exo.cs.docs.refguide - 2.1.0-CR02

eXo Collaboration Reference Guide

eXo Platform (eXo Platform)

Acknowledgements

This book is produced by the Wikbook tool. Wikbook is an open source project for converting wiki files into a set of docbook files.

Chapter 1. Configuration

1.1. CS Configuration

1.2. Overview

CS configuration takes place mainly in WEB-INF/conf/cs/cs-plugins-configuration.xml. The configuration details in this article are valid for all components of CS.

1.3. Data Storage

As usual, CS applications and user data are stored in JCR. The default workspace configuration that come with the CS bundle is slightly different to what you may see in basic portal configuration. CS uses a Content Addressable Storage to optimize the emails storage.

```
<value-storage id="collaboration" class="org.exoplatform.services.jcr.impl.storage.value.fs.CASableTreeFileValue
  <properties>
   <property name="path" value="target/temp/values/ws"/>
   <property name="digest-algo" value="MD5"/>
   <property name="vcas-type" value="org.exoplatform.services.jcr.impl.storage.value.cas.JDBCValueContentAddress
   <property name="jdbc-source-name" value="jdbcexo"/>
   <property name="jdbc-dialect" value="hsqldb"/>
  </properties>
  <filters>
   <filter property-type="Binary"/>
  </filters>
</value-storage>
```

1. helps to save disk space by storing a single file for identical workspace binary values. However it has a higher cost in CPU while values are written. If you prefer to favor CPU over disk space, you should configure a classical TreeFileValue Storage :

```
<value-storage id="collaboration" class="org.exoplatform.services.jcr.impl.storage.value.fs.TreeFileValueStorage
  <properties>
    <property name="path" value="../temp/values/collaboration"/>
    <property name="jdbc-source-name" value="jdbcexo"/>
    <property name="jdbc-dialect" value="hsqldb"/>
  </properties>
  <filters>
   <filter property-type="Binary"/>
  </filters>
</value-storage>
```

#info} General configuration of datasources and JCR for the portal is covered in this tutorial : Portal:Database Configuration

{{HContentAddressableValuestorage28CAS29support] to optimize the emails storage.

{code}}{{/HContentAddressableValuestorage28CAS29support] to optimize the emails storage.

{code}}

<value-storage id="collaboration"

class="org.exoplatform.services.jcr.impl.storage.value.fs.CASableTreeFileValueStorage"> <properties> <property name="path" value="target/temp/values/ws"/> <property name="digest-algo" value="MD5"/> <property name="vcas-type" value="org.exoplatform.services.jcr.impl.storage.value.cas.JDBCValueContentAddressStorageImpl"/> <property name="jdbc-source-name" value="jdbcexo"/> <property name="jdbc-dialect" value="hsqldb"/> </properties> <filters> <filter property-type="Binary"/> </filters> </value-storage>

```
# helps to save disk space by storing a single file for identical workspace binary values. However it has a high
```

<value-storage id="collaboration" class="org.exoplatform.services.jcr.impl.storage.value.fs.TreeFileValueStorage"> <properties> <property name="path" value="../temp/values/collaboration"/> <property name="jdbc-source-name" value="jdbcexo"/> <property name="jdbc-dialect" value="hsqldb"/> </properties> <filters> <filter property-type="Binary"/> </filters> </value-storage>

```
#info}


h1. Notifications and Emails
By default we will use email configuration to send email notification and invitation to take part in an event

h2. For 1.3.x version
```

<component> <key>org.exoplatform.services.mail.MailService</key> <type>org.exoplatform.services.mail.impl.MailServiceImpl</type> <init-params> <properties-param> <name>config</name> <property name="mail.smtp.auth.username" value="%smtp*user*id%" /> <property name="mail.smtp.auth.password" value="%smtp*user*password%" /> <property name="mail.smtp.host" value="%smtp*server%" /> <*property name="mail.smtp.port" value="smtp*port" /> <property name="mail.smtp.starttls.enable" value="true" /> <property name="mail.smtp.auth" value="true" /> <property name="mail.smtp.debug" value="false" /> <property name="mail.smtp.socketFactory.port" value="smtp*port" /> <*property name="mail.smtp.socketFactory.class" value="javax.net.ssl.SSLSocketFactory" /> <*property name="mail.smtp.socketFactory.fallback" value="false" /> </*properties-param> </*init-params> </*component>*

```
1.1 For 2.0 version
#info("Since cs 2.0 we move this configuration to gatein/conf/configuration.properties")
```

...

1. EMail

gatein.email.smtp.username=smtp*user*id gatein.email.smtp.password=smtp*user*password gatein.email.smtp.host=smtp*server* *gatein.email.smtp.port=smtp*port gatein.email.smtp.starttls.enable=true gatein.email.smtp.auth=true gatein.email.smtp.socketFactory.port=smtp*port* *gatein.email.smtp.socketFactory.class=javax.net.ssl.SSLSocketFactory*

```
1 Auto update rss and caldav feeds
```

```
#info("Since cs 1.3 we have plug-in to run automatically generate rss and caldav feeds")

The plug-in configuration is applied in *WEB-INF/conf/cs/cs-plugins-configuration.xml*.
The following information will explain details of configuration.

When cs-plugins-configuration.xml file is executed, the component-plugin named *RssRecordsJob* will be referred
```

..... <component-plugin> <name>RssRecordsJob</name> <set-method>addPeriodJob</set-method> <type>org.exoplatform.calendar.service.AutoGeneratePeriodJob</type> <description>add auto update rss, caldav job to the JobSchedulerService</description> <init-params> <properties-param> <name>job.info</name> <description>save the monitor data periodically</description> <property name="jobName" value="UpdateRssCaldavJob"/> <property name="groupName" value="CollaborationSuite"/> <property name="job" value="org.exoplatform.calendar.service.AutoGeneratePeriodJobImp"/> <property name="repeatCount" value="0"/> <property name="period" value="180000"/> <property name="startTime" value="+0"/> <property name="endTime" value=""/> </properties-param> <properties-param> <name>autogenerate.info</name> <description>save the monitor data periodically</description> <property name="portalName" value="portal"/> <property name="event*number" value="100"/> </properties-param> </init-params> </component-plugin> .....

```
Explanation:

* *RssRecordsJob*: unique key to avoid duplicate names.

* *addPeriodJob*: a method to create job.

* *org.exoplatform.calendar.service.AutoGeneratePeriodJob*: The object defines Calendar Service .

h1. See also

- [CS:CS Integration into an Existing Portal]
- [CS:CS Migration from 1-0 to 1-2]
```

1.4. Calendar

1.4.1. Default Calendar Configuration

1.4.2. Overview

eXo Calendar can initialize some default data automatically :

- default personal calendars
- default group calendars
- default event categories
- default calendar setting

Note

This feature relies on organization listeners. They are only executed when eXo's OrganizationService is used to create/delete users/groups. If you provision your organization data by an external tool make sure you do it by calling OrganizationService or you won't be able to benefit of this.

