `inactive` disable the service. Any other value may also enable the service but is not recommended.

Starting with Calendar Server 7 Update 3, if a calendar account's LDAP `icsStatus` attribute is populated and is not set to `active`, the account is not searched nor are any results fetched for that account when running Calendar Server `davadmin` or WCAP commands. That is, Calendar Server returns search results only for active accounts and does not return unusable data such as inactive calendars.

Store Id Attribute

By default, Calendar Server uses the `davStore` attribute, which indicates the back-end host that stores a user's data if the deployment is configured for multiple back ends as described in Multiple Backend Setup Guide. Also, if the deployment uses the Convergence client or has a Calendar Server 6 and Calendar Server 7 co-deployment setup, this attribute is required. In the latter two cases, if there are no multiple Calendar Server 7 back ends, then the value used by default must be **`defaultbackend`**.

If your deployment consists of multiple back ends, you must use one of the `store.dav.`*`xx`*`.backendid` values previously configured to set as the value for the `davStore` attribute. You must explicitly provision this attribute on your back-end configuration. Neither Calendar Server nor Delegated Administrator auto-provision this attribute.

Once a user is active, do not change the `davStore` attribute value. To move users to another back-end store, see To Move Calendar Users to a New Back-End Database.

External Authentication Attributes

> The attributes described in this section were added in **Calendar Server 7 Update 3** to support authentication against an external Directory Server. For information on configuring external authentication for Calendar Server, see Configuring External Authentication.

- `externalAuthPreUrlTemplate`: This attribute is used for authentication using external Directory Servers. It is used to set the LDAP URL that defines how users must be searched for in the external Directory Server against which authentication is performed. This attribute belongs to the `inetDomainAuthInfo` object class.
- `externalAuthPostUrlTemplate`: This attribute is used for finding the internal Directory Server entry for a user authenticated by using external Directory Servers. It is used to set the LDAP URL that must be used to map the external Directory Server authenticated user to a user in the internal Directory. This attribute belongs to the `inetDomainAuthInfo` object class.

Indexing the Directory Server

Certain attributes need to be indexed for presence, equality, or sub-string search for better performance. Some of this indexing is done by Directory Server itself or by the `comm_dssetup` script. See the following table to decide if you have the correct indexes. Use the `dsconf list-indexes` command to list the current indexes (See dsconf list-indexes doc). Use the `create-index` command for creating new indexes. (See dsconf create-index docs).

Attributes for Indexing

Attribute	Index
associatedDomain	eq, pres
cn	eq, pres, sub
inetDomainBaseDN	eq, pres
mail	eq, pres, sub
mailAlternateAddress	eq, pres
member	eq
memberOf	eq, pres, sub
objectClass	eq
owner	eq
sunPreferredDomain	eq, pres
uid	eq
uniqueMember	eq

If you are concerned about LDAP performance, check the LDAP access logs for the entry `notes=U}` to find unindexed searches. For example:

```
[15/Feb/2012:06:45:11 -0800] conn=110405 op=13 msgId=21 - SRCH
base="o=example.com,o=dav" scope=2
filter="(|(uid=cal*)(cn=*cal*)(mail=*cal*))" attrs="cn davStore
icsStatus mail mailAlternateAddress nsUniqueId owner preferredLanguage
uid objectClass isMemberOf uniqueMember memberURL mgrpRFC822MailMember
kind"
[15/Feb/2012:06:45:17 -0800] conn=110405 op=13 msgId=21 - RESULT err=4
tag=101 nentries=1000 etime=6 notes=U
```

Special Calendar Server Users and Groups

Calendar Server creates two special users during the initial configuration, one for serving as a proxy to the Directory Server, and the other for acting as the administrator ID for Calendar Server itself.

Topics in this section:

- About the Calendar Server Proxy User
- Calendar Server Administrator ID
- LDAP Updates During Configuration

About the Calendar Server Proxy User

Calendar Server uses a proxy user to bind to the Directory Server when making requests. This user belongs to the Calendar End User Administrators Group, which has proxy rights. This special Calendar Server user makes Directory Server requests on behalf of the end user for whom the request is being carried out. The proxy process takes into account the Directory ACIs for that particular end user. The DN (Distinguished Name) of this newly created user is added to the server configuration as the `base.ldapinfo.ugldap.binddn`, for example:

- Server configuration option:

```
base.ldapinfo.ugldap.binddn=uid=cal-admin-sca-peanut.example.com-2011
```

- Sample LDIF file used to create that user and group in the Directory Server:

```
dn: cn=Calendar End User Administrators Group,
 ou=Groups, o=isp
changetype: modify
add: uniqueMember
uniqueMember: uid=cal-admin-sca-peanut.example.com-20111102184916Z,
 ou=People, o=example.com,o=isp

dn: o=isp
changetype: modify
add: aci
aci: (target="ldap:///o=isp")
 (targetattr="*")
 (version 3.0; acl "Calendar Server End User Administrator Proxy
Rights - product=davserver,schema 2
support,class=admin,num=1,version=1"; allow (proxy)
groupdn="ldap:///cn=Calendar End User Administrators Group,
ou=Groups, o=isp";)
```

Calendar Server Administrator ID

The other special user that Calendar Server creates is the Service Administrator user, by default, the `calmaster` user. The Calendar Server `base.ldapinfo.serviceadminsgroupdn` configuration option specifies the name of an LDAP group for which membership in the group means super user privileges as far Calendar Server is concerned. The `calmaster` user belongs to this Service Administrators group and thus has full access to all users' calendaring information. Convergence always authenticates and proxies on behalf of the end user to Calendar Server as `calmaster`. You can add additional users to the Service Administrators group if you require that more users have these privileges.

LDAP Updates During Configuration

The Calendar Server initial configuration program, `init-config`, creates the LDAP tree's base containers and the default domain if it does not already exist. It also sets up the special administrative groups and users previously mentioned. ACIs are also added to give read, search, and proxy rights to the administrative users. An ACI is also added to give read and search ACIs to all authenticated users.

You may also want to enable users to search for other users whose calendars they can subscribe to. This requires permission to search for other users in LDAP. If such a permission is not already granted, you need to add an ACI to the user/group suffix in Directory Server. For more information, see Adding LDAP Access Control for Calendar Server Features.

How Calendar Server Authenticates with Directory Server

The following information describes how Directory Server authenticates a Calendar Server user by using basic user ID and password authentication.

Topics in this section:

Background Information

Most requests to Directory Server are made against the user/group LDAP server (as defined in the `base.ldapinfo.ugldap.*` Calendar Server configuration parameters). Only the bind request to check the user's credentials is issued against the authentication LDAP server (as defined in the `base.ldapinfo.authldap.*` configuration parameters).

When an external directory is configured for the user's domain, a search is first issued against that external directory. The bind request to check the user's credentials is also issued against that external directory. Most of the information discovered through this process (including a hash of the logged-in user passwords) is cached in memory (with a configurable time-to-live value).

First Step: Domain Lookup on UG Server

Consider the following search request:

```
Search Request: base DN = o=dav, scope = 2, filter =
&(objectClass=sunManagedOrganization)(|(sunPreferredDomain=example.com)(ass
```

The domain lookup is performed as follows:

1. The base DN (the top-level of the Directory Server tree) for this search is determined by the `base.ldapinfo.dcroot` Calendar Server configuration parameter.
2. The domain value is extracted from the user-provided user ID by using the `base.ldapinfo.loginseparator` Calendar Server configuration parameter.
   For example, if `arnaudq@example.com` is the user ID and the login separator character is"@," the domain is determined to be `example.com`).
3. Alternatively, if no login separator is found in the user ID, the domain is assumed from the `base.ldapinfo.defaultdomain` Calendar Server configuration parameter. An LDAP lookup is still issued in that case.

Second Step: User Authentication

The default behavior is to use the user/group server and the authentication server for authentication.

If the domain entry retrieved in the previous step has an `externalAuthPreUrlTemplate` LDAP attribute, the user authentication is performed against an external directory.

User Authentication Default Behavior

To illustrate how the user (user/group server) authentication is performed, consider the following search request:

```
Search Request: base DN = o=example.com,o=dav, scope = 2, filter =
uid=arnaudq
```

The base DN for this search is extracted from the domain entry retrieved during the initial domain lookup (with some differences between Schema 1 and Schema 2). Next, the search filter is constructed either from the `inetDomainSearchFilter` LDAP attribute of the domain entry or from the `base.ldapinfo.searchfilter` configuration parameter. In both cases, the configuration value is based on the following template:

- %U--Name part of the login name (that is, everything before the login separator stored in the servers configuration)
- %V--Domain part of the login string
- %o--Original login ID entered by the user (for example, `uid=%U`)

This lookup should return an LDAP entry containing, in addition to a DN, the list of attributes defined by the `base.ldapinfo.userattrs` configuration parameters (the default is `mail` and `ismemberof`).

Credentials Verification: Authentication Server

Consider the following bind request:

```
Bind Request: version = 3, DN =
uid=arnaudq,ou=people,o=example.com,o=dav
```

Unlike the other requests, this one is made by using the authentication LDAP Server. The DN used is the one retrieved during the authentication user lookup step and the password is the one provided by the end user. Assuming that the bind request is successful, the LDAP entry returned during the authentication user lookup step is used to:

1. Check whether the corresponding user belongs to the administrative group.
2. Extract the mail attribute of the entry (the attribute name itself is configurable through the `davcore.ldapattr.mail` Calendar Server configuration parameter). Pay attention to this configuration parameter, as it is also used in different other places.

The authentication ends at this point. A set of principal objects containing the mail value (and optionally administrator privileges) is constructed as a result.

Alternative Behavior Using External Directory Authentication

The `externalAuthPreUrlTemplate` LDAP attribute contains an LDAP URL defining what external directory server to use, and the type of search to issue to find the user within that directory. The host name part of the URL contains some identifier pointing to an actual LDAP Pool defined in the server configuration. The search filter in that LDAP URL is also a template, containing the same keywords as previously described (%o, %U, and %V). Once the entry is found, the credential verification (as previously described) is performed by using that entry's DN. The process of configuring external authentication for a particular domain is described in Configuring External Authentication.

Third Step: User Info Lookup on UG Server

Consider the following search request:

```
Search Request: (base DN = o=example.com,o=dav, scope = 2, filter =
|(mail=arnaud.quillaud@example.com)(mailalternateaddress=arnaud.quillaud@ex
```

While not exposed, externally, the set of (JAAS) authentication modules is configurable. The domain lookup, authentication user lookup, and credentials verification steps correspond to the basic LDAP Auth module but other modules can be plugged in. The contract between an Auth Module and the server is that after successful authentication, the mail attribute of the principal must be made available to the server. Once the mail attribute value is extracted, a new lookup based on that email address is issued. This lookup is part of the core server processing.

The base DN for this search is extracted from the domain part of the email address (as described in First Step: Domain Lookup on UG Server although this information is already most likely cached at that point). The search filter is constructed from the `davcore.uriinfo.emailsearchfiltertemplate` configuration parameter (the default value is `|(mail=%s)(mailalternateaddress=%s)`). The returned entry is used to retrieve user information. The list of requested attributes is controlled by the `davcore.uriinfo.subjectattributes` configuration parameter.

Alternative Behavior Using External Directory Authentication

Once the external directory has authenticated the user, a mapping needs to happen between that entry and the corresponding entry in the Communications Suite directory.

If all external directory user entries have a `mail` attribute value corresponding to their Communications Suite equivalent entry, no further configuration is required that is different from the internal authentication setup described previously. At the second step, the list of attributes to be retrieved must simply include the `mail` attribute.

If no such direct mapping exists, an LDAP search is issued against the internal Communications Suite directory. This second search is defined by the `externalAuthPostUrlTemplate` domain entry attribute. Like `externalAuthPreUrlTemplate`, the `externalAuthPostUrlTemplate` attribute contains an LDAP URL defining what type of search to issue to find the user within that directory. In this case, a server name is not required and should not be defined, as the lookup is done against the Communications Suite directory. The search filter can be a template or fixed filter. They can contain the patterns previously mentioned, as well as the %A *attributename*_ pattern, which is substituted with the first LDAP attribute value retrieved from the external authentication directory in the second step. This pattern may appear multiple times with different attribute names. Of course, those attributes must be part of the list of attributes to retrieve in the LDAP URL.

For a given login id and domain, those patterns are substituted for their actual values before the search is issued. The search is then conducted, the correct entry is found, and the authentication is considered successful.

See the Communications Suite Schema Reference for more information on the `externalAuthPostUrlTemplate` and `externalAuthPreUrlTemplate` attributes.

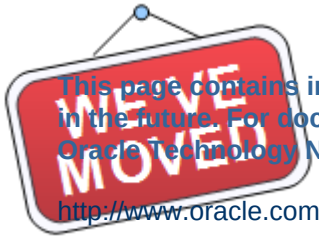
Sample Calendar Server User LDIF

The following sample LDIF file is for a user that has access to Calendar Server. The user was created in Delegated Administrator. Delegated Administrator does not distinguish between Calendar Server 6 and Calendar Server 7 users. It includes object classes from both Calendar Server 6 (`icsCalendarUser`) and Calendar Server 7 (`davEntity`).

```
dn: uid=johndoe,ou=People,o=example.com,o=isp
uid: johndoe
icstimezone: America/Denver
inetuserstatus: active
mail: john.doe@example.com
sn: Doe
surname: Doe
icscalendar: johndoe@example.com
userPassword: {SSHA}LyepThPJ2Y1Bi95fv8hP4WTZVSvdYV8kKUF9Ug==
icsstatus: active
objectclass: daventity
objectclass: iplanetpreferences
objectclass: person
objectclass: inetadmin
objectclass: ipuser
objectclass: icscalendaruser
objectclass: sunucpreferences
objectclass: inetorgperson
objectclass: organizationalperson
objectclass: inetuser
objectclass: top
givenname: John
icsFirstDay: 2
cn: John Doe
```

Chapter 15. Calendar Server Clients

Oracle Communications Calendar Server Clients



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This information describes some specific Calendar Server client behavior.

Topics:

- Connector for Microsoft Outlook and Event Time Modifications

Connector for Microsoft Outlook and Event Time Modifications

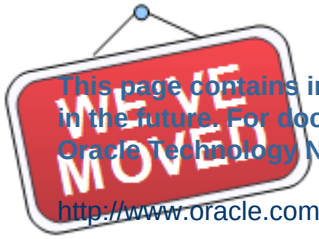
If you use Connector for Microsoft Outlook to create or modify the time of an event, and later make a change to the event time by using Convergence, the event "jumps" to a new time. In some cases, the event appears to "vanish" but in reality it "jumps" to the following day. Currently, there is no workaround.

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.

Chapter 16. Calendar Server Common Topics

Oracle Communications Calendar Server Topics



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This information describes some specific Calendar Server behavior.

Topics:

- How Free/busy Is Calculated for All Day Events
- Configuring Calendar Server Proxy Authentication

How Free/busy Is Calculated for All Day Events

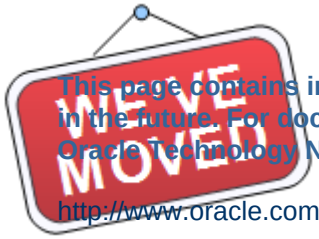
All events are considered to calculate busy time unless explicitly marked as transparent. The time blocked corresponds to the event timings in the event's time zone. For floating events (events with no associated time zone) and all day events, the busy time is calculated in the calendar's time zone. For example if there is a floating event from 12 pm to 1 pm, for lunch, in a calendar whose time zone is America/Los_Angeles, for free/busy calculation purposes, the event would be considered as a 12 pm-1 pm event in the America/Los_Angeles timezone.

Configuring Calendar Server 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 a CalDAV client to do so, when providing the user name and credentials for HTTP basic authentication, give the user name as "admin;user" instead of just user. For the password, use the administrator password. For WCAP clients, the login API defines a proxyauth parameter. See login.wcap for more information.

Chapter 17. Calendar Server Database Upgrade

Calendar Server Database Upgrade



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This page describes Calendar Server database upgrades.

Topics:

About Calendar Server Database Upgrades

Each Calendar Server patch or release might require an upgrade to the database schema or structure. For example, a database table might be altered, or data in the database might be changed. This type of upgrade differs from a database version upgrade, such as requiring an upgrade from MySQL Server 5.1 to 5.5. If a Calendar Server database upgrade is required, allow additional time for the overall Calendar Server upgrade, as the database upgrade can take a while, especially if the database is large.

Database Schema/Structure Version

Each database schema/structure has a Calendar Server back-end version number recorded in the database. Each Calendar Server version only runs with a single database schema version.

Calendar Server Start Up and Automatic Database Upgrade

When Calendar Server starts up in the GlassFish Server container, it connects to each back-end database for which it is configured, one at a time, and attempts to open the database for normal operation.

During this initial opening of each database, Calendar Server checks the database schema version. Calendar Server has the option to either automatically upgrade the database on the spot, or issue a

logging message in the `errors.0` log file explaining the back end cannot be opened and needs to be upgraded. In this latter case, refer to the Calendar Server documentation for more information on what steps you need to perform.

> **Caution**
> Never interrupt or stop either Calendar Server or GlassFish Server while the database is being upgraded. If you do, you need to restore the database from backup.

Calendar Server logs messages in the `errors.0` log file when the database was opened but not that it was being upgraded. If the database has a large amount of data, the upgrade can take a long time and appear to resemble a database hang. See Database Schema Upgrade Patches and Releases for more information on Calendar Server versions that require a database upgrade.

To monitor the database during an upgrade process on MySQL Server, use the following command:

```
mysql> SHOW FULL PROCESSLIST;
```

This command displays database upgrade aspects such as an `ALTER TABLE` command.

Database Schema Upgrade Patches and Releases

You cannot reverse a database schema and structure upgrade once it has been performed. This is why, when performing a Calendar Server upgrade, you must make a backup before starting the upgrade process. Once a database is upgraded, it no longer works with a previous Calendar Server version. Use the following table to understand the Calendar Server database changes by release.

Database Schema Version Changes for Calendar Server Beginning with Release 7 Update 2

Release	Database Change	Comments
7.0.4.15.0	No database changes.	No comments

7.0.4.14.0	Automatic Upgrade to Version 8.
	• (MySQL Server only) The charset and collation of most tables and fields is changed to utf8mb4.
	  • 8 `ALTER TABLE` commands are run.
	• Column added to `CalProp`.
	  • 1 `ALTER TABLE` command is run.
		Fairly intensive for MySQL Server.

7.0.3.8.0	Automatic Upgrade to Version 7. • Full table scan fixes missing values in `Resources.resourcetype = 'CAL_RESOURCE';`.	No comments
7.0.2.4.0	Automatic Upgrade to Version 6. • Rename table `Resource` to `Resources`. • `ALTER TABLE` command run on `CalProp` to change `supported_calendar_component_set` to `supported_cal_component_set`. • Change table `ICalProp` to `InnoDB` (by dropping it and allowing it to repopulate).	No comments

Database Performance During Upgrade

How a database performs during an upgrade depends on data size and hardware. Some database operations, such as copying the entire table, are intensive operations and can take a long time to complete. When your database has 4 million rows or 40 gigabytes of data, you should study the database performance to better understand upgrade times.

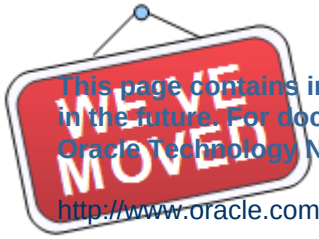
MySQL Server and OracleDB have relevant buffer cache settings that can increase performance. Additionally, if the database can fit in the cache completely with extra space, then operations might run faster.

If the database and temporary table operations exceed the datbase cache, then the disk IO throughput becomes very relevant. Disk IO is also significant when there is plenty of cache because the cache must be loaded at times.

MySQL Server also has memory and cache parameters such as `innodb_buffer_pool_size`, while OracleDB has parameters such as `sga_target`. Refer to MySQL Server and OracleDB documentation for more information on performance tuning.

Chapter 18. Calendar Server Horizontal Scalability and Multiple Hosts Examples

Oracle Communications Calendar Server Horizontal Scalability and Multiple Hosts Examples



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

These examples show how to scale Oracle Communications Calendar Server (Calendar Server 7) horizontally by deploying multiple front- and back-end MySQL Server hosts. As of version 7 Update 2, Calendar Server also supports Oracle Database as a calendar back end.

> **Note**
> To add a new Calendar Server back-end host, see Configuring Multiple Calendar Server Back-end Hosts.

Topics:

- Scenario 1: A Single CalDAV Server and Two MySQL Back-End Servers
- Scenario 2: Four CalDAV Servers and Three MySQL Back-End Servers

Scenario 1: A Single CalDAV Server and Two MySQL Back-End Servers

This scenario shows a deployment consisting of one front-end server (`davserver1.example.com`) and two back-end servers (`mysql1.example.com` and `mysql2.example.com`). The `mysql1.example.com` server has one database, `caldav1`, and the `mysql2.example.com` server has one database, `caldav`.

The following table shows the deployment data for this scenario.

Scenario 1 Deployment Data

davstore ID	MySQL Server	Database Name	JDBC URL
backend1	mysql1.example.com	caldav1	jdbc:mysql://mysql1.example.com:3306/caldav1
defaultbackend	mysql2.example.com	caldav	jdbc:mysql://mysql2.example.com:3306/caldav

Note
After you install the Calendar Server software then run the `init-config` script, the `defaultbackend`-related settings are auto-created and set up.

The front-end host is configured with the following configuration parameters:

```
store.dav.backend1.backendid=backend1
store.dav.backend1.jndiname=jdbc/backend1
store.dav.backend.backendid=defaultbackend
store.dav.backend.jndiname=java:comp/env/jdbc/defaultbackend
```

On the GlassFish Server that hosts Calendar Server, a JDBC connection pool is created for each database, as the following table shows.