Configuration of this is done by plugins in `cs-plugins-configuration.xml`.

1.4.3. Default personal calendar

Each user can have a default personal calendar created. Use the `NewUserListener` to configure that.

```
<component-plugin>
  <name>calendar.new.user.event.listener</name>
  <set-method>addListenerPlugin</set-method>
  <type>org.exoplatform.calendar.service.impl.NewUserListener</type>
  <description>description</description>
  <init-params>
    <value-param>
      <name>defaultCalendarCategory</name>
      <value>My group</value>
    </value-param>
    <value-param>
      <name>defaultCalendar</name>
      <value>Default</value>
    </value-param>
    <value-param>
      <name>defaultEventCategories</name>
      <value>Anniversary,Calls,Clients,Holiday,Meeting</value>
    </value-param>
  </init-params>
</component-plugin>
```

You may want to change the following :

Table 1.1.

parameter	description	value
defaultCalendarCategory	Name of the calendar group	any string
defaultCalendar	Default the name of a Calendar that will be created for that user	comma separated list of calendar names (ex: personal,business)
defaultEventCategories	Default event categories for user	Comma separated list of category names (ex: lunch,meeting,conference,sport)

Advanced users may want to change the behaviour by using their own plugins. The following information may be interesting to them :

- calendar.new.user.event.listener: unique key to avoid duplicate names.
- addListenerPlugin: This API of Organization service to allow other listener overwrite.

- org.exoplatform.calendar.service.impl.NewUserListener: The class that is used to define Calendar Service to insert database.
- description: Some brief descriptions about plugin.

#info("Since cs 1.3.1 we add exception create default calendar to the configuration file")

- We need to add this in side the init-params> tag

```
<values-param>
  <name>ignoredUsers</name>
  <description>Definition users to ignore create default calendar</description>
  <value>demo</value>
  <value>marry</value>
</values-param>
```

Table 1.2.

parameter	description	value
ignoredUsers	definition users to ignore create default calendar	user id, use multiple by each line

1.4.4. Default calendar settings

Default user settings for eXo Calendar application are also configured by the NewUserListener:

```
  <value-param>
    <name>viewType</name>
    <value>1</value>
  </value-param>
  <value-param>
    <name>timeInterval</name>
    <value>15</value>
  </value-param>
  <value-param>
    <name>weekStartOn</name>
    <value>2</value>
  </value-param>
  <value-param>
    <name>dateFormat</name>
    <value>MM/dd/yyyy</value>
  </value-param>
  <value-param>
    <name>timeFormat</name>
    <value>HH:mm</value>
  </value-param>
  <value-param>
    <name>localeId</name>
    <value>BEL</value>
  </value-param>
  <value-param>
    <name>timezoneId</name>
    <value>Europe/Brussels</value>
  </value-param>
  <value-param>
    <name>baseUrlForRss</name>
    <value></value>
  </value-param>
  <value-param>
    <name>isShowWorkingTime</name>
    <value>false</value>
  </value-param>
```

```
<value-param>
  <name>workingTimeBegin</name>
  <value>08:00</value>
</value-param>
<value-param>
  <name>workingTimeEnd</name>
  <value>18:00</value>
</value-param>
```

You may want to change the following :

Table 1.3.

parameter	description	value
viewType	default view after user logins and goes to Calendar portlet	1-7 (see below)
timeInterval	the time unit interval when you drag and move the event (in Day view and Week view only)	integer in minutes
weekStartOn	day to use as the begining of the week. only affect on Week view	1-7 (see below)
dateFormat	The display format for dates	valid Java Date format>http:--java.sun.com-j2se-1.5.0-docs-api-
timeFormat	The display format for time	valid Java Date format>http:--java.sun.com-j2se-1.5.0-docs-api- (ex : HH:mm)
timezoneId	user time zone	valid TimeZone id>http:--www.unicode.org-cldr-data-docs-desig
localeId	ID of the geographic locale	valid locale ID>http:--userpage.chemie.fu-berlin.de-diverse-
isShowWorkingTime	Indicate if the working time should be hilighted in day view	true,false
workingTimeBegin	the start time in working time	time in timeFormat
workingTimeEnd	the end time in working time.	time in timeFormat

viewType parameter is encoded by a number as follow:

- 0 : Day view
- 1 : Week view
- 2 : Month view
- 3 : Year view
- 4 : List view
- 5 : Schedule view

- 6 : Working days view

weekStartOn parameter is encoded as follow:

- 1 : Sunday
- 2 : Monday
- 3 : Tuesday
- 4 : Wednesday
- 5 : Thursday
- 6 : Friday
- 7 : Saturday

#info("Since CS 2.1 we has a newfeature about timezone. It now takes affect to the created events or tasks ")

- When you add calendar the time zone and language is CAN NOT select able, it will follow your calendar setting
- Event/task will change follow your timezene setting
- When you are sharing the calendar, events and tasks will be display flow your timezone setting

1 Group calendars

To create a default group calendar: Beside using the Organization Portlet to create an user, we also use it to create a group. Calendar service will generate a Calendar for that group (Group calendar means : we have a calendar for all users in that group and the user can do actions in that Calendar). To do that we have some more listeners define below:

```
<component-plugin>
  <name>calendar.new.group.event.listener</name>
  <set-method>addListenerPlugin</set-method>
  <type>org.exoplatform.calendar.service.impl.NewGroupListener</type>
  <description>description</description>
  <init-params>
    <value-param>
      <name>defaultEditPermission</name>
      <value>*.*</value>
    </value-param>
    <value-param>
      <name>defaultViewPermission</name>
      <value>*.*</value>
    </value-param>
    <value-param>
      <name>defaultLocale</name>
      <value>BEL</value>
    </value-param>
    <value-param>
      <name>defaultTimeZone</name>
      <value>Europe/Brussels</value>
    </value-param>
  </init-params>
</component-plugin>
```

- Name: defaultEditPermission: The default Edit permission value: **.** it means all users in that group have

modify and add, remove calendar, events of calendar .(Multi value membership, use coma (,) to split values)

- Name: defaultViewPermission. Value: **.** stands for all in that group can view this calendar and all the events of this calendar (Multi value membership, use coma (,) to split values)
- Name: defaultLocale: the default locale of the calendar Value: BEL (see more locale ids http://userpage.chemie.fu-berlin.de/diverse/doc/ISO_3166.html)
- Name: defaultTimeZone. The default time zone of calendar value: Europe/Brussels (see more for timeZone ids http://www.unicode.org/cldr/data/docs/design/formatting/zone_log.html#windows_ids)
- Note: the permission you can define by two ways : use user name or use membership type permission is suitable (ex: .manager,root,.moderator).

Note

Since cs 1.3.1 we add exception create default group calendar to the configuration file

- We need to add this in side the init-params> tag

```
<values-param>
 <name>ignoredGroups</name>
 <description>Definition group to ignore create default calendar</description>
 <value>/platform/guests</value>
</values-param>
```

Table 1.4.

parameter	description	value
ignoredGroups	Definition group to ignore create default calendar	group id, use line to define multiple value

1.4.5. Email Reminder Plugin Configuration

1.4.6. Overview

eXo Calendar can send event reminders by email. You will probably need to adjust this configuration to your own need. The feature is based on a periodic poll of the stored reminders.

1.4.7. Setting

File to edit: `cs-plugins-configuration.xml`:

```
<external-component-plugins>
    <target-component>org.exoplatform.services.scheduler.JobSchedulerService</target-component>
    <component-plugin>
      <name>RemindersJob</name>
      <set-method>addPeriodJob</set-method>
      <type>org.exoplatform.calendar.service.ReminderPeriodJob</type>
      <description>add a job to the JobSchedulerService</description>
      <init-params>
```

```
        <properties-param>
          <name>job.info</name>
          <description>save the monitor data  periodically</description>
          <property name="jobName" value="ReminderJob"/>
          <property name="groupName" value="CollaborationSuite"/>
          <property name="job" value="org.exoplatform.calendar.service.ReminderJob"/>
          <property name="repeatCount" value="0"/>
          <property name="period" value="180000"/>
          <property name="startTime" value="+0"/>
          <property name="endTime" value=""/>
        </properties-param>
        <properties-param>
          <name>reminder.info</name>
          <description>save the monitor data  periodically</description>
          <property name="timeZone" value="This is TimeZone"/>
          <property name="account" value="reminders@mycompany.com"/>
          <property name="password" value="secret"/>
          <property name="ssl" value="true"/>
          <property name="outgoing" value="smtp.mycompany.com"/>
          <property name="port" value="465"/>
       </properties-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

The interesting settings are :

Table 1.5.

property	**description**	**values**
account	the SMTP account that the reminder service will use to send emails	smtp username
password	the corresponding account password	smtp password
SSL	the option to use Secure Socket Layer for SMTP or not	true/false
Outgoing	SMTP Server hostname	hostname(ex: smtp.mycompany.com) or ip address
Port	SMTP Server's port.	TCP port for the SMTP service
Timezone	default time zone of mail	TZdata

1.4.8. Advanced configuration

Advanced users may have noticed that the feature relies on eXo Job Scheduler. The following information may help you to replace fully the reminder feature by your own :

Below are some explanations about property parameters:

- target-component : `org.exoplatform.services.scheduler.JobSchedulerService`: keep this value. The feature is based on eXo job scheduler service.
- Name: RemindersJob: name of a schedule job. Keep this value.

- set-method: addPeriodJob: plugin registering method. Keep this value
- type: `org.exoplatform.calendar.service.ReminderPeriodJob`: the actual reminder job plugin class name. Keep this value

The following properties define the poll job :

- job.info: key of property. Keep this value.
- jobName: the name of job.
- groupName: the job group's name.
- job: Actual job class name. Keep `org.exoplatform.calendar.service.ReminderJob`
- repeatCount: how many times to run this job. (use '0' which means 'run forever')
- period: the time interval (millisecond) between job executions.

1.4.9. WebDav CalDav Access Configuration

Note

This configuration will allow you access the .ics file with authentication permission

- override portal configuration for webdav access
- make sure you have this in configuration file exo-tomcat/webapps/portal/WEB-INF/web.xml
- for integrated project with cs override here web/csportal/src/main/webapp/WEB-INF/web.xml
- add the filter

```
<filter>
   <filter-name>AnonymousUserContextRedirectionFilter</filter-name>
   <filter-class>org.exoplatform.ws.frameworks.servlet.AnonymousUserContextRedirectionFilter</filter-class>
               <init-param>
     <param-name>context-name</param-name>
     <param-value>/rest/private</param-value>
   </init-param>
 </filter>
```

- add filter mapping

```
  <filter-mapping>
 <!-- This must be before ThreadLocalSessionProviderInitializedFilter for URI /rest/private/*-->
   <filter-name>AnonymousUserContextRedirectionFilter</filter-name>
   <url-pattern>/rest/private/*</url-pattern>
 </filter-mapping>

 <filter-mapping>
   <filter-name>ThreadLocalSessionProviderInitializedFilter</filter-name>
   <url-pattern>/rest/private/*</url-pattern>
```

```
  </filter-mapping>
```

- and servlet mapping

```
  <servlet-mapping>
    <servlet-name>RestServer</servlet-name>
    <url-pattern>/rest/private/*</url-pattern>
  </servlet-mapping>
```

1.5. AdressBook

1.5.1. Default Address Book Configuration

1.5.2. Overview

eXo AddressBook is a contact management application which is part of the eXo Collaboration Suite product. It is written with eXo's Portal:WebUI framework as a portlet. It leverage JCR as its data storage. As usual, main components are Exo Container components and can be configured through XML. System integration and extension points are defined as Component Plugins.

1.5.3. Configuration

CS configuration is applied mainly in **WEB-INF/conf/cs/cs-plugins-configuration.xml**. The following information will explain details of eXo AdressBook configuration.

When cs-plugins-configuration.xml file is executed, the component-plugin named **contact.new.user.event.listener** will be referred to org.exoplatform.contact.service.impl.NewUserListener to create Users for Contact correspondingly to the Users existing in the Organization database. Users for Contact are saved in eXo JCR.