JDBC Connection Pools Per Calendar Server

Pool Name	Value
caldav1Pool	jdbc:mysql://mysql1.example.com:3306/caldav1
caldavPool	jdbc:mysql://mysql2.example.com:3306/caldav

A JDBC resource pointing to each connection pool is created, as the following table shows.

To create a connection pool, see To Use the GlassFish Admin Console to Create a Connection Pool for Non-Default Back Ends or To Use the Command Line to Create a Connection Pool for Non-Default Back Ends.

JDBC Resource Per Connection Pool

JNDI Name	Pool Name
jdbc/backend1	caldav1Pool
jdbc/defaultbackend	caldavPool

You need to restart GlassFish Server after the pools are created.

Scenario 2: Four CalDAV Servers and Three MySQL Back-End Servers

This scenario shows a deployment that consists of four front-end CalDAV servers (`davserver1.example.com`, `davserver2.example.com`, `davserver3.example.com`, and `davserver4.example.com`), and three back-end MySQL servers (`mysql1.example.com`, `mysql2.example.com`, and `mysql3.example.com`). The `mysql1.example.com` server has one database (`caldav1`), the `mysql2.example.com` has one database (`caldav`), and the `mysql3.example.com` server has one database (`caldav2`).

The following table shows the deployment data for this scenario.

Scenario 2 Deployment Data

davstore ID	MySQL Server	Database Name	JDBC URL
backend1	mysql1.example.com	caldav1	jdbc:mysql://mysql1.example.com:3306/caldav1
defaultbackend	mysql2.example.com	caldav	jdbc:mysql://mysql2.example.com:3306/caldav
backend2	mysql3.example.com	caldav2	jdbc:mysql://mysql3.example.com:3306/caldav2

Note
After you install the Calendar Server software then run the `init-config` script, the `defaultbackend`-related settings are auto-created and set up.

Each front-end host is configured with the following configuration parameters:

```
store.dav.backend1.backendid=backend1
store.dav.backend1.jndiname=jdbc/backend1
store.dav.backend.backendid=defaultbackend
store.dav.backend.jndiname=java:comp/env/jdbc/defaultbackend
store.dav.backend2.backendid=backend2
store.dav.backend2.jndiname=jdbc/backend2
```

On each GlassFish Server that hosts Calendar Server, a JDBC connection pool is created for each database, as the following table shows.

To create a connection pool, see To Use the GlassFish Admin Console to Create a Connection Pool for Non-Default Back Ends or To Use the Command Line to Create a Connection Pool for Non-Default Back Ends.

JDBC Connection Pools Per Calendar Server

Pool Name	Value
caldav1Pool	jdbc:mysql://mysql1.example.com:3306/caldav1
caldavPool	jdbc:mysql://mysql2.example.com:3306/caldav
caldav2Pool	jdbc:mysql://mysql3.example.com:3306/caldav2

A JDBC resource pointing to each connection pool is created, as the following table shows.

JDBC Resource Per Connection Pool

JNDI Name	Pool Name
jdbc/backend1	caldav1Pool
jdbc/defaultbackend	caldavPool
jdbc/backend2	caldav2Pool

You need to restart GlassFish Server after the pools are created.

Chapter 19. Configuring CalDAV Clients for Calendar Server 7

Configuring CalDAV Clients for Calendar Server 7

This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This information describes how to configure Apple and Lightning clients to communicate with Calendar Server.

> **Note**
> Starting with version 7.3 patch 13, Connector for Microsoft Outlook can be used as a client with Calendar Server 7 Update 2. See the Connector for Microsoft Outlook User's Guide for more information.

Topics:

Prerequisites

1. Users need to be provisioned in Directory Server. For more information, see Automatically Provisioning Calendar Server 7 Users, Provisioning for Calendar Server Users, and Calendar Server and Directory Server Integration.
2. Obtain the following information on your Calendar Server 7 deployment:
   - GlassFish Enterprise 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:

To Configure Apple iCal for Calendar Server

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. Type your user name, password, and server name, then click Create.
6. The client indicates "Verifying CalDAV account", then "Account verified."
7. You can now use the Calendar application.

To Configure Apple iPhone for Calendar Server

1. Navigate to the Mail, Contacts, and Calendar settings menu.
2. Select Add Account.
3. Select Other.
4. Select Add CalDAV Account.
5. Type 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.

To Configure 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.

1. Download Lightning 1.0 beta2 to your client machine.
   See 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. Type the URL of the calendar, for example:

```
https://example.com/dav/home/jsmith@example.com/calendar/
```

k. Type your name, choose a color scheme, choose to set alarms or not, and select your email address.
l. Type 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.

To Configure 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.

1. Download Lightning 1.0 beta 1 to your client machine.
See 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. Type 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*`/`*base-uri*`/dav/home/`*email-address*`/calendar/`

    k. Type your name, choose a color scheme, choose to set alarms or not, and select your email address.
    l. Type 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.

To Configure 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.

1. Download Lightning 0.9 to your client machine.
   See 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. Type 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*`/`*base-uri*`/dav/home/`*email-address*`/calendar/`

   k. Type your name, choose a color scheme, choose to set alarms or not, and select your email address.
   l. Type 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.

To Access 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)

To Configure 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 Best Practices for Calendar Server.

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. Type 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. Type 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. Type 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. Type your user name and password.
6. In Server Address, type 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 type the principal URI, for example:

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

To Configure 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, 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 iPhone Configuration Utility to install and manage installation profiles. Enterprises might find this utility helpful to manage their end user accounts. For more information, see http://support.apple.com/kb/dl851.

Exporting and Importing Calendars in Thunderbird Lightning

This section contains the following tasks:

- To Export a Calendar
- To Import 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).

To Import a Calendar

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

CalDAV Client Issues

See Troubleshooting CalDAV Clients.

Where to Go From Here
For information on some specific client behaviors, see Calendar Server Clients.

Configuring Proxy Authentication

See Configuring Calendar Server Proxy Authentication.

More Information

See Calendar Server Clients.

Chapter 20. Configuring Multiple Calendar Server Back-end Hosts

Configuring Multiple Calendar Server Back-end Hosts

This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This information describes how to configure multiple Calendar Server back-end hosts. For conceptual information, see Calendar Server Front-End and Back-End Servers. For information on performing an initial installation of Calendar Server, including back-end hosts, see Installation Scenario - Calendar Server.

> Starting with **Calendar Server 7 Update 2**, you can use either MySQL Server or Oracle Database as the calendar server back end (calendar store).

Topics:

Installing and Configuring Calendar Server Back-End Hosts Overview

A standard Calendar Server installation consists of a default back-end database that contains user data, and an iSchedule back-end database for iScheduling requests. Over time, you might want to add additional back-end user data bases to your initial deployment.

In the case of multiple Calendar Server front ends, configure each to use the same initial default database back end and ischedule back end. Then, you add additional back ends to each front end. Only one iSchedule back end is needed for all front ends to share.

Each back end database, including the iSchedule database, must have its own document store. In the case of multiple front ends, all document stores must be available to all front ends. You cannot make all document stores local to a front end in a multiple front-end deployment.

High-Level Steps

The high-level steps to configure multiple back-end Calendar Server hosts include:

1. Installing, configuring, and preparing (by running either the `config-mysql` or `config-oracle` script) the new database
2. Gathering the database host names, ports, and other database names
3. Installing all front-end servers, if not already done
4. Configuring all front-end servers by using the `init-config` program, if not already done
   Running the `init-config` program creates the `config-backend` script for use in the next step. Information on any back-end server can be used for this step. If you have already run the `init-config` command, and you want to rename the default back-end server, see To Rename the Default Calendar Server Back End.
5. On each front-end server, running the `config-backend` script
6. On each front-end server, enabling connection pool validation
7. On each front-end server, restarting GlassFish Server

Skip Steps 2 and 3 if you are adding new back-end servers to an existing front-end installation.

1. Install the database software on each back-end host.
   Choose one of the following:
   - To Install a MySQL Database for Calendar Server
   - To Install and Create an Oracle Database Instance for Calendar Server
2. Decide if you need to create additional Oracle Database or MySQL Server back-end databases. If MySQL, continue with this step. If Oracle, skip to Step 3.

   Note
   If the Calendar Server software is not installed on the back-end host, copy the `config-mysql`, `config-oracle`, and `Util.pm` scripts from an installed Calendar Server host and adjust the following path to those scripts accordingly.

   - If this is first database on the host, set up the instance, and create the user and database by running the following command.

     ```
     <cal-svr-base>/tools/unsupported/bin/config-mysql -s -u -c
     ```

   - If there is already a database on the host, just create the calendar database by running the following command.

     ```
     <cal-svr-base>/tools/unsupported/bin/config-mysql -c
     ```

     Skip to Step 4.

3. To create the Oracle database user and schema, run the following command.

   ```
   <cal-svr-base>/tools/unsupported/bin/config-oracle -c
   ```

4. On each front-end host, run the `config-backend` script.
   This script creates a JDBC connection pool and a JDBC resource on the GlassFish Server, and a `davserver` attributed back-end configuration.

   ```
   <cal-svr-base>/sbin/config-backend
   ```

- If current deployment is using MySQL, you are prompted for the following information:

```
Remote database server host name
Remote database server port
Calendar db name on remote server
Calendar db user name
Calendar db user password
Verifying the database input...
Database input is verified
Backend identifier for the remote db
Document store directory (leave blank if store is remote)
Document store host (leave blank if store is local)
Document store port (leave blank if store local)
Application Server admin user password
```

  Make sure the value for "Calendar db name on remote server" is the one that you used for the `config-mysql -c` command.

- If current deployment is using Oracle Database, you are prompted for the following information:

```
Remote database server host name
Remote database server port
Oracle database service name on remote server
Calendar db user name
Calendar db user password
Verifying the database input...
Database input is verified
Backend identifier for the remote db
Document store directory (leave blank if store is remote)
Document store host (leave blank if store is local)
Document store port (leave blank if store local)
Application Server admin user password
```

  Make sure the value for "Calendar db user name" is the one that you used for the `config-oracle -c` command.

5. Type **Y** when prompted to perform the tasks for creating the JDBC connection pool and resource, and `davserver` back-end identifier.
   The system responds that the database back-end configuration is configured successfully.
6. On each front-end host, enable Connection Validation for the connection pools (both CalDav back-end and iSchedule pools) so that Calendar Server automatically reconnects to the back-end database if it goes down:
   a. In the GlassFish Server Admin Console, choose Resources, then JDBC, then Connection Pools.
   b. Select the pool.
   c. Check the Required box for Connection Validation.
   d. Choose table for the Validation Method.
   e. Type DUAL for Table Name.
   f. Click Save.
   g. Click the Advanced tab.
   h. Type 60 for Validate Atmost Once.
   i. Click Save.
   This configuration then issues the command "select count(*) from DUAL;" on every connection, at most one time every 60 seconds. (If the connection is not being used, it is

not checked.)

> **Note**
> Use these settings as a starting point and adjust where necessary. For example, if validation is not important, you can turn it off. Additionally, you might want to adjust the "Validate At Most Once" time duration or validate each time a connection is requested (by setting the value to 0). HA deployments might also use different values.

7. Restart GlassFish Server, for example:

```
# /opt/SUNWappserver/bin/asadmin stop-domain domain1
Domain domain1 stopped.
# /opt/SUNWappserver/bin/asadmin start-domain domain1
Starting Domain domain1, please wait.
Log redirected to
/opt/SUNWappserver/domains/domain1/logs/server.log.
Please enter the admin user name>admin
Please enter the admin password>adminpass
Please enter the master password>adminpass
```

8. Provision accounts for a multiple back-end deployment.
   See Provisioning Calendar Accounts in a Multiple Back-end Deployment.

To Rename the Default Calendar Server Back End

The Calendar Server `init-config` script creates the JDBC connection pool and resource, and adds the information to the `davserver.properties` file, for the one back-end host specified during the front-end configuration. The JDBC resource used is **`defaultbackend`**.

If you need to change this JDBC resource, to match other naming conventions, follow these steps.

1. On each front-end GlassFish Server, create a JDBC resource associated with the `caldavPool` connection Pool.
   For example, you might use `db1` as the resource name.
2. Save this change then restart GlassFish Server.

```
# /opt/SUNWappserver/bin/asadmin stop-domain domain1
Domain domain1 stopped.
# /opt/SUNWappserver/bin/asadmin start-domain domain1
Starting Domain domain1, please wait.
Log redirected to
/opt/SUNWappserver/domains/domain1/logs/server.log.
Please enter the admin user name>admin
Please enter the admin password>adminpass
Please enter the master password>adminpass
```

3. Add the following two lines to each `davserver.properties` file.

```
store.dav.db1.backendid=<JDBC resource>
store.dav.db1.jndiname=jdbc/<JDBC resource>
```

   For example, if your resource name is `db1`, then you would add:

```
store.dav.db1.backendid=db1
store.dav.db1.jndiname=jdbc/db1
```

The new resource *name* can be used in `davstore` attribute values.

> **Note**
> Once your Calendar Server deployment is up and running, do not change the user back-end ID as defined by the `davStore` attribute.

Provisioning Calendar Accounts in a Multiple Back-end Deployment

For calendar accounts to know which back-end host they should connect to, you need to provision accounts with the `davstore` attribute. The `davStore` attribute indicates the back-end host that stores a user's data if the deployment is configured for multiple back ends. For more information, see davStore Attribue and Calendar Server and Directory Server Integration.

Examples: Creating Connection Pools and JDBC Resources for Non-Default Calendar Server Back Ends

> The following examples apply only to **Calendar Server 7** and **Calendar Server 7 Update 1**. Starting with **Calendar Server 7 Update 2**, this process is automated by the `config-backend` script.

This section contains the following topics:

- To Use the GlassFish Server Admin Console to Create a Connection Pool for Non-Default Back Ends
- To Use the Command Line to Create a Connection Pool for Non-Default Back Ends

To Use the GlassFish Server Admin Console to Create a Connection Pool for Non-Default Back Ends

This example uses the GlassFish Server Admin Console to create a connection pool `caldav1Pool` and JDBC resource `jbc/backend1`, and to enable Connection Validation.

1. Select New and type the following information.
   - Name: `caldav1Pool`
   - Resource Type: `javax.sql.DataSource`
   - Database Vendor: `MySQL`
2. Click Next.
3. Set the following properties and be sure that the URL property is either not set or set correctly. You can also delete all the default properties and keep just the following six properties.
   - databaseName: `caldav1`
   - portNumber: `3306`
   - networkProtocol: `jdbc`
   - serverName: `localhost` (or your MySQL server host name)
   - user: `mysql`
   - password: `mysql`
4. Save.
5. Select the pool and use the Ping button to test the pool.

If ping was not successful, you can delete all the default properties and keep just the six properties previously mentioned. Then retry the ping.

6. Create JDBC resource for the connection pool.
   Go to Resources -> JDBC -> JDBC Resources.
7. Select New and type the following information:
   - JDNI Name: `jdbc/backend1`
   - Pool Name: `caldav1Pool`
   - Status: check Enabled
8. Enable Connection Validation for the connection pool:
   a. In the GlassFish Server Admin Console, choose Resources, then JDBC, then Connection Pools.
   b. Select `caldav1Pool`.
   c. Check the Required box for Connection Validation.
   d. Choose table for the Validation Method.
   e. Type DUAL for Table Name.
   f. Click Save.
   g. Click the Advanced tab.
   h. Type 60 for Validate Atmost Once.
   i. Click Save.

To Use the Command Line to Create a Connection Pool for Non-Default Back Ends

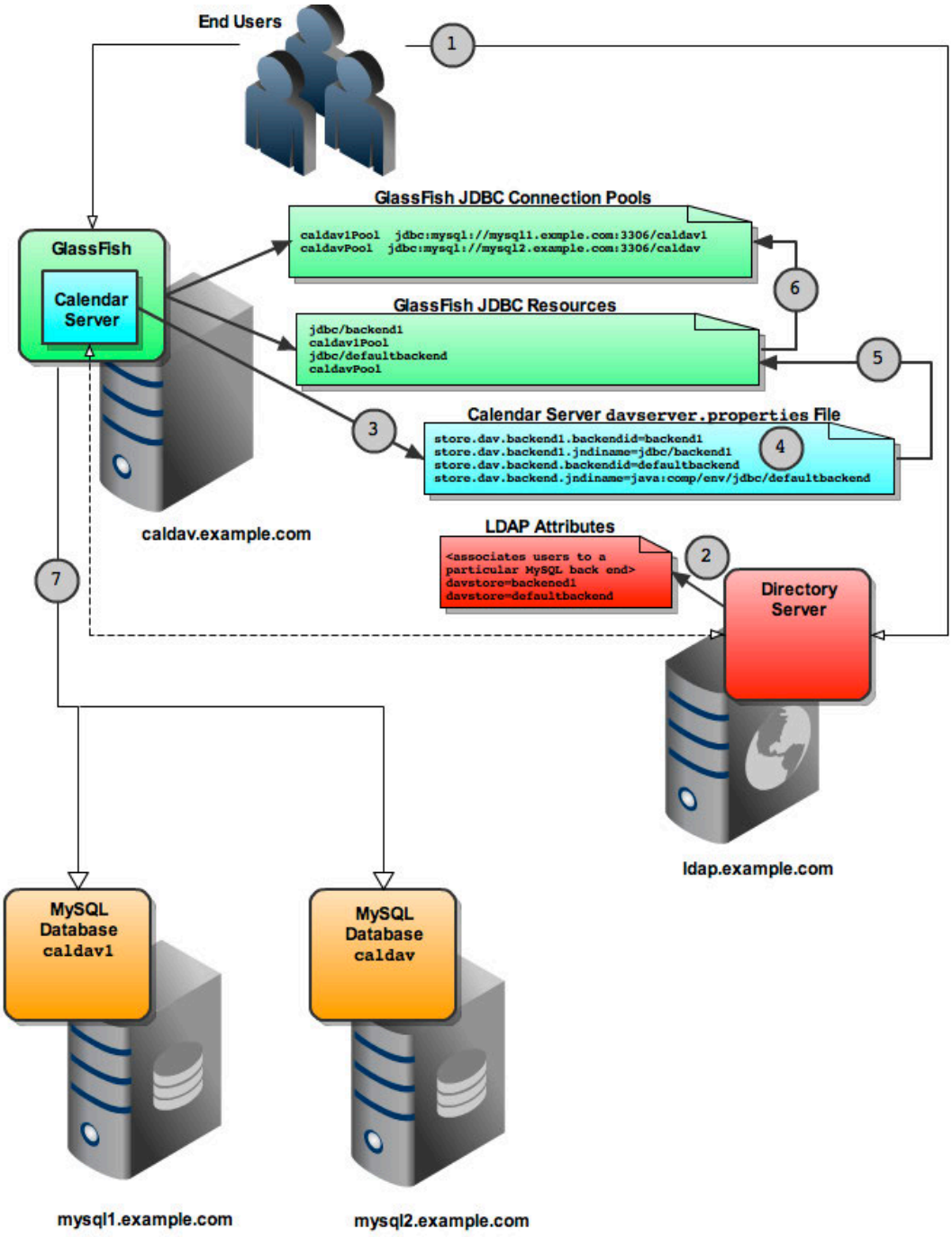
This example uses the command-line interface to create a connection pool `caldav1Pool` and JDBC resource `jbc/backend1`.

```
% /opt/SUNWappserver/bin/asadmin create-jdbc-connection-pool --user
admin --datasourceclassname
com.mysql.jdbc.jdbc2.optional.MysqlDataSource --restype
javax.sql.DataSource --property
"DatabaseName=caldav1:serverName=mysqlhost:user=caldav:password=mysqlpass:p
caldav1Pool
% /opt/SUNWappserver/bin/asadmin create-jdbc-resource --user admin
--connectionpoolid caldav1Pool jdbc/backend1
```

Multiple Back-ends Flow of Information

The following figure shows a Calendar Server 7 multiple back-end configuration.

Calendar Server 7 Multiple Back-End Configuration



In this figure, the information flow is as follows:

1. Users log in to Calendar Server.
2. LDAP attribute `davstore` is read, indicating which back end the user is associated with.
3. The `davstore` attribute is mapped to one of the `store.dav.`*xx*`.backendid` values in the `davserver.properties` file.
4. The corresponding JNDI name (`store.dav.`*xx*`.jndiname`) is obtained.
5. This points to a JDBC Resource associated with the MySQL back-end database.
6. The JDBC Resource points one of the JDBC Connection pools.
7. The user gets a Calendar Server database connection.

For example, in the preceding figure, assume that `user1` has a `davstore` attribute set to `backend1`. This would be mapped to the following:

```
store.dav.backend1.backendid=backend1
store.dav.backend1.jndiname=jdbc/backend1
```

The JNDI name would be obtained from:

```
store.dav.backend1.jndiname=jdbc/backend1
```

This, in turn, resolves to the following JDBC Resource:

```
Name=jdbc/backend1
ConnectionPool=caldav1Pool
```

Next, the following JDBC Connection pool is obtained:

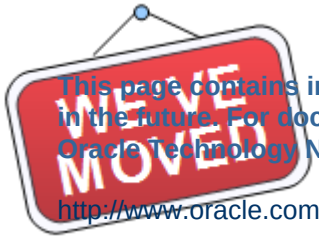
```
ConnectionPool=caldav1Pool

Name=caldav1Pool
DB=jdbc:mysql://mysql1.example.com:3306/caldav1
```

Finally, `user1` is given a connection to the MySQL back-end host with the `caldav1` database.