```
......
<component-plugin>
     <name>contact.new.user.event.listener</name>
     <set-method>addListenerPlugin</set-method>
     <type>org.exoplatform.contact.service.impl.NewUserListener</type>
     <description>description</description>
   </component-plugin>
   <component-plugin>
     <name>contact.new.membership.event.listener</name>
     <set-method>addListenerPlugin</set-method>
     <type>org.exoplatform.contact.service.impl.NewMembershipListener</type>
     <description>description</description>
   </component-plugin>
......
```

Explanation:

- **contact.new.user.event.listener**: unique key to avoid duplicate names.

- **addListenerPlugin**: This API of Organization service to allow other listener overwrite.

- **org.exoplatform.contact.service.impl.NewUserListener**: The object defines Contact Service to insert database.

- **description**: Some brief descriptions about plugin.

After Users for Contact are created and saved in eXo JCR, the function postSave() in org.exoplatform.contact.service.impl.NewUserListener will be executed to create Contact and Address Book default for each user. This function also defines public contact group default basing on User's role. Each group contains contacts of User having the same role.

For example, users have administrator role are grouped in Administrator group, users have moderator role are grouped in Moderator group, users have normal user role are grouped in Users group. So, when login into AdressBook Application by specific User, his/her default Address book, default Contact and many public contact groups are shown.

1.5.4. How to configure Contact Service?

1.6. Mail

1.6.1. Default Mail Account Configuration

1.6.2. Overview

exo Mail is an email client application which is part of the eXo Collaboration Suite product. It is written with eXo's Portal:WebUI framework as a portlet. It leverage JCR as its data storage. As usual, main components are Exo Container components and can be configured through XML. System integration and extension points are defined as Component Plugins.

1.6.3. Configuration

You should have 'cs-plugins-configuration.xml' in exo.cs.web.portal project for configuration. Normally, to create default account for each user when an user works with eXo portal, user uses organization portlet to create new user, so the eXo Mail service will create a default account for the created user. To do that, you have to configure <code>cs-plugins-configuration.xml</code> file:

```
      <component-plugin>
               <name>mail.new.user.event.listener</name>
               <set-method>addListenerPlugin</set-method>
               <type>org.exoplatform.mail.service.impl.NewUserListener</type>
               <description>description</description>
         </component-plugin>
```

Some default information for default account must be declared in this code snippet.

```
<component-plugin>
         <name>mail.new.user.event.listener</name>
         <set-method>addListenerPlugin</set-method>
         <type>org.exoplatform.mail.service.impl.NewUserListener</type>
         <description>description</description>
         <init-params>
               <value-param>
```

```
                        <name>protocol</name>
                        <value>imap</value>
                  </value-param>
                  <value-param>
                        <name>ssl</name>
                        <value>false</value>
                  </value-param>
                  <value-param>
                        <name>incomingServer</name>
                        <value>in</value>
                  </value-param>
                  <value-param>
                        <name>incomingPort</name>
                        <value>143</value>
                  </value-param>
                  <value-param>
                        <name>incomingFolder</name>
                        <value>Inbox</value>
                  </value-param>
                  <value-param>
                        <name>outgoingServer</name>
                        <value>out</value>
                  </value-param>
                  <value-param>
                        <name>outgoingPort</name>
                        <value>25</value>
                  </value-param>
            </init-params>
          </component-plugin>
```

Since version 1.1 we do not create default account .

1.6.4. Mail in other CS Modules

1.6.5. Overview

eXo Mail can be used in other CS applications for integration.

1.6.6. Integration

Integrated mail service in calendar application

By default when users create an event, they can use their account to send invitation to other user, but if we not make default email account for user and the user does not create any account, server will use default configuration email account for sending event invitation, it will be used mail service form kernel. Then we need add to `cs-plugins-configuration.xml` file:

```
<component>
                <key>org.exoplatform.services.mail.MailService</key>
                <type>org.exoplatform.services.mail.impl.MailServiceImpl</type>
                        <init-params>
                                <properties-param>
                                        <name>config</name>
                                        <property name="mail.smtp.auth.username" value="%username%" />
                                        <property name="mail.smtp.auth.password" value="%password%" />
                                        <property name="mail.smtp.host" value="smtp.gmail.com" />
                                        <property name="mail.smtp.port" value="465" />
                                         <property name="mail.smtp.starttls.enable" value="true" />
                                        <property name="mail.smtp.auth" value="true" />
                                        <property name="mail.smtp.debug" value="false" />
                                        <property name="mail.smtp.socketFactory.port" value="465" />
                                        <property name="mail.smtp.socketFactory.class" value="javax.net.ssl.SSLS
                                        <property name="mail.smtp.socketFactory.fallback" value="false" />
                                </properties-param>
                        </init-params>
```

```
        </component>
```

Note

Some value will be defined by mail server provider, so you need to check the provider configuration before putting any value

1.6.7. Mail Sending

Using mail service to send mail from contact application

As the same way system processed in calendar application to send email form one contact to other contact, it will use the default account of the user to send mail. If having no default and no created account, the system will use the mail service in kernel to send mail. The configuration use for both calendar and contact application.

1.6.8. Authentication Logout Listener

In eXo Mail, when a user check messages for one account, the remote mailbox fetch is performed as a background job. Before CS 1.2, the job was continuing until all messages had been retrieved or when the user stopped the check through the UI. Hence, even when a user was not logged in, the background job was continuing. This can be resource intensive for the server if many users have large mailboxes.

Since CS 1.2, we added the capability to halt the background job when the user session terminates (logout or time out). It makes CS more friendly with server resources. If you want to activate this feature, you need to add a bunch of xml congifuration in cs-plugins-configuration.xml :

```
<external-component-plugins>
          <target-component>org.exoplatform.services.listener.ListenerService</target-component>
                 <component-plugin>
          <name>exo.core.security.ConversationRegistry.unregister</name>
                   <set-method>addListener</set-method>
          <type>org.exoplatform.mail.service.AuthenticationLogoutListener</type>
                    <description>description</description>
          </component-plugin>
</external-component-plugins>
```

1.7. Chat

1.7.1. eXo Chat Server configuration

1.7.2. Openfire Configuration

eXo CS 1.3 chat services is a Jabber engine powered by Openfire. eXo will delegate the actual Jabber protocol communication to Openfire.

Note

CS 1.3 has been tested only with Openfire 3.4.5. Should you try to upgrade yourself, this would be at your own risks

You have full latitude to configure Openfire. There are two possible ways do it :

- the admin console : http://localhost:9090/
- the openfire.xml file, in $openfire*home/conf/*

Note

CS 2.0 has been updated with Openfire 3.6.4. You need to notice more about configuration in openfire.xml file, refer [here |http://wiki.exoplatform.org/xwiki/bin/view/CS/Chat+Configuration#eXo addition configuration] for more information

1.7.2.1. Configuring openfire.xml

The Openfire server has a single configuration file called openfire.xml and located under exo-openfire/conf directory. Configuration is based on properties expressed in an XML syntax. For example, to set property *prop.name.is.blah=value*, you would write this xml snippet :