Chapter 21. Migrating From Sun Java System Calendar Server 6 to Calendar Server 7

Migrating From Sun Java System Calendar Server 6 to Oracle Communications Calendar Server



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This functionality is available starting with **Oracle Communications Calendar Server 7 Update 1**.

Introduction to Migrating

Because the database formats differ between Calendar Server 6 and Oracle Communications Calendar Server (formerly known as Sun Java System Calendar Server 7), you need to migrate your Calendar Server 6 user data. That is, you cannot directly upgrade from Calendar Server 6 to Oracle Communications Calendar Server. Oracle Communications Calendar Server provides the `davadmin migration` utility to migrate the data.

The `davadmin migration` utility migrates all relevant Calendar Server 6 data and properties, including calendar data, invitations, ACLs, subscriptions, and so on. There is no migration or updating of LDAP data stored in the Directory Server. Instead, Calendar Server 6 calendar-related properties, such as notifications, previously stored in LDAP, are now stored in the Oracle Communications Calendar Server back-end database, which can be either MySQL Server or Oracle Database (as of Calendar Server 7 Update 2). For more information on how Calendar Server uses Directory Server, see Calendar Server and Directory Server Integration.

Note
Domain level ACLs are set in LDAP but using a different set of attributes and formats as described in Managing Domain Access Controls. The migration does not migrate domain level ACLs.

The migration does not check such items as event invitees, or event owners or delegated owners. The migration transfers whatever information is found in the LDAP directory to the Oracle Communications Calendar Server database. Additionally, if you migrate the same user multiple times, the user does not end up with duplicate events in the migrated calendar.

The calendar data migration process assumes the following conditions:

- You can allow a reasonable amount of downtime to complete the migration.
- You can perform a trial run to check results before doing the actual migration.
- You can examine and fix events and todos that failed to migrate, by manually updating `ics` files and importing them.
  In general, when fixing and importing make sure the values that used to be "calid" are changed to their "mail" equivalent in all `ics` components.
- Any updates to already scheduled events are not implicit after the migration.
- If your deployment consists of multiple domains, ACIs are configured to enable the Calendar Server administrator ID, set by the `service.siteadmin.userid` configuration parameter (default is `calmaster`), for search and read access to non-default domains.

Note
You can also have a coexistent deployment of Calendar Server 6 and Calendar Server 7 hosts. For more information, see Creating a Calendar Server 6 and Calendar Server 7 Coexistent Deployment.

How the Migration Works

The migration utility performs the following sequence of events:

1. Communicates with the Calendar Server 7 host by using the JMX protocol and invoking the `AdminMigration` process.
2. Creates a log file for that migration task, which is returned to the `davadmin` client.
3. The `AdminMigration` process invokes the Migration Service, which gets the list of users to be migrated from LDAP or passed in from the migration client, and builds a list of users.
4. Starts a migration thread for each user in the list, until it reaches the `serverlimits.maxmigrationthreads` setting. Each migration thread then performs the migration. When done, each thread is reassigned a new user.
5. Each migration thread uses the Migration HTTP client and authenticates to the Calendar Server 6 host by using the provided `administrator` settings and gets a session ID. This session ID is reused for all further WCAP requests to the Calendar server 6 host, for this particular user.
   a. First, the user's list of calendars and other preferences are fetched by using the `get_userprefs` command.
   b. For each of the user's calendars, the calendar properties are fetched using the (`get_calprops`) command.
   c. Then using the `fetch_by_range` command events/tasks uids and type, for the specified period are fetched.
      A map of the uids is made and individual fetches (`fetch_event_by_id` or `fetch_todos_by_id`) are done to get actual data. Fetch is done with the recurring=1 flag set, to get the master and exceptions as one record. This data is stored in the user's

Calendar Server 7 back-end, as specified by the LDAP `davStore` attribute. The fetched data is massaged to fit the format for the Calendar Server 7 host before being stored. Different mapping helper functions perform the data massaging.

6. The user's calendar is autocreated before adding any events or tasks. The fetched calendar properties are mapped and used to update the properties of the newly created calendar.
7. The migration does not overwrite or duplicate any calendar data. When the migration service tries to write and finds an already existing calendar entry, it just ignores it and continues.

> **Note**
> If a calendar being migrated from Calendar Server 6 already exists on Calendar Server 7 because it was previously migrated or was explicitly created in Calendar Server 7, the migration resets the calendar's properties to that of the values of the original Calendar Server 6 calendar.

8. Any failed event or task is logged, and the migration task continues.
9. When migration is done, the calendar migration status is set in the migration log file, with more details on what was migrated.
10. The migration steps are repeated for every calendar of the user.
11. When done with one user, the migration thread removes the user from the user list map and picks up the next unmarked user in the list, if any.

Notes:

- You migrate all accounts (users and resources) either by providing the `migration` command with the accounts' mail addresses, or by specifying the LDAP base and filter that finds the information on the users and resources in LDAP. Resource calendars are not migrated by just migrating the resource owner but by explicitly migrating the resource account itself.
  Resource Owner
- In Calendar Server 7 Update 3, the migration sets the owner of a resource to the value of the primary owner. If the primary owner is not set, the migration uses the LDAP owner field. If there is no LDAP owner, the migration does not set an owner for the resource. (Bug 13942095)
- An account cannot be partially migrated. The process migrates all calendars under that account, as well as subscription lists, calendar settings such as access controls, owner list, notification settings, freebusy settings, and so on.
- Because subscription lists are migrated without checking if the calendars pointed to are migrated as well, some broken subscription links could result until all calendars have been migrated.
- Calendar Server 7 Update 3 introduces a new configuration parameter, `davcore.autocreate.rescalcomponents`. It enables migrated resource calendars to allow the `VEVENT` and `VTODO` components that were permitted in Calendar Server 6. (The default value for this parameter is `VEVENT`.) To ensure that Calendar Server 6 resource calendars correctly have all their `VEVENT` and `VTODO` information migrated to Calendar Server 7, set this parameter to VEVENT VTODO before doing the migration, if your deployment has resource calendars that contain more than just events. For more information, see Calendar Server 7 Configuration Parameters.

High-Level Migration Steps for Administrators

Migrating from Calendar Server 6 to Oracle Communications Calendar Server involves the following high-level steps:

1. Applying the most current Calendar Server 6 patch to fix migration-related issues
2. Making a list of users to be migrated
3. Informing users of migration window and down-time
4. In a multiple domain deployment for LDAP Schema 1 environments, if not already done, configuring LDAP ACIs so that the Calendar Server administrator ID, set by the

`service.siteadmin.userid` configuration parameter (default is `calmaster`), has search and read access to non-default domains
For example:
   a. Ensure that the `icsDomainNames:` *non-default domain* attribute is added to the default domain in the dc tree.
   b. Ensure that the `icsExtendedDomainPrefs: domainaccess=` *cal_svr_admin_id@defaultdomain*`^a^r^g` attribute is added to the non-default domain in the dc tree.
5. Setting the `service.http.migrationcompatible` parameter to **yes** (and restarting the Calendar Server)
6. Readying the Calendar Server 7 deployment for migration by setting the required configuration options
7. Running the `populate-davuniqueid` script to update users, groups, and resources in LDAP with the unique identifier attribute (`populate-davuniqueid` is available starting in Calendar Server 7 Update 3)
8. Updating the LDAP attribute `davStore` for each user being migrated, to indicate which back-end Calendar Server 7 host to use

> **Note**
> Not setting this attribute means the back-end host is local to the Calendar Server 7 installation.

9. Performing a backup of the user calendars that are being migrated
10. Creating a file with list of mail addresses of users to be migrated or determining the LDAP filter to use
11. Setting the Calendar server 6 host to "Read Only" mode as described in To Put a Database in Read-only Mode
12. Migrating users by running the `davadmin migration` script and with the user list file created or the LDAP base and filter
    For more information, see `davadmin migration`. The default action for `davadmin migration` is `migrate` and so can be omitted.
13. Verifying migration status
14. For successfully migrated users, performing the following steps:
    a. Deleting Calendar Server 6 calendars by using `cscal delete -o` *uid*
    b. Setting the LDAP attribute `icsAllowedServiceAccess` to `-http:all`
15. Fixing and importing any failure log files
16. Notifying migrated users and informing them how to access the Calendar Server 7 deployment

Migrating From an SSL-enabled Calendar Server 6 Deployment

For a Calendar Server 7 host to communicate with a Calendar Server 6 host over SSL, the Calendar Server 7 host needs to have the Calender Server 6 host's certificate available. You do this by exporting the certificate from the Calendar Server 6 host and importing it into the Java runtime environment that is used by the Calendar Server 7 host.

1. To export a certificate from a Calendar Server 6 host, run the following `certutil` command:

```
certutil -L -a -n <alias_name> -d <CS6_config_directory> >
<CS6_certificate_file.rfc>
```

   where:

alias_name	Specifies the alias name of the certificate in the Calendar Server 6 Application Server NSS database
CS6_config_directory	Specifies the path to the Calendar Server 6 host's `config` directory
CS6_certificate_file.rfc	Specifies the path to the output file for the certificate

2. Once you have the `.rfc` file, transfer it to the Calendar Server 7 host and import it into the Java runtime environment that is used by Calendar Server 7.
   The Java runtime file that holds the certificates is called **`cacerts`**. It should have a path similar to **`/usr/jdk/latest/jre/lib/security/cacerts`**. The import command used to update the cacerts file is:
   keytool -import -alias **`alias_name`** -keystore /usr/jdk/latest/jre/lib/security/cacerts -file **`CS6_certificate_file.rfc`**
3. You must restart GlassFish Server for these changes to take affect.

Migration Logging

The `davadmin migration` command returns a `log_tag` string, which is is the path to the `master_log` file on the server host that contains the current state of that migration. The current state includes a list of migrated users and migration status. This `master_log` file is synchronously updated as the migration progresses.

The `davadmin migration status -G log_tag` command returns the contents of this `master_log` file and can be used to view the current state of the migration. You can also view the `master_log` file by using UNIX commands such as `cat`, `tail`, `vi`, and so on. The migration is finished when the last line in the `master_log` file is "Migration complete."

Here is an example `log_tag` string:

```
/var/opt/sun/comms/davserver/logs/davserver_migration/2010-06-29_153301/mas
```

To view this file while the migration is in progress, you could use the following command:

```
tail -f
/var/opt/sun/comms/davserver/logs/davserver_migration/2010-06-29_153301/mas
```

The root of the directory tree that holds the migration information is `davserver_migration` and defaults to the Calendar Server 7 `log` directory. To store the migration information tree in a different location, use the `-l` option. This option tells `davadmin` where the directory should be placed. The structure of the `davserver_migration` tree is the following:

Migration Logging Root Directory	Time Stamp Named Directory	User Level Directory	Calendar Level Directory	Calendar Detail Information
`davserver_migration`				
	Migration time stamp for each migration			
		`master_log`		
		Directory for each user being migrated		
			Default calendar name	
				`calendar_log`
				`.ics` files (if migration failed for any event or todo)
			Any secondary calendar name	
				`calendar_log`
			`trace_information`	
				Files containing results of commands used to obtain old calendar data from the Calendar Server 6 host. Use this information to diagnose migration problems.

Under the `davserver_migration` directory is a time stamp named `directory`. This directory is created when you start a migration operation and the time stamp reflects when you started that migration.

The time stamp named directory contains the `master_log` file and a directory for each user being migrated.

The user's directory contains a directory for each of that user's calendars and a `trace_information` directory if the `-c` option was used on the command line. The `trace_information` directory contains files that show the results of commands issued to the Calendar Server 6 host.

Each calendar directory contains the `calendar_log` file and `.ics` files. The `calendar_log` contains information about the migration of that particular calendar. The `.ics` files contain the iCalendar data for any event/todo that failed to migrate. If a migration failed, you might be able to fix the iCalendar data in these `.ics` files and import them into the new calendar.

Set the Calendar Server 7 log level to FINE during migration, for easy detection of issues.

Use the `-c` or `--capture` flag to further debug issues that might come up during migration. Migration logging can be quite verbose and takes up lots of disk space.

How the Migration Reformats the Calendar Server 6 Data

To ensure that the Oracle Communications Calendar Server data store does not accept non-iCal compliant data, the migration program manipulates or reformats migrated data as follows.

- Email alarms require a description and summary. If either or both are missing, a new value is constructed from the event or task's summary and added.
- Display alarms require a description. If missing, a new value is constructed from the event or task's summary and added.
- Organizer and attendee values are corrected to be valid `mailto:` URIs.
- If an event's `dtend` is not after its `dtstart`, or if they are not of the same type (date only or timed), the bogus `dtend` is discarded and a new one is constructed. If `dtstart` has a date only value, the new `dtend` is one day after the `dtstart`. If the `dtstart` is a timed value, the new `dtend` is one hour after the `dtstart`.
- If a todo has a `dtstart` that is after its due date, or if the value type of both properties do not match, the bogus `dtstart` is discarded. A new `dtstart` with the same value as due is added.
- If a recurring todo has no `dtstart`, a new one is constructed and added. If due exists, the new `dtstart` is the same as due. If no due exists, `dtstart` is set to the current time.
- If a priority field is missing, it is added for todos with attendees.
- If `dtstart` is missing or is bogus, the relative alarm trigger values of todos, relative to due, are set.
- Time strings in events and todos are converted from Zulu to values in their original timezones, and the relevant `vtimezone` information is included.
- A new alarm attendee property is added for explicitly setting SMS alarms that were set by using the `X-S1CS-SMS-EMAIL` property.
- Unending recurring series are truncated to a count specified by `X-NSCP-RECURRENCE-BOUND` in the Calendar Server 6 host, to retain the same recurrence behavior.
- If `dtend` is before or equal to `dtstart`, it is discarded and a new one is created. For all day events, the new dtend is equal to `dtstart` plus 1 day. For timed events, the new `dtend` is equal to `dtstart` plus one 1 hour.

Troubleshooting the Migration

Topics in this section:

Count Reported by Tools Differs

The count reported by the Calendar Server 7 migration utility, for example, "Migrated 341 events in: jsmith," is different from the count reported by some Calendar Server 6 tools, such as `csexport`

("number of events=1436"). This difference occurs because most of the Calendar Server 6 tools report number of events in expanded mode (where each instance of a recurring event counts as one), while the migration utility reports the number of events in master plus exception mode.

No Events Migrated When The Calendar Contains Events

If your migration completed successfully but did not migrate events or tasks from a calendar that does contain events or tasks, check to see if the `calmaster` ID has permission to access the calendar events and tasks for that user.

For example, if `user1` on Calendar Server 6.3 resembles the following:

```
# ./cscal -v list user1
user1@example.com: owner=user1@example.com status=enabled
...
number of events=27
number of tasks=0
number of deleted events=0
number of deleted tasks=0
number of deleted recurring events=0
number of deleted recurring tasks=0
```

but when migrated to Calendar Server 7 produces following:

```
# ./davadmin migration -u admin -a user1@example.com -X calmaster -L
cs6server.example.com:80

[user1@example.com] Creating calendar user1 with name: null
[user1@example.com] Migrated 0 events and 0 tasks in: user1
[user1@example.com] Subscribed to
[user1@example.com] Migration completed successfully.
```

you should perform the following steps:

1. On the Calendar Server 6.3 host, look in the `http.log` file log for entries similar to the following. Such entries indicate that the `calmaster` ID is denied access to `user1`'s data:

   ```
   [20/Nov/2011:15:54:11 -0800] cs6server cshttpd[14567]: Account
   Information: login [123.10.10.10] calmaster plaintext sid=DCGF1veS5U8
   [20/Nov/2011:15:54:11 -0800] cs6server cshttpd[14567]: General Error:
   ac: Fetchcomponents: User "calmaster" denied access on fetching from
   calendar "user1".
   [20/Nov/2011:15:54:11 -0800] cs6server cshttpd[14567]: General Error:
   calstore_fetchcomponents_by_range(): call to ac_fetchcomponents()
   returning err = 13.
   [20/Nov/2011:15:54:11 -0800] cs6server cshttpd[14567]: General Error:
   calstore_fetchcomponents_by_range(): db error (pError->iErr) = 28.
   ```

2. On the Calendar Server 6.3 host, check that the `ics.conf` file contains the following two parameters:

   ```
   service.siteadmin.userid
   service.virtualdomain.support
   ```

   a. For non-virtual domain data, the parameters should be set to the following values:

```
service.virtualdomain.support = "no"
service.siteadmin.userid="calmaster"
```

Here, `calmaster` is an example. Use your site's Calendar Server administrator ID.

b. For virtual domain data, the parameters should be set to the following values:

```
service.virtualdomain.support = "yes"
service.siteadmin.userid="calmaster@example.com"
```

Again, `calmaster@example.com` is an example. Use your site's Calendar Server administrator ID.

3. If the two Calendar Server 6.3 configuration parameter values are not consistent, then permission problems arise and the Calendar Server administrator ID is not allowed access to the calendars that are to be migrated.

To correct the problem:

1. Change the Calendar Server 6.3 configuration parameters as previously described.
2. Stop and restart calendar services on the Calendar Server 6.3 host.
3. Rerun the migration.
4. Check the migration output to verify that some number of events were migrated.
5. Check the `http.log` file to ensure that the "denied access" messages no longer occur.

Exception on creation of userpref Error

If your migration produces intermittent errors similar to the following:

```
2010-04-23_155050/master_log:[zw127108@gotmail.example.com ] Exception on
creation of userpref command for zw127108@gotmail.example.com : URI build
exception: Invalid query
```

then rerun the migration for the failed user.

Back-end Error

If your migration produces errors similar to the following:

```
[user@host.example.com] Validation exception: backend throws an error
(com.sun.comms.davserver.backends.BackendException: SQL error: Error in
allocating a connection. Cause: In-use connections equal max-pool-size and
expired max-wait-time. Cannot allocate more
connections.(INTERNAL_STORE_ERROR)) while doing nodeExists on
dav/home/user@host.example.com/calendar/dropbox/00000000000000000000000000000
while storing:
00000000000000000000000000000405ce44e000063da0000003100004aa0
```

then you might need to do the following:

1. Reduce the value of `davcore.serverlimits.maxmigrationthreads`.
   The default value is 2. Change the value to 2*(number of CPUs).
2. Change the MySQL connection pool size.
   The default is 32. Change this value to 4*(number of threads). To change this value, edit the `/etc/my.cnf` and increase the MySQL `max_connection` setting, for example,

```
max_connections = 200.
```

LDAP Error

If your migration using an LDAP filter produces an error like:

```
LDAP search failure: error result
```

then your filter might be too broad and you might be running into an LDAP search limit exceeded issue. Try narrowing down your filter.
For example, if your filter was:

```
./davadmin migration -u admin -X calmaster -L cs6.example.com:8080 -B
"o=dirbase" -R "objectclass=icscalendaruser"
```

change it so that it resembles the following:

```
./davadmin migration -u admin -X calmaster -L cs6.example.com:8080 -B
"o=dirbase" -R "(&(uid="a*")(objectclass=icscalendaruser))"
```

If you use this approach, you need to run multiple migration commands to complete the migration for the entire directory.

Read Timed Out Error

If your migration produces errors similar to the following:

```
2010-04-19_151418/master_log:[user@host.example.com] Exception on creation of
http://host-cs.example.com:80: Read timed out
```

then you might need to increase the `httpconnectiontimeout` configuration options.

To do so, change the `davcore.serverlimits.httpconnecttimeout` and `davcore.serverlimits.httpsockettimeout` options by using the `davadmin config` command.

Error Parsing iCal Data

If your migration produces errors similar to the following:

```
2010-04-20_094910/master_log:user@host.example.com Error parsing ical
data for :
000000000000000000000000000000000486c05b900001c6b000001c400000532: failed
to parse - Error at line 48:Illegal character in opaque part at index
27: user1@varrius.com\,user@example.com\,user1_smith@varrius.com
```

then you need to fix the line giving the error in the `ics` file located in the migration directory and import it.

You can use either the `davadmin import` or `client import` commands to import the file.

The following examples provide other potential errors of this type.

Error Parsing iCal Data Example 1

```
For
Error at line 32:I
illegal character in scheme name at index 6: mailto\:mattp@example.com
Change value to mailto:mattp@example.com
```

Error Parsing iCal Data Example 2

```
For
Error at line 38:I
llegal character in scheme name at index 6:
mailto\:mailto\\\\\\\:mailto\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\:mailto\\\\\\\\
value to mailto:ms104926@iplanet.com
```

Error Parsing iCal Data Example 3

```
For
Error at line 38:Illegal character in opaque part at index 16:
mailto:Marco@example Microsystems
Change value to a vaild email address like mailto:Marco@example.com
```

.

Owner Property Error

If the migration log shows the message "FAILED TO SET OWNER PROPERTY" when migrating a resource, there might be a problem with the "other owners" of the resource that you are migrating. A known issue with the Calendar Server 6 `csresource create` command could have caused the "other owners" field to be created incorrectly.

If you did the migration using the `-c` option, the log directory contains a `trace_information` directory for the resource. In the `trace_information` directory, search the `getcalprops*` file for the `X-NSCP-CALPROPS-OWNERS` line. If this line contains a comma separated list of users, these "other users" were set up incorrectly in Calendar Server 6. There should be one `X-NSCP-CALPROPS-OWNERS` line for each of the "other owners" for the resource.

If this is the case, correct the problem as follows:

1. Change the "other owners" by using the Calendar Server 6 `cscal` command with the `-y` option and space delimited user names.
   For example:

   ```
   ./cscal modify -y "" <uid>
   ./cscal modify -y "<1st owner> <2nd owner> ..."
   ```

   The first command clears out the "other owners" field and the second sets it correctly.

2. On Calendar Server 7, delete the account that you migrated and run the migration again.

Other Migration Errors

The section describes possible migration errors that you need to individually fix and import. The failed

`ics` files are located under the `calendar` subdirectory for individual users. Fix then import these files either by using the `davadmin import` command or by having the end users import their own files. In general, when fixing and importing make sure the values that used to be "calid" are changed to their "mail" equivalent in all `ics` components.

Topics in this section:

Multiple Components with the Same uid but Different Date Type: DATETIME_WITH_TZ, DATETIME_UTC

This error happens if two separate events or tasks end up with the same uid and the `dtstart`/`dtend` types do not match. Separate out the events into two separate `ics` files and import.

For example, this event:

```
BEGIN:VCALENDAR^M
PRODID:-//Sun/Calendar Server//EN^M
METHOD:PUBLISH^M
VERSION:2.0^M
X-NSCP-CALPROPS-LAST-MODIFIED:20031101T061300Z^M
X-NSCP-CALPROPS-CREATED:20010418T022506Z^M
X-NSCP-CALPROPS-READ:999^M
X-NSCP-CALPROPS-WRITE:999^M
X-NSCP-CALPROPS-RELATIVE-CALID:user@example.com^M
X-NSCP-CALPROPS-NAME:Business^M
X-NSCP-CALPROPS-PRIMARY-OWNER:user@example.com^M
X-NSCP-CALPROPS-TZID:America/New_York^M
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^wdeic^g^M
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^rsf^g^M
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^^g^M
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g^M
X-NSCP-CALPROPS-RESOURCE:0^M
X-S1CS-CALPROPS-ALLOW-DOUBLEBOOKING:1^M
X-NSCP-WCAP-ERRNO:0^M
BEGIN:VEVENT^M
UID:3bcdec24000054d40000000d00000db5^M
DTSTAMP:20100503T225631Z^M
SUMMARY: Meeting^M
CREATED:20011029T161741Z^M
```

```
LAST-MODIFIED:20091009T200936Z^M
PRIORITY:0^M
SEQUENCE:0^M
CLASS:PUBLIC^M
STATUS:CONFIRMED^M
TRANSP:OPAQUE^M
RRULE:FREQ=MONTHLY;WKST=MO;COUNT=41;INTERVAL=1;BYDAY=1FR^M
DTSTART;TZID=America/New_York:20011102T083000^M
DTEND;TZID=America/New_York:20011102T123000^M
X-S1CS-RECURRENCE-RDATELIST:20020405T133000Z^M
.....
END:VEVENT^M
BEGIN:VEVENT^M
UID:3bcdec24000054d40000000d00000db5^M
RECURRENCE-ID:20020405T133000Z^M
DTSTAMP:20100503T225631Z^M
SUMMARY:A Bday^M
DTSTART;VALUE=DATE:20020410^M
DTEND;VALUE=DATE:20020411^M
CREATED:20011017T203756Z^M
LAST-MODIFIED:20091009T200335Z^M
PRIORITY:0^M
SEQUENCE:0^M
CLASS:PUBLIC^M
STATUS:CONFIRMED^M
TRANSP:TRANSPARENT^M
X-NSCP-ORIGINAL-DTSTART:20020405T133000Z^M
X-NSCP-LANGUAGE:en^M
X-NSCP-DTSTART-TZID:America/New_York^M
X-NSCP-TOMBSTONE:0^M
X-NSCP-ONGOING:0^M
X-NSCP-GSE-COMPONENT-STATE;X-NSCP-GSE-COMMENT=REQUEST-COMPLETED:131074^M
END:VEVENT^M
....
```

```
END:VCALENDAR
```

can be split to the first master event with its timed exceptions and a new master event for the all day event:

```
BEGIN:VCALENDAR^M
PRODID:-//Sun/Calendar Server//EN^M
METHOD:PUBLISH^M
VERSION:2.0^M
X-NSCP-CALPROPS-LAST-MODIFIED:20031101T061300Z^M
X-NSCP-CALPROPS-CREATED:20010418T022506Z^M
X-NSCP-CALPROPS-READ:999^M
X-NSCP-CALPROPS-WRITE:999^M
X-NSCP-CALPROPS-RELATIVE-CALID:user@example.com^M
X-NSCP-CALPROPS-NAME:Business^M
X-NSCP-CALPROPS-PRIMARY-OWNER:user@example.com^M
X-NSCP-CALPROPS-TZID:America/New_York^M
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^wdeic^g^M
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^rsf^g^M
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^^g^M
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g^M
X-NSCP-CALPROPS-RESOURCE:0^M
X-S1CS-CALPROPS-ALLOW-DOUBLEBOOKING:1^M
X-NSCP-WCAP-ERRNO:0^M
BEGIN:VEVENT^M
UID:3bcdec24000054d40000000d00000db5^M
RRULE:FREQ=YEARLY
DTSTAMP:20100503T225631Z^M
SUMMARY:Camille Bday^M
DTSTART;VALUE=DATE:20020410^M
DTEND;VALUE=DATE:20020411^M
CREATED:20011017T203756Z^M
LAST-MODIFIED:20091009T200335Z^M
PRIORITY:0^M
SEQUENCE:0^M
CLASS:PUBLIC^M
TRANSP:TRANSPARENT^M
X-NSCP-LANGUAGE:en^M
X-NSCP-TOMBSTONE:0^M
X-NSCP-ONGOING:0^M
X-NSCP-GSE-COMPONENT-STATE;X-NSCP-GSE-COMMENT=REQUEST-COMPLETED:131074^M
END:VEVENT^M
END:VCALENDAR
```

Failed to store component: F0095280-BC13-11D9-81F6-000A958EAC78: validation error: PERCENT-COMPLETE with invalid value: -1

Edit `PERCENT-COMPLETE` to be between 0 and 100.

Error parsing ical data for: 3d6cd576000000a70000000a00000a55: failed to parse - Error at line 52:Illegal character in opaque part at index 27: MAILTO:user.one@example.com\n

Remove the trailing {{n} character or any other character that would be illegal in an email address.

Failed to store component: 9DE4F2DA-D020-11D8-9530-000A958A3252: validation error: Calendar must contain at least one component

In this case, the `ics` file resembles the following:

```
BEGIN:VCALENDAR^M
PRODID:-//Sun/Calendar Server//EN^M
METHOD:PUBLISH^M
VERSION:2.0^M
X-NSCP-CALPROPS-LAST-MODIFIED:20031101T061545Z^M
X-NSCP-CALPROPS-CREATED:20030916T095049Z^M
X-NSCP-CALPROPS-READ:999^M
X-NSCP-CALPROPS-WRITE:999^M
X-NSCP-CALPROPS-RELATIVE-CALID:user@example.com^M
X-NSCP-CALPROPS-NAME:User One^M
X-NSCP-CALPROPS-LANGUAGE:en^M
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^r^g^M
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^wdeic^g^M
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^sf^g^M
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^\^g^M
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g^M
X-NSCP-CALPROPS-RESOURCE:0^M
X-S1CS-CALPROPS-ALLOW-DOUBLEBOOKING:1^M
X-NSCP-WCAP-ERRNO:59^M
END:VCALENDAR^M
```

You may ignore the file as the event does not exist any longer.

Failed to store component: 00000000000000000000000000000000473defae0000696e000001460000129f: validation error: master is non recurring

Construct an `RRULE` property and add it to the master component.

Failed to store component: 9708780fccab40f18ace29226ef42a8e: validation error: failed to parse - Error at line 84:Illegal character in scheme name at index: =BASE64;VALUE=BINARY;X-S1CS-FILESIZE=49665:ATTACHFMTTYP//MESSAGE/OLEXS1CS

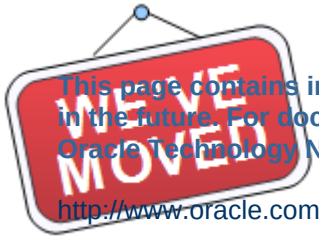
This error occurs when the attachment is missing. Edit the file and remove the line with the `ATTACH` property.

Error parsing ical data for: F0076B82-BC13-11D9-81F6-000A958EAC78: failed to parse - Error at line 51:[EXDATE] Unparseable date: "20000220"

This should not happen as long as the Calendar Server 6 host is patched with the most current Calendar Server patch. However, it you do see this error, edit the `EXDATE` property line and add `VALUE=DATE` `EXDATE;VALUE=DATE:20000220`.

Chapter 22. Creating a Calendar Server 6 and Calendar Server 7 Coexistent Deployment

Creating a Calendar Server 6 and Calendar Server 7 Coexistent Deployment



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This information describes how to set up Calendar Server 7 (CalDAV Server) in an existing Calendar Server 6 deployment. In this environment, you have some users who have migrated to Calendar Server 7 and some users that still exist on Calendar Server 6. In such a deployment, you enable both free/busy lookup and iCalendar Message-Based Interoperability Protocol (iMIP) invitation between the two Calendar Server deployments. That is, users should have the capability to check free/busy information of users on any server, and the capability to invite any user. Invitations to users on a different server would be delivered by using iMIP.

> **Note**
> In this coexistent deployment, users are either on Calendar Server 7 or Calendar Server 6. They do not have calendars on both Calendar Server 7 and Calendar Server 6.

For more information on performing a Calendar Server 7 migration, see Migrating From Sun Java System Calendar Server 6 to Calendar Server 7.

Topics:

Calendar Coexistence Overview

For coexistence to work, each calendar server needs to be able to determine if users are Calendar Server 6 or Calendar Server 7 users.

On Calendar Server 7 servers:

- Users are identified as Calendar Server 7 users by the presence in their Directory Server LDAP entry of a particular attribute which is configured as described in To Configure the Calendar Server 7 Environment. For more information on how Calendar Server uses Directory Server, see Calendar Server and Directory Server Integration.
- All other users that are part of the deployment but do not have this attribute set are considered Calendar Server 6 users

On Calendar Server 6 servers:

- Users are identified as Calendar Server 6 users if they already have a calendar created in the data base.
- All other users are considered Calendar Server 7 users.

For coexistence to function correctly, you must disable user auto-provisioning on all Calendar Server 6 servers.

In addition, the same unique ID attribute that you use for Calendar Server 7 users needs to be present in all Calendar Server 6 users too. This attribute defines the unique value used as the database identifier for each account. This value is used internally to identify a user in other user's access control entries, subscription entries, and so on. The attribute chosen as the unique ID attribute must be present in all user, group, and resource LDAP entries for the deployment.

For example, if Calendar Server 6 uses `uid` as its unique identifier and you configure Calendar Server 7 to use `davuniqueid` as the unique identifier, you need to add the new unique identifier not only to the users migrated to Calendar Server 7, but also to the users remaining on Calendar Server 6 for co-existence to work.

For information on migrating a Calendar Server 6 deployment to Calendar Server 7, see Migrating From Sun Java System Calendar Server 6 to Calendar Server 7.

Minimum Software Requirements

- Calendar Server 6.3: At least patch level 44. (See Calendar Server 6.3 Patches by Release for more information.)
- Calendar Server 7: At least Calendar Server 7 Update 1.

To Configure the Calendar Server 7 Environment

Perform the following steps on the Calendar Server 7 host.

1. Change to the *cal-svr-base*`/config` directory and either edit or create the `ischeduledomainmap.properties` file.
2. Add a line that contains your domain name and the Calendar Server 6 URL to connect to for Calendar Server 6 user inquiry. If the Calendar Server 6 host is set up for multiple front ends and back ends, you can use any front-end server information.
   For example:

   ```
   example.com=http://cs6server.example.com:8080/ischedule/
   ```

3. Set `davcore.scheduling.localuserattr` parameter, which identifies an LDAP entry as that belonging to a Calendar Server 7 account. The recommended value is `davstore`.
   For example:

   ```
   davadmin config modify -o davcore.scheduling.localuserattr -v
   davstore
   ```

4. Add the same value that you used for the `davcore.scheduling.localuserattr` parameter to the `davcore.uriinfo.subjectattributes` parameter.
   For example:

   ```
   davadmin config modify -o davcore.uriinfo.subjectattributes -v
   davstore
   ```

   If you are using an attribute other than `davstore` for the `davcore.scheduling.localuserattr` parameter, add it to the `davcore.uriinfo.subjectattributes` parameter too. Use the `davadmin config` command to first retrieve the current value of `davcore.uriinfo.subjectattributes`, then add the new attribute at the end of the list.
   For example:

   ```
   davadmin config -o davcore.uriinfo.subjectattributes
   davcore.uriinfo.subjectattributes: cn davstore icsstatus mail

   davadmin config -o modify davcore.uriinfo.subjectattributes -v "cn
   davstore icsstatus mail xcs7user"
   ```

5. On the Directory Server, use an LDAP client to verify that the LDAP attribute defined is present for all users who are on the Calendar Server 7 server. Also verify that the attribute is not present for users on the Calendar Server 6 host.
   If using `davStore` and it is a simple single back-end deployment, populate it with the value `defaultbackend`.

To Configure the Calendar Server 6 Environment

Perform the following steps on the Calendar Server 6 host.

1. Patch the server to the following:
   - 121657-37 - Solaris SPARC
   - 121658-37 - Solaris x86
   - 121659-37 - Red Hat Linux
2. Edit the `ics.conf` file as follows:
   For Solaris OS, edit `/etc/`*cal-svr-base*`SUNWics5/config/ics.conf`.
   For Red Hat Linux, edit `/etc/`*cal-svr-base*`/calendar/config/ics.conf`.
   - Set `service.http.caldavcompatible = "yes"`.
   - Set the option for free/busy redirection to point to the Calendar Server 7 server's free/busy URL. If the Calendar Server 7 servers have a multiple front-end and back-end setup, use any front-end's information. The `service.wcap.freebusy.redirecturl` parameter must include the port number of the Calendar Server 7 host.
     For example:

     ```
     service.wcap.freebusy.redirecturl =
     "http://cs7u1server.example.com:8080/davserver/wcap/get_freebusy
     ```

   - Disable autoprovisioning by setting the value of `local.autoprovision`, `user.invite.autoprovision`, `group.invite.autoprovision`, and `resource.invite.autoprovision` to `"no"`.
     For example, the changes resemble the following:

```
service.http.caldavcompatible = "yes"
service.wcap.freebusy.redirecturl =
"http://cs7u1server.example.com:8080/davserver/wcap/get_freebusy
="no"
user.invite.autoprovision = "no"
group.invite.autoprovision = "no"
resource.invite.autoprovision = "no"
```

Note
In the preceding examples, the `service.wcap.freebusy.redirecturl` is directed to a Calendar Server 7 Update 1 host, with a port assignment of 8080.

3. Restart the Calendar Server 6 server, for example:

```
cd /opt/sun/comms/calendar/SUNWics5/cal/sbin
./stop-cal
./start-cal
```

4. For migrated Calendar Server 6 users, perform the following steps on the Calendar Server 6 host:
   a. Delete Calendar Server 6 calendars by using `cscal delete -o` *uid*.
   b. Set the LDAP attribute `icsAllowedServiceAccess` to `-http:all`.
      The setting ensures that Calendar Server 6.3 users no longer log in by mistake and cause calendars to be autocreated.

Workflow Description of This Configuration

This section contains the following topics:

- About the Calendar Server 7 Configuration
- About the Calendar Server 6 Configuration

About the Calendar Server 7 Configuration

When a scheduling invitation or free/busy request comes to a Calendar Server 7 host, the server performs a directory lookup to determine if each of the invitees is local. If the server option `davcore.scheduling.localuserattr` is set, the server additionally checks if the attribute specified by that option is set for each of the invitees found in the directory. If the attribute is set, the user is considered local to the Calendar Server 7 host. If the attribute is not set, the user is assumed to be on the Calendar Server 6 host.

If a user is on the Calendar Server 6 host, for scheduling free/busy lookup, the server uses the Calendar Server 6 host information in `ischeduledomainmap.properties` and contacts the Calendar Server 6 host to make an iSchedule free/busy request.

Sample request:

```
>> Request <<

   POST /ischedule/ HTTP/1.1
   Host: cs6server.example.com
   Content-Type: text/calendar
   Content-Length: xxxx

   BEGIN:VCALENDAR
   VERSION:2.0
   PRODID:-//Sun Microsystems/Sun Calendar Server 7.0-0.13//EN
   METHOD:REQUEST
   BEGIN:VFREEBUSY
   DTSTAMP:20040901T200200Z
   ORGANIZER:mailto:cs7user@example.com
   DTSTART:20040902T000000Z
   DTEND:20040903T000000Z
   UID:34222-232@example.com
   ATTENDEE;CN=Cal7 User:mailto:cs7user@example.sun.com
   END:VFREEBUSY
   END:VCALENDAR
```

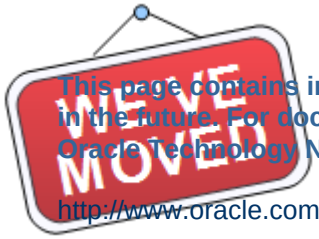
The response is incorporated into the scheduling free/busy response made by the Calendar Server 7 host. If a user is on a Calendar Server 6 host, for scheduling an invitation, an iMIP invitation is sent by the Calendar Server 7 host.

About the Calendar Server 6 Configuration

On a Calendar Server 6 host, setting the `service.http.caldavcompatible` configuration parameter to `"yes"`, enables the ability to handle iSchedule style free/busy requests from the Calendar Server 7 host for users still on Calendar Server 6. The value for the `service.wcap.freebusy.redirecturl` configuration parameter specifies the free/busy redirect URL returned by the Calendar Server 6 host for an invitee not found in its local database. Setting the autoprovision configuration parameters to `"no"` (`local.autoprovision`, `user.invite.autoprovision`, `group.invite.autoprovision`, and `resource.invite.autoprovision`) ensures that on a scheduling invitation request, if the invitee calendar is not found on the local database, an iMIP invitation is sent instead of creating an account in the database.

Chapter 23. Setting Up and Managing Calendar Server Security

Setting Up and Managing Oracle Communications Calendar Server Security



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This information provides an overview about security for the Oracle Communications Calendar Server (also known as Calendar Server 7) product. It also provides links to security topics that provide more indepth information for configuring and administering Calendar Server security.

Topics:

- Overview of Calendar Server
- Secure Installation and Configuration
- Security Features
- Detecting Possible Security Issues

Overview of Calendar Server

For an overview of the product, see Introduction to Calendar Server 7. For information on general security principals, such as security methods, common security threats, and analyzing your security needs, see Designing for Security. For an overview of operating system security, see Oracle Solaris Security for System Administrators.

Secure Installation and Configuration

Topics in this section:

- Installation Overview
- Installing Infrastructure Components
- Installing Calendar Server Components
- Post Installation Configuration

Installation Overview

This section outlines the planning process for a secure installation and describes several recommended

deployment topologies for the systems.

Understanding Your Environment

To better understand your security needs, ask yourself the following questions:

1. Which resources am I protecting?
   In a Calendar Server production environment, consider which of the following resources you want to protect and what level of security you must provide:
   - Calendar Server front end
   - Calendar Server back end (MySQL Server or Oracle Database)
   - Document store (can be local, remote, or dbdocstore)
   - Dependent resources, such as GlassFish Server, Directory Server, and Messaging Server
2. From whom am I protecting the resources?
   In general, resources must be protected from everyone on the Internet. But should the Calendar Server deployment be protected from employees on the intranet in your enterprise? Should your employees have access to all resources within the GlassFish Server environment? Should the system administrators have access to all resources? Should the system administrators be able to access all data? You might consider giving access to highly confidential data or strategic resources to only a few well trusted system administrators. On the other hand, perhaps it would be best to allow no system administrators access to the data or resources.
3. What will happen if the protections on strategic resources fail?
   In some cases, a fault in your security scheme is easily detected and considered nothing more than an inconvenience. In other cases, a fault might cause great damage to companies or individual clients that use Calendar Server. Understanding the security ramifications of each resource help you protect it properly.

Deployment Topologies

You can deploy Calendar Server on a single host or on multiple hosts, splitting up the components into multiple front-end calendar hosts and multiple back-end hosts. You can also install the document store onto a separate host. For more information, see the following information:

- Calendar Server 7 Deployment Planning
- Developing a Communications Suite Logical Architecture

The general architectural recommendation is to use the well-known and generally accepted Internet-Firewall-DMZ-Firewall-Intranet architecture. For more information on addressing network infrastructure concerns, see Determining Your Communications Suite Network Infrastructure Needs.

Installing Infrastructure Components

Calendar Server is deployed within GlassFish Server. For information on how to install and configure GlassFish Server, see Oracle GlassFish Server Installation Guide. To operate GlassFish Server in secure mode, see Secure Administration Overview. For information on how to configure GlassFish Server to use a certificate issued by a Certification Authority (CA) to establish secure sessions through secure sockets layer (SSL) technology, see To Configure GlassFish Enterprise Server to Use a CA Signed Certificate for SSL. For more information, see the Oracle GlassFish Security Guide.

Calendar Server can use either MySQL Server or Oracle Database as the database for storing calendar information. For information on how to install and configure MySQL Server securely, see To Initialize and Configure MySQL Server. For more information on installing MySQL, see Security in MySQL. For information on how to install and configure Oracle Database securely, see To Install and Create an Oracle Database Instance for Calendar Server.

Installing Calendar Server Components

See Installation Scenario - Calendar Server.