```
<prop><name><is><blah>value</blah></is></name></prop>
```

Openfire has an extensive list of configuration properties. You can read a **list of all properties** on this page : http://www.igniterealtime.org/community/docs/DOC-1061

1.7.2.2. eXo specific configuration

eXo CS bundles comes with a pre-configured openfire server. It is bundled with some eXo plugins and configurations that allow connectivity to eXo. The key properties for integration are :

- **provider.auth.className** : An implementation of the AuthProvider interface for authentication of users on the chat server
- **provider.users.className** : An implementation of the UserProvider interface to which openfire will delegate users management
- **provider.groups.className** : An implementation of the GroupProvider interfact to which openfire will delegate groups management

eXo provides implementations for these 3 interfaces with ExoAuthProvider, ExoUserProvider, ExoGroupProvider. These implementations use eXo hosted REST services and let you configure the endpoints within the openfire.xml file with additional properties :

Table 1.6.

Property	Description	default value
provider.authorizedUser.name	username to authenticate against the HTTP REST service	root
provider.authorizedUser.password	password matching with provider.authorizeduser.name	password

Property	Description	default value
exoAuthProvider.authenticationURL	URL to authenticate users	http://127.0.0.1:8080/rest/organization/authentic
exoAuthProvider.authenticationMethod	HTTP method used to pass parameters	POST
exoUserProvider.findUsersURL	URL to find all users	http://127.0.0.1:8080/rest/organization/xml/user
exoUserProvider.findUsersMethod	HTTP method for user/find-all	GET
exoUserProvider.getUsersURL	URL to retrieve a range of users	http://127.0.0.1:8080/rest/organization/xml/user
exoUserProvider.getUsersMethod	HTTP method for user/view-range	GET
exoUserProvider.usersCountURL	URL to count users	http://127.0.0.1:8080/rest/organization/xml/user
exoUserProvider.usersCountMethod	HTTP method for user/count	GET
exoUserProvider.userInfoURL	URL to get user info	http://127.0.0.1:8080/rest/organization/xml/user
exoUserProvider.userInfoMethod	HTTP method for user/info	GET
exoGroupProvider.groupInfoURL	URL to get group info	http://127.0.0.1:8080/rest/organization/xml/grou
exoGroupProvider.groupInfoMethod	HTTP method for info	GET
exoGroupProvider.getGroupsAllURL	URL to view all groups	http://127.0.0.1:8080/rest/organization/xml/grou
exoGroupProvider.getGroupsAllMethod	HTTP method for group/view-all	GET
exoGroupProvider.getGroupsRangeURL	URL to view a group range	http://127.0.0.1:8080/rest/organization/xml/grou
exoGroupProvider.getGroupsRangeMethod	HTTP method for group/view-from-to	GET
exoGroupProvider.getGroupsForUserURL	URL to get groups for a user	http://127.0.0.1:8080/rest/organization/xml/grou
exoGroupProvider.getGroupsForUserMethod	HTTP method for groups-for-user	GET
exoGroupProvider.groupsCountURL	URL to count groups	http://127.0.0.1:8080/rest/organization/xml/grou
exoGroupProvider.groupsCountMethod	HTTP method for group/count	GET

As you can see, the default settings will work for a case where eXo is is deployed on the same host as openfire, on port 8080 and bound to 127.0.0.1 IP address. This is actually the case for the quick steps given in the Install Guide. However, you will need to change them to accommodate your own server deployment.

1.7.2.3. eXo addition configuration

```
<env>
      <serverBaseURL>http://localhost:8080/</serverBaseURL>
      <!--
          "restContextName" is used to specify Openfire server is dedicated for which portal.
          If "eXo.env.restContextName" system property exists, it will override this value.
          "eXo.env.restContextName" system property can be set by specifying the -D option
          to the java command when running Openfire. Example: If Openfire server is dedicated for "portal" porta
          the command will have following format : java -DeXo.env.restContextName=rest -jar ../lib/startup.jar .
          If Openfire server is dedicated for "csdemo" portal, the command will have following format:
          java -DeXo.env.restContextName=rest-csdemo -jar ../lib/startup.jar .
          By default, Openfire server is dedicated to "portal" portal.
       -->
```

```
      <restContextName>rest</restContextName>
</env>
```

1 eXo Service Configuration You will need to tell eXo how to connect to the openfire instance. This is achieved by configuration of the XMPPMessenger component found in exo.cs.eXoApplication.chat.service.

```
<component>
    <type>org.exoplatform.services.xmpp.connection.impl.XMPPMessenger</type>
    <init-params>
      <properties-param>
        <name>openfire-connection-conf</name>
        <property name="host" value="127.0.0.1" />
        <property name="port" value="5222" />
      </properties-param>
...
```

Beside openfire-connection-conf, other properties can be configured on the XMPPMessenger. Below is the full list :

Table 1.7.

properties-param	property name	description	default value
openfire-connection-conf	host	IP address or hostname for the openfire server	127.0.0.1
port	port to connect to on the openfire server. Should be the same that set in openfire configuration "Client to Server"	5222	
send-file	timeout	timeout before aborting attempt to establish a file transfer	openfire defaults

1 System Configuration

Openfire makes use of several ports for communication.

Table 1.8.

Interface	Port	Type	Description		
All addresses	5222	Client to Server	The standard port for clients is to connect to the server. Connections may or may not be encrypted. You can update the security settings for this		

			port.		
All addresses	9090 && 9091		Admin Console		The port used for respectively the unsecured and secured Openfire Admin Console access.
All addresses	7777		File Transfer Proxy		The port used for the proxy service that allows file transfers to occur between two entities on the XMPP network.
All addresses	3478 & 3479		STUN Service		The port used for the service that ensures connectivity between entities when behind a NAT.

You can view above table in http://hostname:9090/index.jsp after you are logged in to openfire's web console and also customize those ports by yourself.

1 AS configuration

To unable propagation of identity across the chat webapp, it is required that you enable the SSO valve on omcat-based ASes

- For jboss server, edit jboss/erver/efault/eploy/boss-web.deployer/erver.xml
- For tomcat server edit tomcat/onf/erver.xml

The valve should already be there, you just need to uncomment it if not already done..

```
<Valve className="org.apache.catalina.authenticator.SingleSignOn" />
```

In clustered deployment, you may want to use ClusteredSingleSignOn instead.