The installation prompts for authentication credentials for the following:

- Database user
- GlassFish administrator
- Directory Server manager (bind DN and password)
- CalDAV administrator

Post Installation Configuration

The high-level steps to configuring Calendar Server for a secure deployment include:

1. Making sure that HTTPS is configured correctly on the front-end GlassFish Server host:
    a. Making use of a CA signed certificate
    b. Setting SSL port to default port of 443 to ease client configurations
    c. Changing `fulluriprefix` configuration option
2. Disabling HTTP on front-end GlassFish Server host
3. Configuring JMX Port for GlassFish Server 3 to Use SSL
4. Enabling LDAP SSL, if not previously done
5. Enabling secure notification mail submission
6. Configuring SSL on back ends
    a. Setting up secure communication to the Calendar Server database, either MySQL Server or OracleDB
    b. Setting up secure communications to the remote document store
7. Changing uuid if still using `nsUniqueId`
8. Disabling Elliptic Curve Encryption to workaround a Java 6 issue
9. Adding LDAP access control for Calendar Server
10. Securing iSchedule Communication

For instructions, see Configuring Calendar Server for a Secure Deployment.

Security Features

This section outlines the specific security mechanisms offered by Calendar Server.

Topics:

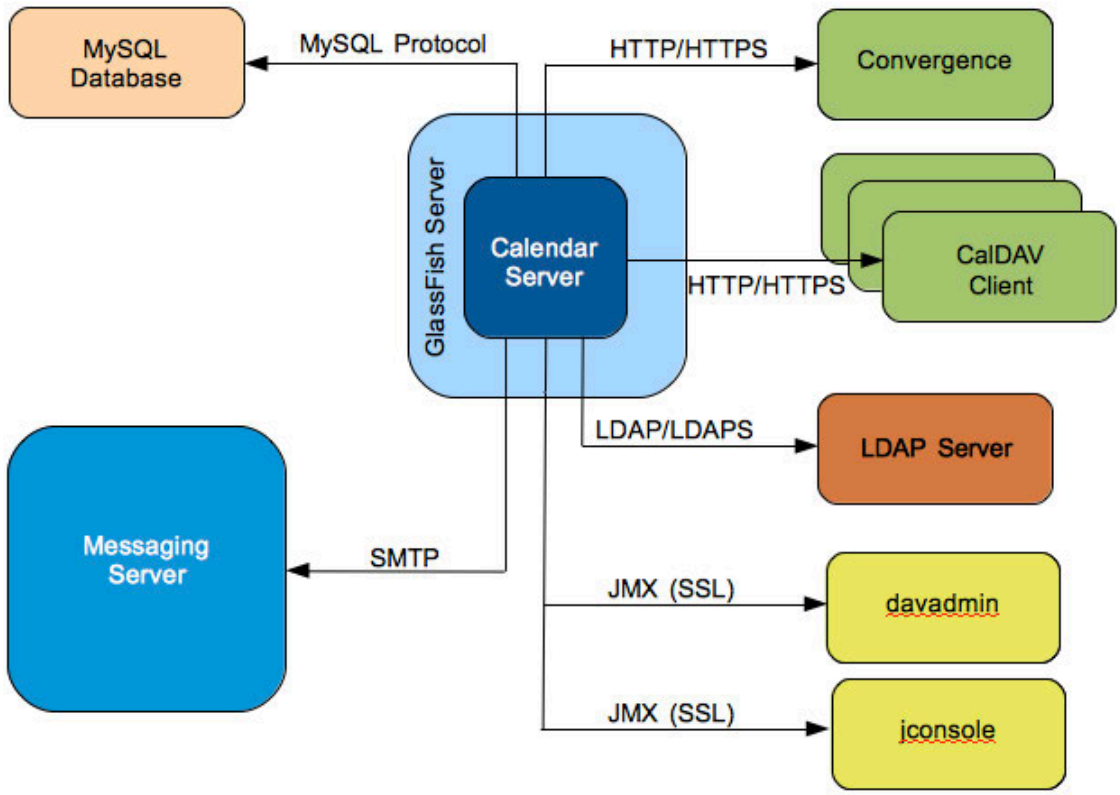
- About System Security in Calendar Server
- The Security Model
- Configuring and Using Authentication
- Configuring and Using Access Control
- Configuring and Using SSL
- Configuring and Using Secure iSchedule
- Making Calendar Server and GlassFish Server 3 Secure

About System Security in Calendar Server

Calendar Server provides a number of security levels to protect users against eavesdropping, unsanctioned usage, or external attack. The basic level of security is through authentication. Calendar Server uses LDAP authentication.

The following figure shows the protocols and communication flows used by Calendar Server that can be secured. The HTTPS, LDAPS, JMX, and SMTPS protocols need to be secured by using SSL. SQL/JDBC connections between GlassFish Server and the database are secured by using SSL as described in Securing Communications to Calendar Server Back Ends.

Calendar Server Protocols



In the preceding figure, HTTPS and SSL provide encryption of data between the server and the respective components. For information on securing SMTP for notifications, see the `notification.dav.smtp.*` parameters in Calendar Server 7 Configuration Parameters. For information on LDAPS, see Enabling LDAP SSL in Calendar Server. For information on securing connections to the MySQL database, see Using SSL for Secure Connections. For more information on designing a secure deployment, see Designing for Security.

The Security Model

Security requirements arise from the need to protect data: first, from accidental loss and corruption, and second, from deliberate unauthorized attempts to access or alter that data. Secondary concerns include protecting against undue delays in accessing or using data, or even against interference to the point of denial of service. The global costs of such security breaches run up to billions of dollars annually, and the cost to individual companies can be severe, sometimes catastrophic.

The critical security features that provide these protections are:

- Authentication
- Access Control

Authentication is the way in which an entity (a user, an application, or a component) determines that another entity is who it claims to be. An entity uses security *credentials* to authenticate itself. The credentials might be a user name and password, a digital certificate, or something else. Usually, servers or applications require clients to authenticate themselves. Additionally, clients might require servers to authenticate themselves. When authentication is bidirectional, it is called *mutual authentication*.

Calendar Server supports LDAP authentication.

Access Control, also known as authorization, is the means by which users are granted permission to access data or perform operations. After a user is authenticated, the user's level of authorization determines what operations the user can perform.

Calendar Server uses Access Control Lists (ACLs) to determine access control for calendars and scheduling. For more information, see Overview of ACLs.

Configuring and Using Authentication

For information on Calendar Server and LDAP authentication, see Provisioning Calendar Server Users.

Configuring and Using Access Control

For information on configuring access control for calendars and scheduling, see Administering Calendar Server Access.

Configuring and Using SSL

Starting with **Calendar Server 7.0.4.14.0**, you can configure the Secure Sockets Layer (SSL) protocol between the Calendar Server front ends and back ends, including the calendar store and the remote document store. For information on configuring SSL between the Calendar Server front ends and back ends, see Securing Communications to Calendar Server Back Ends.

Configuring and Using Secure iSchedule

Starting with **Calendar Server 7.0.4.14.0**, you can specify which hosts are allowed to send iSchedule POST requests. You can also limit this list of hosts to just the SMTP server that uses iSchedule for automatic iMIP handling. For more information on configuring secure iSchedule communication, see Securing iSchedule Port.

Making Calendar Server and GlassFish Server 3 Secure

Topics in this section:

- Preventing Denial of Service Attacks on GlassFish Server 3
- Configuring JMX Port for GlassFish Server 3 to Use SSL

Preventing Denial of Service Attacks on GlassFish Server 3

Using GlassFish Server 3, you can prevent a Denial of Service (DoS) attack against the server by:

- Limiting the size of a POST request
- Specifying a request timeout value
- Creating a blacklist of host names and/or IP addresses

For more information, see Configuring GlassFish Server 3 Denial of Service Prevention.

Configuring JMX Port for GlassFish Server 3 to Use SSL

GlassFish 3 does not enable the JMX port with SSL by default. If you want to make the JMX communications secure, you need to enable security for GlassFish, either through the GlassFish Administration Console, or through the `asadmin` command. For more information, see Creating Secure Communications Between davadmin and Calendar Server.

Detecting Possible Security Issues

You can use the log files to look for possible security problems. This section lists a few possible log messages to scan for. For information on using log files, see Administering Calendar Server 7 Logging.

Login errors resemble the following:

```
INFO [2011-06-08T11:20:44.529-0600] <...LDAPLoginModule.lookupUser>
Error while retrieving user info for user <user>: No results found
```

If you have the logging level set to FINEST, then you see something resembling the following when a login error occurs:

```
FINEST [2011-06-08T11:36:56.304-0600] <...WCAPServlet.service> failed
login or session timeout
```

If someone is trying to bypass the iCal data parsing, you see iCal warnings such as this:

```
FINE [2011-06-08T11:39:53.426-0600] <...RESTServlet.service> Got a non
standard condition: failed to parse - Error at line 4:Illegal property
[BEGI]
```

An unusually high activity of requests (REQ) from the same IP address shows up as follows in the commands log file:

```
Sample request log entry....
[2011-06-07T03:39:05.454-0700] <3887> DavServlet [REQ] REPORT
/dav/home/user/calendar/ <IP_address> server:port
```

Chapter 24. Configuring SSL for the Calendar Server MySQL Back End

Configuring SSL for the Calendar Server MySQL Back End

This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This capability is available starting in **Calendar Server 7.0.4.14.0**.

You can enhance your security by configuring SSL communication between the Calendar Server front ends and back ends. To do so, first enable the back-end database servers for SSL by setting up either the required `trustStore` files or wallets. Then configure the Calendar Server front ends to connect over SSL by making use of the stored certificates.

You can also configure SSL communication between the Calendar Server front ends and the remote document stores. See To Configure Remote Document Store SSL.

Topic:

- To Configure SSL for the Calendar Server MySQL Back End

To Configure SSL for the Calendar Server MySQL Back End

These instructions use a self-signed certificate for the MySQL server and the MySQL Connector/J client within GlassFish. These instructions also assume that the `/etc/mysql` directory already exists.

1. Create your own Certificate Authority and use it to sign a server certificate for the MySQL server instance and a client certificate for use with the MySQL Connector/J.
   For example:

```
shell> cd /etc/mysql

# Create CA certificate
shell> openssl genrsa 2048 > ca-key.pem
shell> openssl req -new -x509 -nodes -days 3650 \
         -key ca-key.pem -out ca-cert.pem

# Create server certificate, remove passphrase, and sign it
# server-cert.pem = public key, server-key.pem = private key
shell> openssl req -newkey rsa:2048 -days 3650 \
         -nodes -keyout server-key.pem -out server-req.pem
shell> openssl rsa -in server-key.pem -out server-key.pem
shell> openssl x509 -req -in server-req.pem -days 3650 \
         -CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 -out
server-cert.pem

# Create client certificate, remove passphrase, and sign it
# client-cert.pem = public key, client-key.pem = private key
shell> openssl req -newkey rsa:2048 -days 3650 \
         -nodes -keyout client-key.pem -out client-req.pem
shell> openssl rsa -in client-key.pem -out client-key.pem
shell> openssl x509 -req -in client-req.pem -days 3650 \
         -CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 -out
client-cert.pem

# Verify both the self-signed client and server cert
shell> openssl verify -CAfile ca-cert.pem server-cert.pem
client-cert.pem
server-cert.pem: OK
client-cert.pem: OK
```

2. Enable SSL in the MySQL instance.
   a. Stop the MySQL instance, if running.
   b. In the [mysqld] section of the `my.cnf file`, add the following configuration parameters:

```
ssl-ca=/etc/mysql/ca-cert.pem
ssl-cert=/etc/mysql/server-cert.pem
ssl-key=/etc/mysql/server-key.pem
```

3. To verify that the MySQL instance is now enabled for SSL, run the `mysql` command-line tool to check the global variable `have_ssl`.
   For example:

```
shell> /opt/mysql/mysql/bin/mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10079
Server version: 5.5.28-enterprise-commercial-advanced MySQL
Enterprise Server - Advanced Edition (Commercial)

Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql> show variables like 'have_ssl';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_ssl      | YES   |
+---------------+-------+
1 row in set (0.00 sec)

mysql> quit
Bye
shell>
```

4. If you want to run the `mysql` command-line tool with SSL, use the `--ssl-ca` option. For example:

```
shell> ./mysql --ssl-ca=/etc/mysql/ca-cert.pem -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10089
Server version: 5.5.28-enterprise-commercial-advanced MySQL
Enterprise Server - Advanced Edition (Commercial)

Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql> \s
--------------
./mysql  Ver 14.14 Distrib 5.5.28, for solaris10 (sparc) using
EditLine wrapper

Connection id:          10089
Current database:
Current user:           root@localhost
SSL:                    Cipher in use is DHE-RSA-AES256-SHA
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server version:         5.5.28-enterprise-commercial-advanced MySQL
Enterprise Server - Advanced Edition (Commercial)
Protocol version:       10
Connection:             Localhost via UNIX socket
Server characterset:    utf8
Db     characterset:    utf8
Client characterset:    latin1
Conn.  characterset:    latin1
UNIX socket:            /tmp/mysql.sock
Uptime:                 2 days 23 hours 55 min 2 sec

Threads: 25  Questions: 104093  Slow queries: 0  Opens: 58  Flush
tables: 1  Open tables: 51  Queries per second avg: 0.402
--------------

mysql> quit
Bye
shell>
```

The output from the `\s` command shows that SSL is in use with the cipher information.

5. For the MySQL Connector/J in GlassFish to communicate with the MySQL server in SSL, put your Certificate Authority and the client certificate into a JKS `trustStore` and `keystore` respectively, and use them in the JDBC connection pool setup.
   For example, while remaining in the `/etc/mysql` directory, perform the following commands:

```
shell> keytool -importcert -file ca-cert.pem -keystore cacerts.jks
-storepass secret -storetype JKS
shell> keytool -importcert -file client-cert.pem -keystore
keystore.jks -storepass secret -storetype JKS
```

6. Stop GlassFish Server.
7. Add the following parameters in the `domain.xml` file to the existing `caldavPool` and `ischedulePool` JDBC connection pool definition:

```
<property name="useSSL" value="true"/>
<property name="requireSSL" value="true"/>
<property name="trustCertificateKeyStoreUrl"
value="file:///etc/mysql/cacerts.jks"/>
<property name="trustCertificateKeyStoreType" value="JKS"/>
<property name="trustCertificateKeyStorePassword" value="secret"/>
<property name="clientCertificateKeyStoreUrl"
value="file:///etc/mysql/keystore.jks"/>
<property name="clientCertificateKeyStoreType" value="JKS"/>
<property name="clientCertificateKeyStorePassword" value="secret"/>
```

8. Optional: To log the SSL communication with GlassFish Server, add the following to the JVM options in the `domain.xml` before starting GlassFish Server.

```
<jvm-options>-Djavax.net.debug=ssl</jvm-options>
```

   This setting causes messages such as the following to appear in the GlassFish Server `server.log` file:

```
[#|2013-03-25T16:11:52.799-0700|INFO|sun-appserver2.1|javax.enterpris
WRITE: TLSv1 Handshake, length = 163|#]

[#|2013-03-25T16:11:52.846-0700|INFO|sun-appserver2.1|javax.enterpris
READ: TLSv1 Handshake, length = 74|#]

[#|2013-03-25T16:11:52.846-0700|INFO|sun-appserver2.1|javax.enterpris
ServerHello, TLSv1|#]
```

9. Start GlassFish Server.

Chapter 25. Configuring SSL for the Calendar Server OracleDB Back End

Configuring SSL for the Calendar Server OracleDB Back End

This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This capability is available starting in **Calendar Server 7.0.4.14.0**.

You can enhance security by configuring SSL communication between the Calendar Server front and back ends. First, you enable the Oracle back-end database server for SSL by setting up the required Oracle wallet and Oracle Net Listener. Then, you configure the Calendar Server front ends to connect over SSL by setting the properties on the JDBC connection pool setting.

You can also configure SSL communication between the Calendar Server front ends and the remote document stores. See To Configure Remote Document Store SSL.

For more information about configuring Oracle Database with SSL, see, see SSL with Oracle JDBC Thin Driver.

Topics:

- To Install the Database Server Certificate
- To Enable the TCP/IP SSL Listener in Oracle Net Services
- To Complete Installation and Modify JDBC Connection Pool Settings

To Install the Database Server Certificate

You can install the OracleDB server certificate in an Oracle wallet that is accessible by the Oracle Net listener:

1. Use the `orapki` tool to create the Oracle wallet and a server certificate signing request. For more information about creating wallets, see Managing Oracle Wallets with the `orapki` Utility. For specific commands, see `orapki` Usage Examples.
2. When the certificate is signed by a trusted certificate authority, add it to the wallet for the Oracle Net configuration. For more information about using orapki to add a certificate to a wallet, see `orapki` Utility Syntax.
3. If the certificate authority is not known to the GlassFish Server instance hosting the Calendar

Server front end, import the certificate authority's certificate into the GlassFish Server JVM. Depending on the GlassFish Server version, this is located in one of the following:

- In the JRE CA certificate store that is typically available at:

```
<javahome>/jre/lib/security/cacerts
```

  where *javahome* is the location of the JDK used by GlassFish Server.

- In the GlassFish Server instance `config /cacerts.jks` store:

```
<gfhome>/glassfish3/glassfish/domains/domain1/config/cacerts.jks
```

Or, you can use a self-signed certificate:

1. Use the `orapki` tool to create the root and server certificates, and use the root certificate to sign the server certificate. See Appendix B in SSL with Oracle JDBC Thin Driver.
2. The Oracle wallet and the self-signed server certificate are used in the Oracle Net configuration.
3. On the GlassFish Server machine hosting the Calendar Server front end, import the root certificate into the GlassFish Server JVM as described in Step 3.

To Enable the TCP/IP SSL Listener in Oracle Net Services

To enable the Oracle Net listener for SSL communication, you must modify three configuration files for Oracle Net Services as shown in the following examples.

1. Manually edit the `sqlnet.ora`, `tnsnames.ora` and `listener.ora` files in `$ORACLE_HOME/network/admin`. In the following examples, replace values for host name, wallet location, and port numbers to match your configuration:

sqlnet.ora

```
# sqlnet.ora Network Configuration File:
/u01/app/oracle/product/11.2.0/dbhome_1/network/admin/sqlnet.ora
# Generated by Oracle configuration tools.

SQLNET.AUTHENTICATION_SERVICES= (BEQ, TCPS)

SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER= (SHA1, MD5)

SSL_VERSION = 0

TRACE_LEVEL_CLIENT = SUPPORT

NAMES.DIRECTORY_PATH= (TNSNAMES, EZCONNECT)

SSL_CLIENT_AUTHENTICATION = FALSE

TRACE_LEVEL_SERVER = SUPPORT

SQLNET.CRYPTO_SEED = 'kjlkwebflk090kj92hjkky9hsjkhdhhhwjk'

SQLNET.ENCRYPTION_TYPES_SERVER= (3DES168, AES256, RC4_256, AES192,
AES128, RC4_128, 3DES112)

WALLET_LOCATION =
  (SOURCE =
    (METHOD = FILE)
    (METHOD_DATA =
      (DIRECTORY = /local/oracle/wallet/server)
    )
  )

SSL_CIPHER_SUITES= (SSL_RSA_WITH_AES_256_CBC_SHA,
SSL_RSA_WITH_AES_128_CBC_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_RSA_WITH_RC4_128_SHA, SSL_RSA_WITH_RC4_128_MD5,
SSL_RSA_WITH_DES_CBC_SHA)

ADR_BASE = /u01/app/oracle
```

tnsnames.ora

```
# tnsnames.ora Network Configuration File:
/u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora
# Generated by Oracle configuration tools.

LISTENER_ORCL =
  (ADDRESS = (PROTOCOL = TCP)(HOST = dbhost.example.com)(PORT =
1521))
  (ADDRESS = (PROTOCOL = TCPS)(HOST = dbhost.example.com)(PORT =
2484))


ORCL =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCPS)(HOST = dbhost.example.com)(PORT
= 2484))
      (ADDRESS = (PROTOCOL = TCP)(HOST = dbhost.example.com)(PORT =
1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = orcl.us.oracle.com)
    )
  )
```

listener.ora

```
# listener.ora Network Configuration File:
/u01/app/oracle/product/11.2.0/dbhome_1/network/admin/listener.ora
# Generated by Oracle configuration tools.

SSL_CLIENT_AUTHENTICATION = FALSE

WALLET_LOCATION =
  (SOURCE =
    (METHOD = FILE)
    (METHOD_DATA =
      (DIRECTORY = /local/oracle/wallet/server)
    )
  )

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
    )
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = dbhost.example.com)(PORT =
1521))
    )
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCPS)(HOST = dbhost.example.com)(PORT
= 2484))
    )
  )

ADR_BASE_LISTENER = /u01/app/oracle

TRACE_LEVEL_LISTENER = SUPPORT
```

2. Restart the listener using the `lsnrctl` command.

Or, if you prefer, you can use the Oracle Net Manager graphical user interface (GUI) tool to configure the Oracle Net Services values for profile, name service, and listener. Refer to the examples above when entering values in the GUI.

1. Run `netmgr` from the command line.
2. Expand Oracle Net Configuration, and select Profile under the local configuration icon.
3. Under the Oracle Advanced Security pulldown menu, click the SSL tab.
4. Type the path to the Oracle wallet for the server. You can also add various cipher suites for use in the SSL negotiation.
5. Click the Encryption tab. Choose an encryption method for your SSL configuration.
6. Click the Integrity tab. Choose a data integrity method for your SSL configuration.
7. Under Oracle Net Configuration, expand Service Naming and select the local service icon.
8. Under Address Configuration, click the Address 1 tab, choose the TCP/IP with SSL protocol, and type the host name and port number.
9. To add the additional TCP/IP with SSL address to the listener, under Oracle Net Configuration, expand Listeners and select the listener icon.
10. Click the Address3 tab, choose the TCP/IP with SSL protocol, and type the host name and port number.

11. From the File menu, choose Save Network Configuration.

To Complete Installation and Modify JDBC Connection Pool Settings

1. Install and configure Oracle Communications Calendar Server to communicate with the Oracle database using the non-SSL port.
2. Modify the `caldavPool` and `iSchedulePool` JDBC connection pool settings to use the TCPS protocol to communicate in SSL.
   Change the URL property in the JDBC connection pool setting by editing the `HOST`, `PORT`, and `SERVICE_NAME` according to your deployment.
   For example:

   ```
   jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=dbhost.examp
   ```

Chapter 26. Creating, Exporting, and Importing SSL Certificates

Creating, Exporting, and Importing SSL Certificates

This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This information describes how to create, export, and import SSL certificates, including both self-signed and CA (certificate authority) certificates.

Topics:

Overview of Self-signed and Certificate Authority Certificates

An SSL certificate is necessary for transmission of encrypted data between a client and a server. A self-signed certificate is one that you create for your server, in the server's KeyStore. You then export and import the exported certificate into the client's TrustStore. A certificate authority (CA) certificate (or CA-signed certificate) is a certificate that has been issued by a trusted third party. To obtain a CA-signed certificate, you create a request file from your self-signed certificate and send it to a certificate authority for approval. You then import this CA-signed certificate into the server's KeyStore, replacing the self-signed certificate.

One advantage to using a CA-signed certificate instead of a self-signed certificate is that you do not need to import the CA-signed certificate into the client's TrustStore. The client already has a trusted root certificate for that CA either in the Glassfish Server instance or in the browser itself.

Creating a Self-signed Certificate

In general, you use the Java `keytool` command to create a self-signed certificate on the same server where the KeyStore is located. If you create the self-signed certificate on another server, you need to transfer it from that server to the server where it will be used to create the KeyStore.

1. Decide on a certificate name.

At a minimum, the certificate file should have `.jks` as its extension.

2. Determine if the KeyStore file already exists on the server.

```
keytool -list -v -keystore <path_to_keystore_file>
```

3. Generate the self-signed certificate and place it in the KeyStore.

```
keytool -genkeypair -alias <alias> -keyalg RSA -validity
<#_of_days> -keysize 2048 -keystore <path_to_keystore_file>
```

where:

- *alias*: Specifies a word of your choice, for example, the fully qualified domain name of the server host.
- *#_of_days*: Specifies the number of days that the certificate is to be valid.
- `2048` is the recommended key size.
- *path_to_keystore_file*: Specifies the path to the KeyStore. This file or its parent directory must exist.

1. If you are going to have the certificate signed by a CA, you can skip to Creating a CA-signed Certificate Request. However, while you are waiting for the CA to return your certificate, you can use your self-signed certificate by continuing with the next steps.
2. Export the certificate needs to a certificate file.

```
keytool -export -alias <alias> -keystore <path_to_keystore_file>
-rfc -file <path_to_certificate_file>
```

where:

- *alias*: Specifies the same alias that was used to generate the certificate.
- *path_to_keystore_file*: Specifies the same KeyStore path that was used to generate the certificate.
- *path_to_certificate_file*: Specifies the exported certificate file, often given an extension of `.cert`.

Installing the Self-signed Certificate on the Client

The certificate file generated in the previous step needs to be transferred to the client host and imported into the truststore of the client.

- To import the certificate into the truststore of your client:

```
keytool -importcert -alias <alias> -file <path_to_certificate_file>
-keystore <truststore_file>
```

where:

- *alias*: Specifies a word of your choice, usually the same word you used when generating the certificate.
- *path_to_certificate_file*: Specifies path to where you stored the certificate file.
- *glassfish_truststore_file*: Specifies the TrustStore file used by your client.

Creating a CA-signed Certificate Request

To generate a CA certificate request using the `keytool` command:

```
keytool -certreq -alias <alias> -keystore <path_to_keystore_file> -file
<request_file>
```

where:

- *alias*: Specifies the alias that you gave to the self-signed certificate when you created it.
- *path_to_keystore_file*: Specifies path to the KeyStore file that holds your self-signed certificate.
- *request_file*: Specifies path to the request file output. This file is sent to the CA.

Submit the certificate request and get the certificate approved by the CA. As there are many ways to have your certificate request approved, this step is left up to you.

Importing a CA-signed Certificate

When you receive the certificate from the CA it can be imported into your KeyStore on the server.

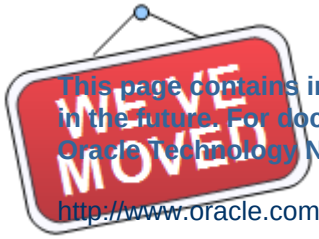
```
keytool -import -trustcacerts -alias <alias> -file <certificate_file>
-keystore <keystore_file>
```

where:

- *alias*: Specifies the alias that you gave to the self-signed certificate when you created it.
- *certificate_file*: Specifies the path to the CA-signed certificate file.
- *keystore_file*: Specifies the path to the server KeyStore.

Chapter 27. Managing Domain Access Controls

Managing Domain Access Controls in Calendar Server



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

The following feature documented in this section was introduced in **Calendar Server 7 Update 2.**

Domain Access Control Lists (ACLs) control calendar operations that span multiple domains. Oracle Communications Calendar Server 7 combines domain ACLs with the calendar and scheduling ACLs to grant or deny levels of access to these operations. All operations within a single domain rely strictly on the calendar and scheduling ACLs.

Topics:

Domain ACLs Overview

In Calendar Server, domain ACLs act as a gateway from one domain to another. When a user in one domain attempts to access a calendar in another domain, Calendar Server first checks the target domain's ACL. If that ACL allows access to the target domain then Calendar Server checks the calendar ACL. The domain ACL itself can also restrict access to the target calendar. For example, if the domain ACL only allows "read" access but the calendar ACL allows "write" access, the request is limited to "read" only.

Calendar domain ACLs are in the same form as WCAP calendar and scheduling ACLs. They consist of one or more Access Control Entries (ACEs) separated by semi-colons. The ACE is made up of a "who" part and a "privilege level" part in the form of `<who>:<privilege>`. A "who" can be a domain name in the form of *@domain* or `@` to designate the privilege level for `all`. The privilege level consists of a single letter.

In general, a domain-level ACL and a user-level ACL behave the same, as they both perform implicit denies and give precedence to more specific ACLs. The only difference occurs when the value is blank. If the domain-level ACL is blank, the server is allowed to continue its check of the user-level ACL. If the

user-level ACL is blank, access is explicitly denied.

Authenticated Access

The two LDAP attributes used with domain ACLs are `icsDomainNames` and `icsDomainAcl`. Each domain can have zero or more `icsDomainNames`, which indicate the other domains that are known to this domain. The `icsDomainAcl` attribute, if it exists, contains the ACL for the domain.

icsDomainNames Attribute Overview

Each `icsDomainNames` attribute consists of the name of a domain, such `example.com`. This attribute indicates what other domains a particular domain is aware of. It is used for search operations across domains.

The search operation gets a list of target domains from the `icsDomainNames` attribute of the requester's domain. The operation first searches the requester's own domain then the domains in the list. As each domain in the list of target domains is processed, the `icsDomainAcl` of each domain and each calendar's ACL is checked, as described in the next section.

For a search in each of the domains specified in `icsDomainNames` to succeed, the corresponding domain nodes should have the correct LDAP ACIs and `icsDomainAcl` setting allowing search from this domain. See Adding LDAP Access Control for Calendar Server Features for more information on setting LDAP ACIs for domains. See the next section for more information on `icsDomainAcl` settings.

icsDomainAcl Attribute Overview

If the target domain does not have an `icsDomainAcl` attribute, the operation continues and the following checks are done by using the calendar and scheduling ACL to see if the operation can be completed.

1. If the `icsDomainAcl` attribute exists, the ACL is checked for the requesting user's domain. If the ACL has an ACE that contains the requesting user's domain, the privileges given to that domain in the ACE are returned and used to restrict what is available from the calendar and scheduling privileges. If the ACL contains both the domain ACE and the @ ACE, the domain ACE is always used.
2. If the `icsDomainAcl` attribute exists and does not contain an ACE for the requesting user's domain but it does contain the @ ACE, the privilege level of the @ ACE is used.
3. If the `icsDomainAcl` attribute exists but does not contain an ACE for the requesting user's domain and also does not contain the @ ACE, access is denied for that user.

The following table provides more detail on `icsDomainAcl` conditions and results. The Domain Privileges Returned column describes the rights that are available at the domain level and how these are used to restrict the rights allowed at the calendar and scheduling level.

icsDomainAcl Conditions and Results

	icsDomainAcl	Domain in ACL	Privilege Level Versus Requested Level	Domain Privileges Returned
U1	Null or empty			All
U2	Exists	Yes	Domain privilege equal to or higher	Based on level in domain ACE
U3	Exists	Yes	Domain privilege lower	None
U4	Exists	Yes plus @	Domain privilege lower and @ privilege is higher	None (Always use domain ACE)
U5	Exists	No and no @		None
U6	Exists	No but @ exists	@ privilege equal to or higher	Based on level in @ ACE
U7	Exists	No but @ exists	@ privilege lower	None

Authenticated Access Examples

This section provides examples that show different levels of cross-domain calendar access.

Example 1

This example enables `userA` in domain `a.com` to be able to read `userB`'s calendar in `domain b.com`.

Domain a.com	Domain b.com
LDAP entry = `icsDomainNames: b.com`	No `icsDomainAcl` defined
	`userB`'s calendar has `acl=userA@a.com:r`

Example 2

This example enables `userA@a.com` to write to the calendar for `userB@b.com`.

Domain a.com	Domain b.com
LDAP entry = `icsDomainNames: b.com`	LDAP entry = `icsDomainAcl: @a.com:w`
	`userB`'s calendar has `acl=userA@a.com:w`

Example 3

This example blocks `userA@a.com` from writing to `userB@b.com`'s calendar.

Domain a.com	Domain b.com
LDAP entry = `icsDomainNames: b.com`	LDAP entry = `icsDomainAcl: @a.com:r`
	`userB`'s calendar has `acl=userA@a.com:w`

Anonymous Access Overview

If there is no `icsDomainAcl` attribute for the target domain, all rights are returned and further checks are then done by using the calendar and scheduling ACLs.

If the target domain has an `icsDomainAcl`, but that ACL does not contain the @ ACE, access is denied.

If the target domain has an `icsDomainAcl` and the ACL contains the @ ACE and the privilege level of that ACE is higher than the requested privilege, the ACE's privilege level is returned and processing continues with the calendar and scheduling ACLs.

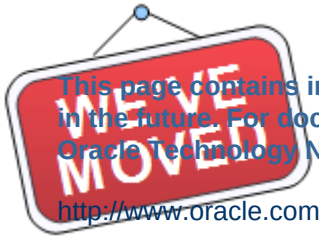
The following table provides more detail on anonymous access conditions and results. The Domain Privileges Returned column describes the rights that are available at the domain level and how these are used to restrict the rights allowed at the calendar and scheduling level.

Anonymous Conditions and Results

	icsDomainAcl	@ in icsDomainAcl	Privilege Level Versus Requested Level	Domain Privileges Returned
A1	Null or empty			All
A2	Exists	No		None
A3	Exists	Yes	@ privilege equal to or higher	Based on level in @ ACE
A4	Exists	Yes	@ privilege lower	None

Chapter 28. Calendar Server Unique Identifier

Calendar Server Unique Identifier



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

Topics:

How Does Calendar Server Use a Unique Identifier?

Calendar Server requires a unique identifier in the form of an LDAP attribute whose value is used to map each calendar entry (in the LDAP Directory Server) to a unique account in the calendar database (the MySQL Server or Oracle Database that stores Calendar Server data). 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.

When planning to deploy Calendar Server, and before installing and configuring Calendar Server, decide on the LDAP attribute to be used as the unique identifier. This is a critical decision. Realistically, you cannot change the attribute you use as the unique identifier once you deploy Calendar Server and start using it. Starting with version 7 Update 3, Calendar Server provides the `davUniqueId` attribute, which is the recommended attribute to use.

For more information, see `davUniqueId` in the Communications Suite Schema Reference.

How Do You Set the Unique Identifier?

You set the attribute to be used as the unique identifier during the Calendar Server initial configuration phase. (When installing a Communications Suite component such as Calendar Server, you first install the software and then configure it. These are two separate steps.) The Calendar Server configurator program, `init-config`, prompts you to enter the attribute you have chosen as your unique identifier. The configurator then stores the value to the `davcore.uriinfo.permanentuniqueid` Calendar Server configuration parameter.

Ever since the release of Calendar Server 7, the LDAP operational attribute `nsUniqueId` has been chosen as the default value used for the unique identifier. However, it was discovered that this choice has a potential serious downside. The problem with using `nsUniqueId` is that if the LDAP entry for a

user, group, or resource is deleted and recreated in LDAP, the new entry would receive a different `nsUniqueId` value from the Directory Server, causing a disconnect from the existing account in the calendar database. As a result, recreated users cannot access their existing calendars.

The Calendar Server 7.0.4.15.0 documentation explains this situation in more detail. See Changing User uuid.

To prevent this potentially serious issue, Calendar Server 7 Update 3 introduced a new attribute, `davUniqueId`, in the `davEntity` objectclass, to use as the unique identifier. You should provision this attribute for your users, groups, and resources LDAP entries with globally unique IDs. To use Delegated Administrator as your provisioning tool, make sure that you are running at least Delegated Administrator 7 Patch 6, which supports provisioning the `davUniqueId` attribute.

In the Calendar Server data base, the unique identifier value is case sensitive. If you need to move or recreate the corresponding LDAP entry, make sure to retain the case of the value as is. However, because the value is considered as case insensitive for LDAP comparisons, do not create a unique identifier value for another user or resource entry by just changing the case of the value.

How Do You Configure Calendar Server to Use the davUniqueId Attribute?

Installing Calendar Server 7.0.4.15.0 for the First Time

For a fresh installation:

1. During initial configuration, when prompted by the Calendar Server `init-config` configurator, choose the default value, `davUniqueId`.
   This is stored to the `davcore.uriinfo.permanentuniqueid` configuration parameter. For more information, see `davUniqueId` in the Communications Suite Schema Reference.
2. In LDAP, populate calendar users, groups, and resources with the `davUniqueId` attribute. You can:
   a. Use the `populate-davuniqueid` tool (introduced in Calendar Server 7 Update 3).
   b. Use your own LDAP tools.

Upgrading to Calendar Server 7 Update 3

If you initially deployed Calendar Server 7 Update 2 to use the `nsUniqueId` attribute as your unique identifier, you should switch to using the `davUniqueId` attribute, new in Calendar Server 7 Update 3. For more information on the problems with using `nsUniqueId`, see Changing User uuid.

To upgrade to Calendar Server 7.0.4.15.0, see Calendar Server 7.0.4.15.0 Upgrade.

Chapter 29. Calendar Server 7 Command-Line Utilities

Calendar Server 7 Command-Line Utilities

This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

Topics:

Overview of the Command-Line Utilities

Calendar Server provides a number of command-line utilities for administering the server. These utilities run under the umbrella command, `davadmin`, which is a simple shell script. The `davadmin` utility is installed in the *cal-svr-base*`/sbin` directory with user or group `bin/bin` permissions.

> **Note**
> The `davadmin` command-line utilities administer aspects of the server and do not affect any LDAP entries.

davadmin Security Enhancements

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 passfile Operation for more information.

Syntax

The syntax of the `davadmin` command is the following:

```
davadmin <operation> <action> <options>
```

You can abbreviate the *operation* and *action* command-line parameters as long as they are unique to the other operations and actions. For example, for the command `davadmin calendar list`, you could use `davadmin cale l`. In this example, you must use at least the first four letters of calendar because "cal" conflicts with *calcomponent*.

The default action is `list`, for most commands, if the action is not specified on the command line. For example, `davadmin account -a` *user* lists the account information for *user*.

davadmin Class of Operations

Argument	Description

`version`	Prints 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` This argument is available starting in **Calendar Server 7 Update 2**.	Adds information for an additional back-end calendar store.
`cache` This argument is valid starting in **Calendar Server 7 Update 3 (Patch 8)**.	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` The `calcomponent` argument replaces the `calresource` argument starting in **Calendar Server 7 Update 1**.	Performs resource operations, such as listing resources that meet a specified criteria, importing resources, or deleting resources.
`calresource` This argument is valid for **Calendar Server 7**.	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.
`db`	Performs database related operations, like backing up and restoring the database.
`ldappool` This argument is available starting in **Calendar Server 7 Update 3**.	Performs `ldappool` operations, including creating, listing, and modifying LDAP pools.
`migration` This argument is available starting in **Calendar Server 7 Update 1**.	Performs migration of Calendar Server 6 data to Calendar Server Server 7 Update 1 and greater.
`passfile` This argument is available starting in **Calendar Server 7 Update 2 Patch 5**.	Creates, deletes, lists (as of Calendar Server 7.0.4.14.0), or modifies passwords in the `password` file.
`vscan` This argument is available starting in **Calendar Server 7 Update 2**.	Performs virus scanning operations.

Each operation supports a list of additional parameters to pass information to the server. Some common options used by all operations (except `davadmin db` operations) are described in the Common Options table.

Note
Any option that contains special characters needs to be enclosed in quotes ("") so that they are passed "as is" to the `davadmin` command.

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"
```

Common Options

The following table describes the options that are common to the `davadmin` operations.

Common Options

Short Option	Long Option	Description	Required or Optional
`-u` *adminuserid*	`--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` This option is available starting in **Calendar Server 7 Update 2 Patch 5**.	`--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` *file*	`--clifile`	File with bootstrap information that you use to specify command-line options so that they don't have to be entered at the command line. Each line in the bootstrap file is in the form *property*=*value*. Note that 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 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` *host*	`--hostname`	Server host name.	Optional. Defaults to `localhost`.
`-p` *port*	`--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` *path*	`--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 can have its own specific options, as the following sections show.

Clifile Properties

The following table describes the possible properties in the bootstrap file (clifile):

Clifile Properties

Property	Description
`userid`	GlassFish Administrator user ID.
`usepasswordfile` This option is available starting in **Calendar Server 7 Update 2 Patch 5**.	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` The `dbhost` property replaces the `dbhostname` property starting in **Calendar Server 7 Update 2**.	Host name where the database server resides.
`dbhostname` This argument is valid for **Calendar Server 7** and **Calendar Server 7 Update 1**.	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 *host*`:`*port* 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 *host*`:`*port*.

Options Precedence

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 `root` user can use the `davadmin.properties` file. The `davadmin.properties` file is installed in the `config` directory of the installation (default location is *cal-srv-base*`/config`).

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 `root` 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 usually contains options for `userid`, `hostname`, `port`, `secure`, `dbhost`, `dbport`, and `dbuserid`.

version Example

- To display the current Calendar Server version (starting with Calendar Server 7 Update 2, you are prompted to enter the administrative password):

```
davadmin version
Enter Admin password: <password>
Oracle Communications Calendar Server version: 7u2-4.19 (built
2011-09-09T10:25:10-0700)
```

account Operation

The `davadmin account` command can be followed by one of the following actions.

Actions for account Operation

Command	Description
`create` This command is available starting in **Calendar Server 7 Update 2**.	Creates an account for user who has been provisioned in the LDAP Directory Server. The user must have an email address.

`delcomponents` This command is available starting in **Calendar Server 7 Update 2**.	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. Starting with **Calendar Server 7 Update 2 Patch 5**, `list` 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. Starting with **Calendar Server 7 Update 3**, 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`, and `-B`, and the new option in Calendar Server 7 Update 3, `-v`.
`modify`	Modifies an account.
`repair` This action is available starting in **Calendar Server 7 Update 2**.	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` This action is available starting in **Calendar Server 7.0.4.14.0**.	Subscribe to a calendar belonging to another user. That other user must grant the requesting user access before this can be done.

`unsubscribe` This action is available starting in **Calendar Server 7.0.4.14.0**.	Remove a calendar from a user's subscription list.

The `account` operation is followed by these parameters.

Options for account Operation

Short Option	Long Option	Description
`-a` *account*	`--account`	Required. Principal account information provided as email address. You can also supply the account information with the `DAVADMIN_ACCOUNT` environment variable.

`-y` *property*	`--property`	Comma-separated list of all *property=value* options for the specified calendar. Possible properties include: `acl` - The scheduling privileges set on the account. For more information about ACLs, see Administering Calendar Server Access. `set-ace` (available starting in **Calendar Server 7.0.4.14.0**) - Sets one or more individual ACEs in the ACL. semi-colon separated list of ACEs. `remove-ace` (available starting in **Calendar Server 7.0.4.14.0**) - Removes one or more individual ACEs from the ACL. Semi-colon 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. Starting with **Calendar Server 7 Update 2 Patch 5**, `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 *user*}`:`*properties*, where *properties* 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)`
`-r`	`--force`	Force the operation (do not prompt for confirmation).
`-h`	`--help`	Displays `davadmin account` usage help.

`-c` This option is available starting in **Calendar Server 7.0.4.14.0**.	`--collectionuri`	The full path of a collection to be added to a user's subscription list. For example, `/home/user@example.com/mycal1/`. Be sure to include the `/` at the end of the path.
`-C` This option is available starting in **Calendar Server 7.0.4.14.0**.	`--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: `/home/user2@example.com/cal1/`

The `delete` operation is followed by these parameters.

Options for delete Operation

Short Option	Long Option	Description
`-a` *account*	`--account`	Required. Principal account information provided as email address. You can also supply the account information with the `DAVADMIN_ACCOUNT` 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` *uniqueID*	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 `davadmin account delete` command fails due to User Not Found, you can delete the user's database data by specifying `-g` *uniqueID*, where *uniqueId* is the user's old LDAP uniqueID. Only use `-g` when users are no longer defined in LDAP.

The `repair` operation is followed by these parameters.

Options for repair Operation

Short Option	Long Option	Description	Default?
`-m` This option is available starting in **Calendar Server 7 Update 2 Patch 5**.	`--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` This option is available starting in **Calendar Server 7 Update 2 Patch 5**.	`--ownerlists`	Updates the owner lists of all accounts. Valid only for the `repair` action.	No

The `davadmin account list` command has the following options.

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.
`-B` *uri*	`--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` This argument is available starting in **Calendar Server 7 Update 3.**	`--verbose`	Detailed information is displayed about each of the users in the database.	Used with the `-f` and `-B` options.

account Examples

- To list the account for a user:

```
davadmin account list -a john.smith@example.com
```

- 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 To Remove 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 (Calendar Server 7 Update 2 Patch 5 and greater):

```
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 (starting with Calendar

Server 7.0.4.14.0):
Input File:

```
user1@example.com:notifemail=0,attendanceflag=5
user2@example.com:notifemail=1,notifrecipients=user4@example.com;user
```

Command:

```
davadmin account create -f input_file.txt
```

- To modify the previous two accounts and set their properties by using an input file (starting with Calendar Server 7.0.4.14.0):

```
davadmin account modify -f input_file.txt
```

backend Operation

> This argument is available starting in **Calendar Server 7 Update 2**.