```
<Valve className="org.jboss.web.tomcat.service.sso.ClusteredSingleSignOn" />
```

1.8. Content

1.8.1. User RSS Feeds Configuration

Place of the configuration file

- exo-tomcat/webapps/portal/WEB-INF/conf/portal/user/%username%/content.xml

or override on your own product

- web/csportal/src/main/webapp/WEB-INF/conf/portal/user/%username%/content.xml

```
<?xml version="1.0" encoding="utf-8"?>
<content-navigation>
  <owner>%username%</owner>
  <description> description for content navigation</description>
  <content-nodes>
    <content-node>
            <id>news</id>
            <label>World News</label>
            <description></description>
            <type>desc</type>

            <content-node>
              <id>cnn</id>
              <url>http://rss.cnn.com/rss/cnn_topstories.rss</url>
              <label>CNN</label>
              <description></description>
              <type>rss</type>
            </content-node>

    </content-node>

  </content-nodes>
</content-navigation>
```

1.8.2. RSS Reader with Proxy

1.8.2.1. Proxy without authentication

To use a proxy without authentication, you need to add some parameters to java command line in eXo.bat or eXo.sh : **-Dhttpclient.proxy.host=proxyHost ~~Dhttpclient.proxy.port=proxyPort~~ Dhttpclient.proxy.username="" ~~Dhttpclient.proxy.password=""~~**

1.8.2.2. Proxy with authentication

To use a proxy with authentication, you need to add some parameters to java command line in eXo.bat or eXo.sh : **-Dhttpclient.proxy.host=proxyHost ~~Dhttpclient.proxy.port=proxyPort~~ Dhttpclient.proxy.username="proxyUsername" ~~Dhttpclient.proxy.password="proxyPassword"~~**

1.8.2.3. Proxy with NTLM authentication

To use a proxy with NTLM authentication, you need to add some parameters to java command line in eXo.bat or eXo.sh : **-Dhttpclient.proxy.host=proxyHost ~~Dhttpclient.proxy.port=proxyPort~~ Dhttpclient.proxy.username="proxyUsername" ~~Dhttpclient.proxy.password="proxyPassword"~~ Dhttpclient.proxy.ntlm.host=ntlmHost ~~Dhttpclient.proxy.ntlm.domain=domain~~**

- ntlmHost is the name of the machine on which tomcat is running.

- domain is the NTLM domain

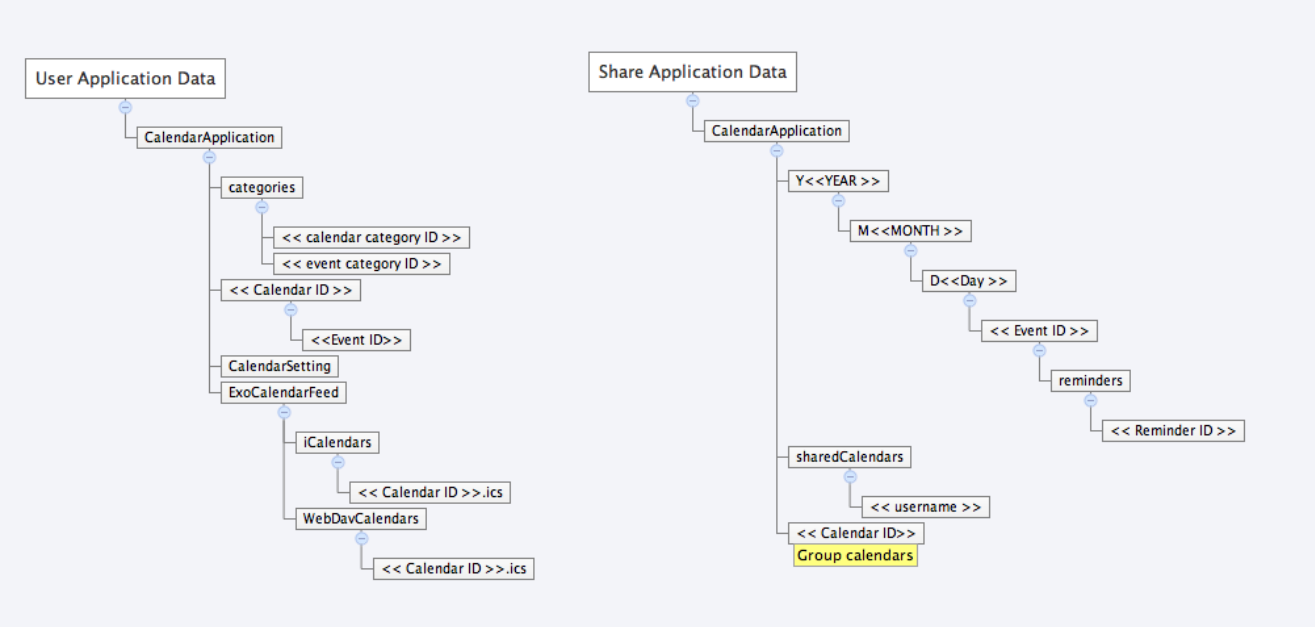
Chapter 2. Development

2.1. Calendar

2.1.1. Calendar JCR Structure

2.1.2. Overview

eXo Calendar is a JCR based application. Calendar information is saved in eXo-JCR under user's application data directory.



2.1.3. Categories

Categories are used to classify events and tasks. For example, you could use a 'Meeting' category for all meetings with your colleagues or customers in your company. By default, eXo Calendar provides 5 categories:

- Calls
- Meeting
- Holiday
- Clients
- Anniversary

Categories are stored in `/Users/%username%/ApplicationData/CalendarApplication/categories/%calendarCategory`*`id%`*

2.1.4. Event Categories

Event Categories are used to classify events and tasks. For example, you could use a 'Meeting' category for all meetings with your colleagues or customers in your company. By default, eXo Calendar provides 5 categories: Calls, Meeting, Holiday, Clients, Anniversary.Event Category is storage in /Users/%username%/ApplicationData/CalendarApplication/%eventCategory*id% with shared events we will support reference event*category for receive user.

2.1.5. Calendar Settings

Calendar Settings allow personalizing your calendar view according to available features. eXo calendar brings you a wide range of custom settings to set your own calendar.

The date base will store here:

<tt> /Users/%user*name%/ApplicationData/CalendarApplication/calendarSetting* </tt>