The following table describes the possible actions for the `backend` operation.

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` This argument is available starting in **Calendar Server 7 Update 3**.	Immediately purges calendar data marked for expiration from Calendar Server back-end database(s).

The `backend` operation is followed by these parameters.

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.

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. For more information, see 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
```

cache Operation

The `davadmin cache` command can be followed by the following action.

Action for cache Operation

Command	Description
`clear` This command is available starting in **Calendar Server 7 Update 3 (Patch 8)**.	Clears the various Calendar Server caches.

The `cache` operation is followed by this parameter.

Option for cache Operation

Short Option	Long Option	Description	Required or Optional
`-t`	`--cachelist`	Comma-separated list of caches, possible values are: • `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`.	Optional.

calendar Operation

The `davadmin calendar` command can be followed by one of the following actions.

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.

The `calendar` operation is followed by these parameters.

Options for calendar Operation

Short Option	Long Option	Description
`-a` *account*	`--account`	Required. Principal account information provided as email address. You can also supply the account information with the `DAVADMIN_ACCOUNT` environment variable.
`-n` *collection*	`--name`	The unique calendar collection name that corresponds to the last part of the calendar collection URI. For the default calendar, *collection* it is always `calendar`. This name might not correspond with the calendar's display name at all times. 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` *property*	`--property`	Comma-separated list of all *property=value* options for the specified calendar. Possible properties include: `set-ace` (available starting in **Calendar Server 7.0.4.14.0**) - Specifies a semi-colon separated list of ACEs to add or modify to the calendar permissions (ACL). `remove-ace` (available starting in **Calendar Server 7.0.4.14.0**) - Specifies a semi-colon 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. `displayname` - The calendar name. Defaults to the name given with the `-n` option. `calendar-description` - Description string. No default. `supported-calendar-component-set` - Space-separated list of supported components. The default is `VEVENT VTODO VFREEBUSY`. This option is only available for creation of secondary calendars. It cannot be used for creation of the default calendar. The following properties are available starting in **Calendar Server 7 Update 1**. `wcaptzid` - The timezone tzid set on the calendar, for example, `America/Los_Angeles`. `acl` - The access control string set on the calendar. For more information about ACLs, see Administering Calendar Server Access.
`-f` *file*	`--file`	Local commands input file for batch operation. Each line has colon-separated entries for account information, calendar name, and property list. For example: user1@example.com:testcal:calendar-description=user1's test cal

`-v` This option is available starting in Calendar Server **7.0.4.16.0.**	`--verbose`	Used with the `list -a` *user* command to display a summary for all calendars belonging to the user.
`-h`	`--help`	Displays `davadmin calendar` usage help.

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"
```

- The following example shows `caluser30` with two calendars, a default calendar and `newcal20141006203235`. The calendar `newcal20141006203235` is shared with `caluser31`, `caluser32`, and `caluser33` with the different rights:

```
# ./davadmin calendar list -a caluser30@example.com -v

Calendar Account Name: caluser30@example.com
Calendar Name: calendar
Calendar Display Name: caluser30
Calendar Owner: [/dav/principals/caluser30@example.com/]
Calendar Creation Date: Tue Oct 07 03:32:36 UTC 2014
Calendar Last Modification Date: Tue Oct 07 03:32:36 UTC 2014
Supported Components: [VEVENT, VTODO, VFREEBUSY]
ACL:
Number of events: 0
Number of tasks: 0

Calendar Account Name: caluser30@example.com
Calendar Name: newcal20141006203235
Calendar Display Name: newcal20141006203235
Calendar Owner: [/dav/principals/caluser30@example.com/]
Calendar Creation Date: Tue Oct 07 03:32:38 UTC 2014
Calendar Last Modification Date: Tue Oct 07 03:32:38 UTC 2014
Supported Components: [VEVENT, VTODO, VFREEBUSY]
ACL:
caluser31@sun.com:a;caluser32@sun.com:w;caluser33@example.com:r
Number of events: 0
Number of tasks: 0
```

calcomponent Operation

> The `calcomponent` argument replaces the `calresource` argument starting in **Calendar Server 7 Update 1**.

The `davadmin calcomponent` command can be followed by one of the following actions.

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.

The `calcomponent` operation is followed by these parameters.

Options for calcomponent Operation

Short Option	Long Option	Description
`-a` *account*	`--account`	Required. Principal account information provided as email address. You can also supply the account information with the `DAVADMIN_ACCOUNT` environment variable.
`-n` *collection*	`--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` *property*	`--property`	Comma-separated list of all *property=value* 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 *yyyymmdd*`T`*hhmmss*`Z`. 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 *yyyymmdd*`T`*hhmmss*`Z`. 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.
`-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. Starting with **Calendar Server 7 Update 2 Patch 5**, the `davadmin calcomponent import` command enables the import to continue even if an error occurs on an item being imported.

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=VTODO
```

- 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
```

config Operation

The `davadmin config` command can be followed by one of the following actions.

Actions for config Operation

Command	Description
`list` This argument is available starting in **Calendar Server 7 Update 2**.	Lists all configuration settings. This is the default action if not included on the command line.
`modify` This argument is available starting in **Calendar Server 7 Update 2**.	Modifies a configuration setting.

At least one of the following options can be provided, unless you are displaying usage by using the `-h` option. See Calendar Server 7 Configuration Parameters for the complete list of configuration parameters.

Options for config Operation

Short Option	Long Option	Description
`-o` *option*	`--option`	Configuration option name. Gets the optional value if specified without `-v`. Sets the option value if specified with a `-v`.
`-v` *value*	`--value`	Configuration option value.
`-f` *file*	`--file`	Local file with list of configuration `option`=*value* 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 "`^/principals/\z`", you would use "`^/principals/\\z`". This is due to the way that Java reads in properties files. For more information, see the `load(Reader reader)` method of the `java.util.Properties` class at http://docs.oracle.com/javase/6/docs/api/index.html.
`-M` This argument is available starting in **Calendar Server 7.0.4.14.0**.	--modonly	Lists the modified configuration properties (non-default values).
`-d` This argument is available starting in **Calendar Server 7.0.4.14.0**.	`--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`.

config Examples

- To show all configuration parameters (starting with Calendar Server 7 Update 2):

```
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 modify -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 (starting with Calendar Server 7.0.4.14.0):

```
davadmin config list -o davcore.acl.defaultschedulingacl -d -u
admin
Enter Admin password:
davcore.acl.defaultschedulingacl: @:s
```

- To modify to the default setting (starting with Calendar Server 7.0.4.14.0):

```
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
```

db Operation

The `davadmin db` command can be followed by one of the following actions.

Actions for db Operation

Command	Description
`backup`	Backs up a database.
`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.

Starting with **Calendar Server 7.0.4.14.0**, 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.

The `db` operation can be followed by these parameters in addition to the common parameters:

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`.	`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
`-u`	`--dbuser`	Specifies the database user.	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`	List the resources and header.	`list`

`-A` This option is available starting in **Calendar Server 7 Update 1**.	`--docstore`	Specifies the document store (remote store specified as host:port or local store by fully qualified path to root of document store).	`backup`, `restore`, `list`
`-t` This option is available starting in **Calendar Server 7 Update 2**.	`--dbtype`	Specifies the type of database, either `mysql` or `oracle`. The default is `mysql`.	All
`-O`	`--overwrite`	Overwrites existing data.	`backup`, `restore`
`-s` This option is available starting in **Calendar Server 7.0.4.14.0**.	`--dbsecure`	Supplies the path to the `trustStore` file that contains the SSL certificate for secure communications with the remote document store.	`backup`
`-e`	`--detail`	Lists each resource as it is either saved or considered for restore.	`backup`, `restore`

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 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 (starting with Calendar Server 7.0.4.14.0):

```
davadmin db backup -k /tmp/backup_file -O -A
docstore_host.example.com:8008 -s /my_home/my_truststore -u mysql
```

ldappool Operation

> This argument is available starting in **Calendar Server 7 Update 3**.

The `davadmin ldappool` command can be followed by one of the following actions.

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.)

The `ldappool` operation is followed by these parameters.

Options for ldappool Operation

Short Option	Long Option	Description
`-n` *poolname*	`--name`	The name of the LDAP pool.
`-y` *property*	`--property`	Comma-separated list of all *property*=*value* 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` *file*	`--file`	Local input file with one line for each account, for batch operation, containing lines in the form *pool_name*`:`*property_list*. The properties are the same ones available for the `-y` option. For `delete` operations, only *pool_name* is used.
`-r`	`--force`	Force the operation (do not prompt for confirmation).
`-h`	`--help`	Displays davadmin ldappool usage help.

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
```

migration Operation

> The following feature documented in this section is available starting in **Calendar Server 7 Update 1**.

For more information on migrating from Sun Java System Calendar Server 6 to Oracle Calendar Server 7, see Migrating From Sun Java System Calendar Server 6 to Calendar Server 7.

The `davadmin migration` command can be followed by by one of the following actions.

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`. In addition, the `migration` option supports the following options:

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 *server*`:`*port*.	Required unless information is provided in `clifile`.
`-l` *log-directory*	`--logpath`	Logs information about the migration status.	Optional. Defaults to the Calendar Server 7 `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 *yyyymmdd*`T`*hhmmss*`Z`. 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 Migration Logging.

The `clifile` that is provided through the `-F` option can be used to provide entries for `migrationadminuser`, `migrationadminpassword` (prior to Calendar Server 7 Update 2), and `migrationserverport`. Note that the long option for `-x` is `--migrationadminpasswordpath` (prior to Calendar Server 7 Update 2), a path to the password file, but the entry in the `clifile` is `migrationadminpassword`, because it is just a password.

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 (objectclass):

```
davadmin migration migrate -B "ou=people,o=example.com,o=isp" -R
"objectclass=icscalendaruser" -X calmaster -L
cs6host.example.com:8080 -u admin
```

passfile Operation

This argument is available starting in **Calendar Server 7 Update 2 Patch 5**.

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. The `davadmin passfile` operation is used to create, delete, or modify this `password` file.

The `davadmin passfile` command can be followed by by one of the following actions. The default action is `list`.

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 `passfile` command supports the following option.

Option for passfile Operation

Option	Description	Available for Following Actions
`-O` This option is available starting in **Calendar Server 7 Update 3.**	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.	`create`, `delete`, `list`, `modify`

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:
```

vscan Operation

> This argument is available starting in **Calendar Server 7 Update 2**.

The `davadmin vscan` command takes one action, `scan`, which is the default.

Actions for vscan Operation

Action	Description
`scan`	Scans calendar data for viruses.

Options for vscan Operation

Short Option	Long Option	Description
`-u` *adminuserid*	`--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` *file*	--clifile	File with bootstrap information that you use to specify command-line options so that they don't have to be entered at the command line. Each line in the bootstrap file is in the form `property=value`. 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 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` *host*	`--hostname`	Host name of the server. Optional, defaults to localhost.
`-p` *port*	`--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` *path*	`--secure`	Path to the truststore file used for a secure connection (HTTPS). Optional. Required if GlassFish is running in secure mode.
`-a` *account*	`--account`	The account information (email address) of the user to be scanned.
`-n` *name*	`--name`	The name of the target backendID.
`-B` *uri*	`--ldapbaseuri`	Base URI in LDAP.
`-R` *filter*	`--ldapfilter`	User search filter in LDAP. Default is `(objectClass=icsCalendarUser)`.
`-T` *time*	`--starttime`	Scan data entered into the server after this time. Format: *yyyymmddThhmmssZ*"
`-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 varaiables.
`-h`	`--help`	Help for that particular operation. Optional.

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 attachements 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 in the case of 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. Note that 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.

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.

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.

Notes

If GlassFish 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

- 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` 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. In previous releases, 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.

Deprecated Options

The following tables show the deprecated `davadmin` options and in what release the option was deprecated.

Deprecated Common Option

Short Option	Long Option	Description	Required or Optional
`-W` *passfile* 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.

Deprecated Clifile Properties

Property	Description
`userid`	GlassFish Administrator user ID.
`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.

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.

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`.

Chapter 30. Calendar Server 7 Configuration Parameters

Calendar Server 7 Configuration Parameters

This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

The following table lists the configuration parameters and descriptions for Calendar Server 7. Where applicable, this table shows in which release a particular parameter was made available. See the `config` command for information on how to update or change configuration parameters.

> **Note**
> The information on this page is generated automatically. Any changes made to this page will be overwritten the next time the information is updated. If you would like to request a change to this information, use the Comments feature. All comments will be considered.

Calendar Server 7 Configuration Parameters

Parameter	Description
base.ldapinfo.cachesize	Size of the LDAP Authentication cache. Syntax: integer Minimum: 1 Maximum: 1000000 Default: 1000
base.ldapinfo.cachettl	Time to live (in seconds) of cached LDAP Authentication info. Syntax: integer Unit: seconds Minimum: 1 Maximum: Maximum int value Default: 60
base.ldapinfo.dcroot	Root of DC tree (Schema 1) or of the domain and users tree (Schema 2) in Directory Server Syntax: string Default: o=isp

base.ldapinfo.defaultdomain	Default domain Syntax: string Default: demo.example.com
base.ldapinfo.domainattrs	Space separated list of LDAP attributes to use when retrieving domain information. Syntax: string Default: icsStatus icsDomainNames icsDomainAcl externalAuthPreUrlTemplate externalAuthPostUrlTemplate corpDirectoryUrl
base.ldapinfo.loginseparator	Character(s) to be used as login separator (between userid and domain) Syntax: string Default: @
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 Communications Suite Schema Reference). Syntax: string Default: (uid=%U)
base.ldapinfo.serviceadmindn	DN of single admin in LDAP in absence of admin group. Syntax: string Default:
base.ldapinfo.serviceadminsgroupdn	DN of service admins group in LDAP Syntax: string Default:
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:
base.ldapinfo.authldap.bindpassword	Password to use when authenticating. Syntax: password Default:
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 reverts back to the primary directory server. If set to -1, don't refresh. Syntax: integer Unit: minutes Minimum: 1 Maximum: 60 Default: 1
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.ugldap.binddn	Distinguished Name to use when authenticating. Syntax: string Default:
base.ldapinfo.ugldap.bindpassword	Password to use when authenticating. Syntax: password Default:
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 reverts back to the primary directory server. If set to -1, don't refresh. Syntax: integer Unit: minutes Minimum: 1 Maximum: 60 Default: 1

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 which 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.ldapusessl	Use SSL to connect to the LDAP host. Syntax: boolean Default: false
base.ldappool.*.binddn	Distinguished Name to use when authenticating. Syntax: string Default:
base.ldappool.*.bindpassword	Password to use when authenticating. Syntax: password Default:
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 reverts back to the primary directory server. If set to -1, don't refresh. Syntax: integer Unit: minutes Minimum: 1 Maximum: 60 Default: 1
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 which 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
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 parameter is 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	Whether Apple Private/Confidential Events extension support is on or off. Syntax: boolean Default: false
davcore.acl.calendaranonymousall	If true, map '@'(all) in calendar acls to authenticated and unathenticated 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:
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 unathenticated users. If false, map '@'(all) in acl to just authenticated users. Syntax: boolean Default: true
davcore.attachment.enable	Enable/Disable Attachments. Syntax: boolean Default: true
davcore.auth.cert.enable	Enable Certificate-based client authentication Syntax: boolean Default: false
davcore.auth.cert.fallback	Fallback to username and password authentication Syntax: boolean Default: true
davcore.autocreate.calattendantresourceflags	Calendar attendant flags to set on resource scheduling inbox. Default setting used on autocreation. This value can be altered by the presence of the icsAutoAccept and icsDoubleBooking attributes. The value is a bitmask of the following: 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. Syntax: long Minimum: 0 Maximum: 7 Default: 3

davcore.autocreate.calattendantuserflags	Calendar attendant flags to set on users scheduling inbox. Default setting used on autocreation. This value can be altered by the presence of the icsAutoAccept and icsDoubleBooking attributes. The value is a bitmask of the following: 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. Syntax: long Minimum: 0 Maximum: 7 Default: 0
davcore.autocreate.calcomponents	Supported calendar components set for a new user calendar on autocreation. Default setting used on autocreation. Syntax: string Default: VEVENT VTODO 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 backend 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:
davcore.homeuri.*.rank	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. Syntax: integer Minimum: 0 Maximum: Maximum int value Default: 1
davcore.homeuri.*.subjectdomain	Once it is determined that a uri matches the pattern, this domain template is used to identify the subject owning with this resource. The template can reference the $1, $2,... 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". Can be empty indicating the default domain. Syntax: string Default: $2
davcore.homeuri.*.subjectfilter	Once it is determined that a uri matches the pattern, this LDAP filter template is used to identify the subject owning with this resource. The template can reference the $1, $2,... 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 will become "(mail=john@example.com)". Can be empty, indicating that this namespace is not associated with a particular subject. Syntax: string Default: (mail=$1@$2)

davcore.homeuri.*.uripattern	Regex pattern to be matched by the uri. This pattern can contain regex groups (identified by () parenthesis) which will be saved into $1, $2,... The last regex group identifies the local path if there is any. 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". Syntax: string Default: ^/home/([^/]+)@([^/]+)(/\z|/.*)
davcore.ldapattr.commonname	Common name attribute. Syntax: string Default: cn
davcore.ldapattr.corpdirectoryurl	LDAP attribute to locate a custom external corporate directory for this domain. Syntax: string Default: corpDirectoryUrl
davcore.ldapattr.davstore	Logical backend id attribute. Syntax: string Default: davStore
davcore.ldapattr.defaultresourcetype	Default CUTYPE value to use when the resource typ LDAP attribute of a calendar resource is not present. Syntax: string Default: ROOM
davcore.ldapattr.dngroupmember	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 Objectclass values indicating an LDAP group. Syntax: string Default: groupofuniquenames groupofurls inetmailgroup

davcore.ldapattr.icsautoaccept	LDAP attribute to use to determine whether autoaccept should be turned on. The attribute value can be 1 (autoaccept) or 0 (no autoaccept). It is used only during autocreate. Syntax: string Default: icsAutoAccept
davcore.ldapattr.icsdoublebooking	LDAP attribute to use to determine whether doublebooking is allowed. The attribute value can be 1 (double booking allowed) or 0 (no double booking allowed). It is used only during autocreate. 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: mgrprfc822mailmember
davcore.ldapattr.memberattr	LDAP attribute listing the groups the entry is a member of. Syntax: string Default: ismemberof
davcore.ldapattr.preferredlang	Language attribute. Syntax: string Default: preferredLanguage
davcore.ldapattr.resourceobject	Space separated list of Objectclass values indicating an LDAP resource. Syntax: string Default: icsCalendarResource
davcore.ldapattr.resourceowner	LDAP attribute to use to determine the owner of a calendar resource (e.g. conference room). The attribute value must be a DN. It is used only during autocreate. 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: * 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 Objectclass values indicating a calendar user. Syntax: string Default:
davcore.otheruri.*.backendid	Once it is determined that a uri matches the pattern, this backendid template is used to identify the backend 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:
davcore.otheruri.*.rank	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. Syntax: integer Minimum: 0 Maximum: Maximum int value Default: 1

davcore.otheruri.*.subjectdomain	Once it is determined that a uri matches the pattern, this domain template is used to identify the subject owning with this resource. The template can reference the $1, $2,... 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". Can be empty indicating the default domain. Syntax: string Default: $2
davcore.otheruri.*.subjectfilter	Once it is determined that a uri matches the pattern, this LDAP filter template is used to identify the subject owning with this resource. The template can reference the $1, $2,... 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 will become "(mail=john@example.com)". Can be empty, indicating that this namespace is not associated with a particular subject. Syntax: string Default: (mail=$1@$2)
davcore.otheruri.*.uripattern	Regex pattern to be matched by the uri. This pattern can contain regex groups (identified by () parenthesis) which will be saved into $1, $2,... The last regex group identifies the local path if there is any. 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". Syntax: string Default: ^/home/([^/]+)@([^/]+)(/\z|/.*)
davcore.presence.advancepresencetriggerinterval	Specifies the number of seconds in advance to trigger for presence update. That is, how long before event start/end is the publication made. Changes to this value will NOT affect existing event triggers. Syntax: long Unit: seconds Minimum: 0 Maximum: Maximum long value Default: 0
davcore.presence.enable	Enable/Disable Presence info publication. Syntax: boolean Default: true

davcore.principalsuri.*.backendid	Once it is determined that a uri matches the pattern, this backendid template is used to identify the backend 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:
davcore.principalsuri.*.rank	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. Syntax: integer Minimum: 0 Maximum: Maximum int value Default: 1
davcore.principalsuri.*.subjectdomain	Once it is determined that a uri matches the pattern, this domain template is used to identify the subject owning with this resource. The template can reference the $1, $2,... 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". Can be empty indicating the default domain. Syntax: string Default: $2
davcore.principalsuri.*.subjectfilter	Once it is determined that a uri matches the pattern, this LDAP filter template is used to identify the subject owning with this resource. The template can reference the $1, $2,... 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 will become "(mail=john@example.com)". Can be empty, indicating that this namespace is not associated with a particular subject. Syntax: string Default: (mail=$1@$2)

davcore.principalsuri.*.uripattern	Regex pattern to be matched by the uri. This pattern can contain regex groups (identified by () parenthesis) which will be saved into $1, $2,... The last regex group identifies the local path if there is any. 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". Syntax: string Default: ^/home/([^/]+)@([^/]+)(/\z|/.*)
davcore.reverseuri.*.backendid	Backend id on which to apply this reverse mapping. There should be only one mapping per backend. Syntax: string Default:
davcore.reverseuri.*.uritemplate	Canonical form of the URI prefix for this backend. This template should have a corresponding uripattern. It should not end with a slash. 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. If no template is defined for a given backend, the uriinfo.defaulthomeuritemplate config parameter is used. Syntax: string Default:
davcore.scheduling.allowownerdoublebooking	If set, owners of resource calendars can doublebook even if the resource account prevents doublebooking. Syntax: boolean Default: false
davcore.scheduling.calendarinboxexpirytime	Specifies the number of seconds after which the resources in calendar-inbox will be deleted. Syntax: long Unit: seconds Minimum: 0 Maximum: Maximum long value Default: 2592000
davcore.scheduling.calendaroutboxexpirytime	Specifies the number of seconds after which the resources in calendar-outbox will be deleted. Syntax: long Unit: seconds Minimum: 0 Maximum: Maximum long value Default: 604800

davcore.scheduling.includememberinattendee	If set, scheduling messages would include the MEMBER=group attribute in the ATTENDEE property where group is the LDAP group which this attendee is a member of. The default is set to false in order to accommodate compatibility with Leopard Apple iCal. Syntax: boolean Default: false
davcore.scheduling.ischedulebackendid	iSchedule Backend identifier. If not set, incoming iSchedule requests are disabled. Syntax: string Default:
davcore.scheduling.itiptransmittablexprops	Space separated list of iCalendar X-properties that are transmittable via iTIP. All other X- properties are not transmitted between organizer and attendees. Syntax: string Default: X-S1OC-OWNER-APPT-ID X-S1OC-TZID X-S1OC-OUTLOOK-SENDER-EMAIL X-S1OC-OUTLOOK-SENDER-CN X-S1OC-OUTLOOK-ACCEPTED-BY-NAME X-S1OC-OUTLOOK-EVENT-REPLY-TIME
davcore.scheduling.localuserattr	Local Dav 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). Then, users with a valid value for that attribute in LDAP, are considered local. Others are considered remote. Make sure the same attribute is added to davcore.uriinfo.subjectattributes values. Syntax: string Default:
davcore.scheduling.maxattendeesforrefresh	Above this limit of attendees, attendee's reply are only sent to the Organizer and are no longer propagated to the other attendees. Syntax: long Minimum: 0 Maximum: Maximum long value Default: 50
davcore.scheduling.maxbookingwindow	Specifies the number of days calendars that donnot allow doublebooking can be booked in advance. A valid range is [0, 2G]. Syntax: integer Unit: days Minimum: 1 Maximum: Maximum int value - 1 Default: 365

davcore.scheduling.maxretry	Specifies maximum number of attempts to deliver a scheduling message (e.g. SMTP Server or remote iSchedule is temporary down). Syntax: integer Minimum: 0 Maximum: 999 Default: 24
davcore.scheduling.minbookingwindow	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 min booking window will not be honored during the free busy check. Syntax: integer Unit: days Minimum: -1 Maximum: Maximum int value - 1 Default: -1
davcore.scheduling.rejectinactiverecipients	When set to true, recipients of scheduling messages who have their icsStatus set to inactive will be treated as unknown recipients. Otherwise, those recipients will be treated as iMIP recipients. Syntax: boolean Default: false
davcore.scheduling.retryinterval	Specifies the number of seconds to wait between two attempts to deliver a scheduling message (e.g. SMTP Server or remote iSchedule is temporary down). Syntax: long Unit: seconds Minimum: 0 Maximum: Maximum long value Default: 3600
davcore.scheduling.schedulinglogfilter	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. Syntax: string Default:
davcore.scheduling.synchronousdelivery	If set, scheduling messages (invitations, replies, cancel,...) will be delivered synchronously on submit. This option should not be set under normal circumstances. Syntax: boolean Default: false
davcore.serverdefaults.exportconfigdir	Directory path for export XSL transformation files. Syntax: filepath Default: config/export

davcore.serverdefaults.importconfigdir	Directory path for import properties and translation files. Syntax: filepath Default: config/import
davcore.serverdefaults.jsonprefix	Default prefix to append to all JSON output. Syntax: string Default: {}&&
davcore.serverdefaults.tzid	Default TZID to return when a calendar collection does not have one explicity set. Syntax: string Default:
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 addressbook 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
davcore.serverlimits.maxcontentlength	Maximum size of a resource. Might be overwritten for certain types of content (e.g. text/calendar). Syntax: long Unit: bytes Minimum: 0 Maximum: Maximum long value Default: 10000000

davcore.serverlimits.maxgroupexpansion	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
davcore.serverlimits.maxresults	Maximum number of resources returned by a single fetch operation (WebDAV PROPFIND, CalDAV Reports, WCAP fetch or export, etc...). A value of 0 means no limit. Admins are not affected by this limit. Syntax: integer Minimum: 0 Maximum: Maximum int value Default: 10000
davcore.serverlimits.maxsearchtimerange	Maximum bounded search range in days. Syntax: long Unit: days Minimum: 0 Maximum: 366000 Default: 3660

davcore.serverlimits.maxuploadcontentlength	Maximum size when uploading data. This affects operations that let you create multiple resources in one request (e.g. import). It does not affect regular PUT. Syntax: long Unit: bytes Minimum: 0 Maximum: Maximum long value Default: 20000000
davcore.serverlimits.migrationtimeout	Maximum number of hours to wait before terminating a migration. Syntax: integer Minimum: 1 Maximum: 100 Default: 8
davcore.serverlimits.minsearchcharacters	Minimum number of characters allowed in a text filter search. Syntax: integer Minimum: 0 Maximum: 256 Default: 3
davcore.serverlimits.templockretry	Maximum number of attemps to acquire a temporary lock when doing write operations. Syntax: integer Minimum: 1 Maximum: Maximum int value Default: 20
davcore.serverlimits.templocktimeout	Maximum amount of time to wait for a temporary lock when doing write operations. Syntax: integer Unit: seconds Minimum: 1 Maximum: Maximum int value Default: 60
davcore.serverlimits.templockusebackend	If true, temporary lock are ensured at the backend level instead of staying local to a server instance. Syntax: boolean Default: false
davcore.uriinfo.backendidtemplate	The backendid template is used to identify the backend hosting the home of a given subject. The template can reference LDAP attributes of the subject, using the ${attrname} syntax or the ${attrname,defaultvalue} syntax. Syntax: string Default: ${davStore,defaultbackend}

davcore.uriinfo.defaultdavuriprefix	Canonical form of DAV URI prefix for WebDAV based protocols. This prefix corresponds to one of the the DavServlet specific path (e.g. /dav) as defined in web.xml. It should not end with a slash. Syntax: string Default: /dav
davcore.uriinfo.defaulthomeuritemplate	Canonical form of a subject home URI prefix. This template should have a corresponding uripattern. It should not end with a slash The template can reference LDAP attributes of the subject, using the ${attrname} syntax or the ${attrname,defaultvalue} syntax. The ${domain} syntax can be used to reference the domain of the subject. Syntax: string Default: /home/${mail}
davcore.uriinfo.defaultprincipaluritemplate	Canonical form of a subject principal URI prefix. This template should have a corresponding uripattern. It should not end with a slash. The template can reference LDAP attributes of the subject, using the ${attrname} syntax or the ${attrname,defaultvalue} syntax. The ${domain} syntax can be used to reference the domain of the subject. Syntax: string Default: /principals/${mail}
davcore.uriinfo.defaultresturiprefix	Canonical form of REST URI prefix for WebDAV based protocols. This prefix corresponds to one of the RESTfulServlet specific path as defined in web.xml. It should not end with a slash. Syntax: string Default: /rest
davcore.uriinfo.directoryrootcollection	Defines the root collection of all directory collections. (without any prefix). Syntax: string Default: /directory/
davcore.uriinfo.emailsearchfiltertemplate	LDAP Filter used when searching a subject by email address. The %s token will be replaced by the email value to search. Syntax: string Default: |(mail=%s)(mailalternateaddress=%s)

davcore.uriinfo.fulluriprefix	Full url prefix to use wherever a full url is required. It should not end with a slash. This prefix is used, among other things to construct attachment urls embedded in calendar resources. Modifying this parameter won't change full urls in already existing calendar resources. If ssl is used, the hostname part of this prefix should match the host name associated with the certificate. Syntax: string Default: http://localhost
davcore.uriinfo.ldapcachesize	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 parameter is changed. Can be set to 0, indicating no cache. Syntax: integer Minimum: 0 Maximum: Maximum int value Default: 1000
davcore.uriinfo.ldapcachettl	Maximum time (in seconds) that subjects (LDAP users, resources and groups) are kept in cache when mapping URIs and subjects. Syntax: integer Unit: seconds Minimum: 1 Maximum: Maximum int value Default: 60
davcore.uriinfo.permanentuniqueid	Name of an LDAP attribute present in the LDAP entry of all subjects (users, groups, resources,...) and defining a permanent and unique identifier for each subject. 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). !!! WARNING !!! Changing this configuration parameter will result in data loss once users repository have been created. Syntax: string Default: davuniqueid
davcore.uriinfo.principalsrootcollection	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. Syntax: string Default: /principals/

davcore.uriinfo.subjectattributes	Space separated list of LDAP attribute names to retrieve when doing a search for users, group or resources. Syntax: string Default: cn davstore icsstatus mail mailalternateaddress nsuniqueid owner preferredlanguage uid objectclass ismemberof uniquemember memberurl mgrprfc822mailmember kind
davcore.uriinfo.subjectsearchfilter	LDAP Filter used when a user is searching for other users. The %s token will be replaced by the search string. Syntax: string Default: (|(uid=%s*)(cn=*%s*)(mail=*%s*))
davcore.uriinfo.subjectsearchfilterminimum	The mimimum number of characters allowed for the search string. Syntax: integer Minimum: -2147483648 Maximum: Maximum int value Default: 3
davcore.uriinfo.uricachesize	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 parameter is changed. Can be set to 0, indicating no cache. Syntax: integer Minimum: 0 Maximum: Maximum int value Default: 10000
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.useldapproxyauth	If true, use proxy authorization for any LDAP search on behalf of a user. If false, use administrator credentials for all LDAP searches. Syntax: boolean Default: true
davcore.virusscan.auth	The virus scan connection should use user/passwd authorization (true/false). 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:

davcore.virusscan.debug	The virus scan smtp connection should use debug (true/false). Syntax: boolean Default: false
davcore.virusscan.emailaddress	Email Recipient Address which the MTA is configured to use to trigger a custom virus scan. (Requires mta configuration). Syntax: string Default:
davcore.virusscan.host	Host of the MTA configured to accept virus scans. Syntax: string Default:
davcore.virusscan.onlineenable	Enable/Disable 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:
davcore.virusscan.onlinevirusaction	Action to be performed when a virus is detected during an online submission. Value is empty or reject. Syntax: string Default:
davcore.virusscan.pass	The smtp authorization password for the smtp virus scan connection. Syntax: password Default:
davcore.virusscan.port	Host Port of the MTA configured to accept virus scans. Syntax: string Default: 25
davcore.virusscan.starttls	The virus scan connection should use starttls (true/false). 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 authorization user for the smtp virus scan connection. Syntax: string Default:
davcore.virusscan.usessl	The virus scan connection should use ssl (true/false). Syntax: boolean Default: false

log.dav.commands.logdateformat	Specify the date format pattern for the log. Syntax: logdateformat Default: yyyy-MM-dd'T'HH:mm:ss.SSSZ
log.dav.commands.logdir	Directory path for log files Syntax: filepath Default: logs
log.dav.commands.loglevel	Specify 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 Application Server log file, in addition to the davserver logs. Syntax: boolean Default: false
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	Specify the date format pattern for the log. Syntax: logdateformat Default: yyyy-MM-dd'T'HH:mm:ss.SSSZ
log.dav.errors.logdir	Directory path for log files Syntax: filepath Default: logs
log.dav.errors.loglevel	Specify 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 Application Server log file, in addition to the davserver 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
log.dav.scan.logdateformat	Specify the date format pattern for the log. Syntax: logdateformat Default: yyyy-MM-dd'T'HH:mm:ss.SSSZ
log.dav.scan.logdir	Directory path for log files Syntax: filepath Default: logs
log.dav.scan.loglevel	Specify 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 Application Server log file, in addition to the davserver 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	Specify the date format pattern for the log. Syntax: logdateformat Default: yyyy-MM-dd'T'HH:mm:ss.SSSZ
log.dav.scheduling.logdir	Directory path for log files Syntax: filepath Default: logs

log.dav.scheduling.loglevel	Specify 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.scheduling.logtoparent	Flag to enable logging to the Application Server log file, in addition to the davserver 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	Specify the date format pattern for the log. Syntax: logdateformat Default: yyyy-MM-dd'T'HH:mm:ss.SSSZ
log.dav.telemetry.logdir	Directory path for log files Syntax: filepath Default: logs
log.dav.telemetry.loglevel	Specify 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 Application Server log file, in addition to the davserver logs. Syntax: boolean Default: false
log.dav.telemetry.maxlogfiles	Maximum number of log files Syntax: integer Minimum: 1 Maximum: 100 Default: 10

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	Directory path for notification configuration files or format files Syntax: filepath Default: config/templates
notification.dav.dateformat	Specify the date format pattern for notification. For example, "EEE MMMMM dd, yyyy" Syntax: dateformat Default: EEE MMMMM dd, yyyy
notification.dav.enableemailnotif	Enable server-wide email notification Syntax: boolean Default: true
notification.dav.enableimip	Enable server-wide iMIP scheduling Syntax: boolean Default: true
notification.dav.enableimipemailnotif	Enable server-wide scheduling email notification that contain iMIP data Syntax: boolean Default: false
notification.dav.enablejmsnotif	Enable server-wide JMS notification Syntax: boolean Default: true
notification.dav.maxpayload	Maximum payload size in bytes Syntax: integer Minimum: -2147483648 Maximum: Maximum int value Default: 10000000
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:
notification.dav.smtppassword	SMTP password Syntax: password Default:
notification.dav.smtpport	SMTP port Syntax: string Default: 25

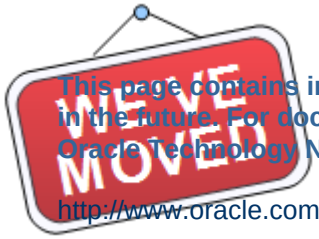
notification.dav.smtpstarttls	SMTP start tls flag Syntax: string Default: true
notification.dav.smtpuser	SMTP user Syntax: string Default: user
notification.dav.smtpusessl	SMTP use SSL flag Syntax: string Default: false
notification.dav.timeformat	Specify the time format pattern for notification. Use 'a' for AM/PM marker. For example, 'hh:mm:ss aaa'. Syntax: timeformat Default: hh:mm:ss aaa
notification.dav.timezoneformat	Specify the timezone format pattern for notification. Use 'z' for general timezone, or 'Z' for RFC822 timezone. Syntax: timezoneformat Default: z
service.dav.blacklist	List of DAV 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 will be denied access. Syntax: string Default:
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 will match 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:
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

service.dav.telemetry.filter	Space separated list of request URI that a particular request should match (start with) to be logged by telemetry (e.g. "/dav/home/jsmith/calendar/ /dav/home/jdoe/calendar/"). Syntax: string Default:
service.dav.telemetry.forcetelemetry	Force telemetry for all users. Warning: this generates a lot of data and should not be used on a production system. Syntax: boolean Default: false
service.wcap.blacklist	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 will be denied access. Syntax: string Default:
service.wcap.maxsessions	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
service.wcap.sessiontimeout	Number of seconds before expiring a session. Syntax: integer Unit: seconds Minimum: 1 Maximum: Maximum int value Default: 1800
store.corpdir.defaultcorpdirectoryurl	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=*)
store.corpdir.enablecorpdir	Enable/Disable corporate directory lookups. Syntax: boolean Default: true

store.corpdir.useldapproxyauth	If true, use LDAP proxy authorization to issue LDAP searches on behalf of the logged in user. If false, use the LDAP Pool credentials for all LDAP searches. This parameter applies only to the default corp directory configuration. Syntax: boolean Default: true
store.dav.*.attachstorehost	Attachment store host. Syntax: string Default:
store.dav.*.attachstoreport	Attachment store port. Syntax: integer Minimum: -2147483648 Maximum: Maximum int value Default: 8008
store.dav.*.backendid	Backend identifier. Syntax: string Default:
store.dav.*.dbdir	Directory path for dav store. Syntax: filepath Default: data/db
store.dav.*.jndiname	JNDI name pointing to this backend's JDBC DataSource, as defined in the J2EE container (e.g. "jdbc/defaultbackend". Syntax: string Default:
store.dav.*.purgedelay	Set the delay between deletion of a resource and its actual removal (purge) from the backend. Setting this value too low may cause sync clients to do a full resync too often. Syntax: long Unit: seconds Minimum: 0 Maximum: Maximum long value Default: 2592000
store.document.password	Password to use when authenticating to a Remote Document Store Server. Syntax: password Default:
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 Remote Document Store Server. Syntax: boolean Default: false

Chapter 31. Calendar Server 7 Configuration Reference

Oracle Communications Calendar Server 7 Configuration Reference



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This information describes configuration files used by Calendar Server. By default, these files are located in the *cal-svr-base*`/config` directory.

Topics:

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 the following:

```
<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 the following:

```
<password parameter>=<value>
<password parameter>=<value>
:
:
```

Document Store Server Configuration File

The `ashttpd.properties` file contains the document store configuration parameters. The following table describes the parameters in the `ashttpd.properties` file.

ashttpd.properties File Parameters

Parameter	Description	Default Value
`service.host`	Server host	*
`service.port`	Server port number	8008
`store.datadir`	Data directory	`/var/opt/sun/comms/davserver`
`store.lockdir`	Lock directory	`/var/opt/sun/comms/davserver/lock`

The format of the `ashttpd.properties` file is the following:

```
<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 as follows:

```
certmap=<name>,<name2>
<name>.<prop1> =<val1>
<name>.<prop2> =<val2>
```

For more information on how to use this file, see Introduction to the Certificate Mapper.

davadmin.properties File

You can provide options to the `davadmin` command by including them in the `davadmin.properties` file.

The following table describes the parameters in the `davadmin.properties` file.

davadmin.properties File Parameters

Parameter	Description
`userid`	Specifies the GlassFish Administrator user ID.
`hostname`	Specifies the GlassFish Server host name.
`port`	Specifies the GlassFish administration port (JMX connector port).
`secure`	Specifies the path to the truststore file used for a secure connection (HTTPS) to Glassfish Server.
`dbtype`	Specifies the type of database, either `mysql` or `oracle`.
`dbhost`	Specifies the database host.
`dbport`	Specifies the database port.
`dbuserid`	Specifies the MySQL Server or Oracle Database user ID for database commands.

The format of the `davadmin.properties` file is the following:

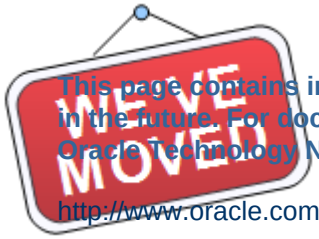
```
<parameter>=<value>
<parameter>=<value>
:
:
```

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. For more information, see Using Calendar Server 7 Notifications.

Chapter 32. Time Zone Database

Time Zone Database



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html

This information lists the Time Zone Database (often called `tz` or `zoneinfo`) IDs that are supported by Calendar Server 7. This list is not necessarily what shows up in WCAP clients. To add any of the following timezones to the WCAP client list, see To Add a New Timezone.

- Africa
- America
- Antarctica
- Arctic
- Asia
- Atlantic
- Australia
- Europe
- Indian
- Pacific

```
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/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/Casey
Antarctica/Davis
Antarctica/DumontDUrville
Antarctica/Macquarie
Antarctica/Mawson
Antarctica/McMurdo
Antarctica/Palmer
Antarctica/Rothera
Antarctica/South_Pole
Antarctica/Syowa
Antarctica/Vostok
```

```
Arctic/Longyearbyen
```

```
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/Azores
Atlantic/Bermuda
Atlantic/Canary
Atlantic/Cape_Verde
Atlantic/Faroe
Atlantic/Madeira
Atlantic/Reykjavik
Atlantic/South_Georgia
Atlantic/St_Helena
Atlantic/Stanley
```

```
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/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/Antananarivo
Indian/Chagos
Indian/Christmas
Indian/Cocos
Indian/Comoro
Indian/Kerguelen
Indian/Mahe
Indian/Maldives
Indian/Mauritius
Indian/Mayotte
Indian/Reunion
```

```
Pacific/Auckland
Pacific/Chatham
Pacific/Chuuk
Pacific/Easter
Pacific/Efate
Pacific/Enderbury
Pacific/Fakaofo
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
```