```
<nodeType name="exo:calendarSetting" isMixin="false" hasOrderableChildNodes="false" primaryItemName="">
                  <supertypes>
                          <supertype>nt:base</supertype>
                  </supertypes>
                  <propertyDefinitions>
                          <propertyDefinition name="exo:viewType" requiredType="String" autoCreated="false" manc
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="exo:timeInterval" requiredType="Long" autoCreated="false" ma
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="exo:weekStartOn" requiredType="String" autoCreated="false" m
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="exo:dateFormat" requiredType="String" autoCreated="false" ma
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="exo:timeFormat" requiredType="String" autoCreated="false" ma
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="exo:location" requiredType="String" autoCreated="false" manc
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="exo:timeZone" requiredType="String" autoCreated="false" manc
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="exo:showWorkingTime" requiredType="Boolean" autoCreated="fal
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="exo:workingTimeBegin" requiredType="String" autoCreated="fal
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="exo:workingTimeEnd" requiredType="String" autoCreated="false
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="exo:baseUrl" requiredType="String" autoCreated="false" manda
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="exo:defaultPrivateCalendars" requiredType="String" autoCreat
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="exo:defaultPublicCalendars" requiredType="String" autoCreate
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="exo:defaultSharedCalendars" requiredType="String" autoCreate
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="exo:sharedCalendarsColors" requiredType="String" autoCreated
                                  <valueConstraints/>
                          </propertyDefinition>
                  </propertyDefinitions>
                  </nodeType>
```

since version 1.1 has new property exo:sendOption

```
                        <propertyDefinition name="exo:sendOption" requiredType="String" autoCreated="false" mand
                                  <valueConstraints/>
</propertyDefinition>
```

Table 2.1.

Setting property	Description	Possible values
exo:viewType	default view each time user log-in and goes to calendar application	list or format of value(s)

Calendar Settings are stored in `/Users/%username%/ApplicationData/CalendarApplication`.

2.1.6. Calendars

Calendars After being created, a calendar will be categorized in the group that you choose for it. It can be added tasks/events, edited, deleted, exported and shared with others users. There are three types of calendar: personal calendars, shared calendars and group calendars. Calendar is storage in */Users/%username%/ApplicationData/CalendarApplication/%calendarid%.*

2.1.7. Reminders

Reminders is the system notification when user need to remind the upcoming event, it has two kind of reminders:

- **Popup Reminders** : Notify user by a popup on user interface.
- **Email Reminders** : Notify user by email.

And both of them will store data in:

`/Public application place/CalendarApplication/Y+%year to created event%/M+%month to created event%/D+%day to creted event%/reminders/%eventid%/%reminderid%`

For example, a reminder (id=9234) for an event (id=19445) that will take place the 15 of October 2010 would be stored as follows: `/Application Data/CalendarApplication/Y2010/M10/D15/reminders/19445/9234`.

2.1.8. Usage of eXo Calendar API

Note

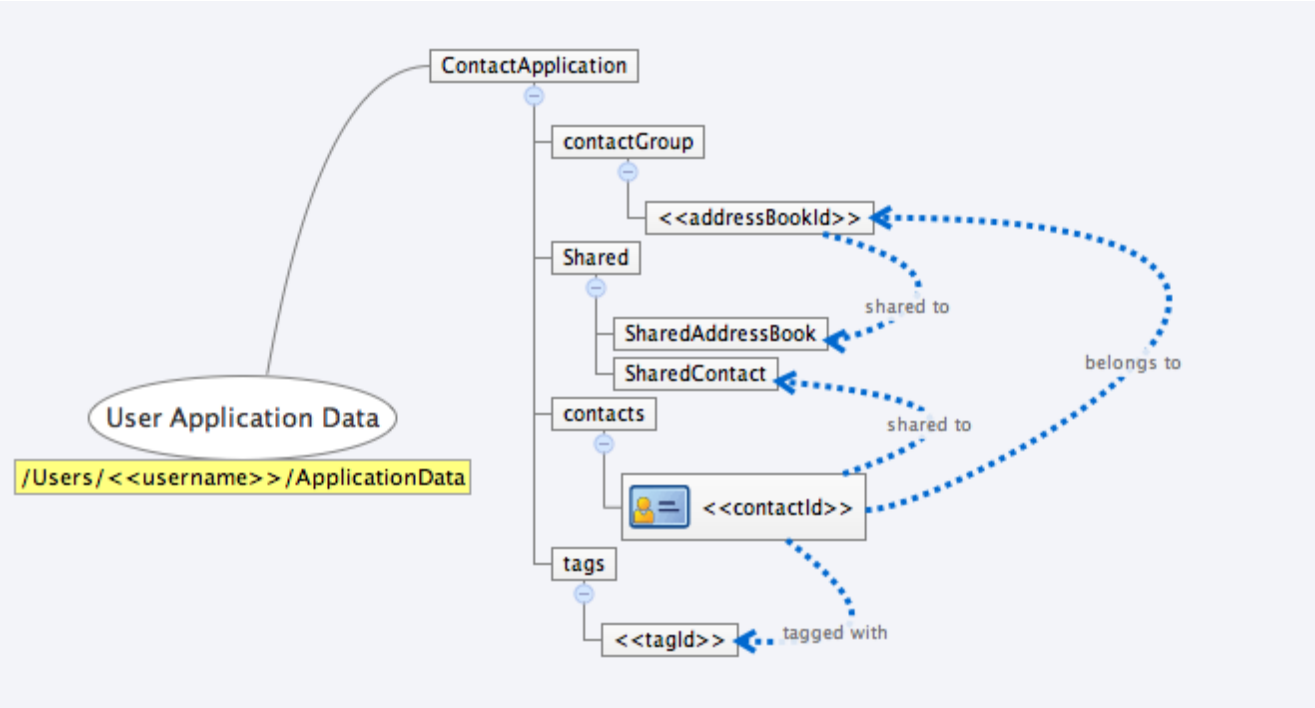
TODO. Refer to [Components Registry|Products Tech Overview:Components Registry#CS]

2.2. AddressBook

2.2.1. AddressBook JCR Structure

2.2.2. Overview

eXo AddressBook stores its data in JCR under each user's Application Data node. The following diagram shows how AddressBook storage structure is organized.



ContactApplication : is the application data root for AddressBook application. Each user has its own application data node. For example for user *root* it would be `/Users/root/ApplicationData/ContactApplication`

contactGroup : is the root node for all personal address books of the user. Each address book is a node of type `exo:contactGroup`. An address book node is named after the ID of the address book. it also contains information such as the displayed name, description and permissions.

contacts is the root node for all personal contacts of the user. Each contact is a node of type `exo:contact`.

Shared is the root node for data shared to the user. In eXo AddressBook one can be shared individual contacts and also entire address books. When shared, an `exo:contactGroup` or an `exo:contact` receives the mixin `exo:contactShared` which gives it the ability to reference one of the node below :

- **SharedAddressBook** : is the `mix:referenceable` node that will be referenced by address books shared to this user
- **SharedContact** : is the `mix:referenceable` node that will be referenced by contacts shared to this user.

tags is the root node for personal tags created by the user. Each tag is a node of type `exo:contactTag`.

Note

TODO : public contacts and address books. What we put in /exo:applications/ContactApplication/contacts

.

We do not put any contact or address book in /exo:applications/.. All **public contacts** we storage in personal path like /Users/%user-id%/ApplicationData/ContactApplication/contacts . And these contacts specified by owner property equals true if it is public contact.

2.2.3. Nodetypes

2.2.3.1. Address Book

Address Books are stored with nodetype `exo:contactGroup`.

Table 2.2.

Name	Description	Type
exo:id	ID of the addressbook, same as the node name	String
exo:name	Display name for the address book	String
exo:description	Description for this address Book	String
exo:viewPermissionUsers	List of usernames that have view permissions on this address book	String (multiple)
exo:editPermissionUsers	List of usernames that have edit permissions on this address book	String (multiple)
exo:viewPermissionGroups	List of groups (/group/subgroup) that have view permissions on this address book	String (multiple)
exo:editPermissionGroups	List of groups that have view permissions on this address book	String (multiple)

2.2.3.2. Contact

Contacts are stored with nodetype `exo:contact`.

Table 2.3.

Name	Description	Type
exo:id	ID of the contact (same as node name)	String
exo:path	-	-
exo:categories	list of address book IDs (name and exo:id property) where this contact appears	String (multiple)
exo:viewPermissionUsers	List of usernames that have view permissions on this contact	String (multiple)

Name	Description	Type
exo:editPermissionUsers	List of usernames that have edit permissions on this contact	String (multiple)
exo:viewPermissionGroups	List of groups (/group/subgroup) that have view permissions on this contact	String (multiple)
exo:editPermissionGroups	List of groups that have view permissions on this contact	String (multiple)
exo:tags	List of tag IDs (name and exo:id property)	String (multiple)
exo:isOwner		Boolean
exo:ownerId	username of owner of this contact	String
exo:lastUpdated	Date of Last update	Date
exo:fullName	Full contact name (initialized with first + last name)	String
exo:firstName	Firstname	String
exo:lastName	Lastname	String
exo:nickname	User screen name	String
exo:gender	gender	String {{Male,Female}}{{/Male,Female}}
exo:birthday	Birth date	Date
exo:jobTitle	Job title	String
exo:emailAddress	email addresses (period-separated)	String
exo:exoId	eXo Chat ID	String
exo:googleId	GTalk ID	String
exo:msnId	MSN Chat ID	String
exo:aolId	GAIM ID	String
exo:yahooId	Yahoo Chat ID	String
exo:ircId	IRC ID	String
exo:skypeId	Skype ID	String
exo:icqId	ICS Chat ID	String
exo:homeAddress	home address	String
exo:homeCity	home city	String
exo:homeState_province	home state/province	String
exo:homePostalCode	home postal code	String
exo:homeCountry	home country	String

Name	Description	Type
exo:homePhone1	home phone	String
exo:homePhone2	secondary home phone	String
exo:homeFax	home Fax	String
exo:personalSite	Personal website	String
exo:workAddress	Work address	String
exo:workCity	Work city	String
exo:workState_province	Work state/province	String
exo:workPostalCode	Work postal code	String
exo:workCountry	Work country	String
exo:workPhone1	Professional phone	String
exo:workPhone2	secondary Professional phone	String
exo:workFax	Professional Fax	String
exo:mobilePhone	Mobile phone number	String
exo:webSite	Professional website	String
exo:note	additionnal notes	String

2.2.3.3. Shared Contacts and Address Books

Shared contacts and address books are represented by the mixin `exo:contactShared`

Table 2.4.

Name	Description	Type
exo:sharedId	Reference to the SharedAddressBook or SharedContact node of target user	Reference (multiple)
exo:sharedUserId	username of the sharing user	String

2.2.4. Tags

Tags are represented by the mixin `exo:contactTag`.

Table 2.5.

Name	Description	Type
exo:iD	ID of the tag	String
exo:name	display name of the tag	String

Name	Description	Type
exo:description	description for this tag	String
exo:color	name of the color (web palette) to use	String

2.2.5. Usage of eXo AdressBook API

Note

Refer to the [Components Registry|Products Tech Overview:Components Registry#CS]

Note

Since 1.3.4 and 2.0

This feature let you can customize your show hidden group of use by modifying xml file

- For 1.3.4 and higher : modify this :
- eXoApplication/contact/service/src/main/java/conf/portal/configuration.xml
- For 2.0 : you have to config in which portal do you need take affect
- csdemo portal : demo/war/src/main/webapp/WEB-INF/conf/csdemo/cs/contact/contact-service-configuration.xml
- extension : extension/webapp/src/main/webapp/WEB-INF/cs-extension/cs/contact/contact-service-configuration.xml

```
<component>
    <key>org.exoplatform.contact.service.ContactService</key>
    <type>org.exoplatform.contact.service.impl.ContactServiceImpl</type>
    <init-params>
      <values-param>
        <name>UserCanSeeAllGroupAddressBooks</name>
        <description>if true, all groups will be listed, if false, only groups of the user will be listed</descr
        <value>false</value>
      </values-param>
      <values-param>
        <name>NonPublicGroups</name>
        <description>Groups that shouldn't be displayed in broadcast list. Wildcards may be used in groups name<
        <value>/platform/users</value>
        <value>/organization/divisions/*</value
      </values-param>
    </init-params>
  </component>
```

UserCanSeeAllGroupAddressBooks : When set to false (default value), the user will be able to list adress books only for groups that he belongs to When set to true, the user will be able to list adressbooks even for groups that he does not belong to.

NonPublicGroups Defines the list of groups that will never be used as addressbooks. They will not be

displayed as group address books Wildcard " **character may be used.**

2.3. Mail

2.3.1. Mail JCR Structure

Mail are saved in eXo-JCR in form of node.

By default when initialize data all data of mail application will store at : /Users/%username%/ApplicationData/MailApplication

```
+ Users
   + %username%
     + ApplicationData
       + MailApplication
         + %account_id%
           + %folder_id%
           + %message_id%
             +%attachment_id%
           + %tag_id%
```

Account: In order to get mails from the other mail service to eXo Mail, you need to create an account in eXo Mail which connects to a real existing email account.Account will add to this /Users/%username%/ApplicationData/MailApplication/%account*id%*

Folders: Using folders makes your message management easier and more flexible. There are some default folders that are generated automatically after completing creating a new account successfully: Inbox, Drafts, Sent, Spam, Trash. Folder is storage in : /Users/%username%/ApplicationData/MailApplication/%account*id%/%folder*id%.

Messages: is the exchange of computer-stored messages by telecommunication. Message in a account storage in : /Users/%username%/ApplicationData/MailApplication/%account*id%/%message*id%.

Tags: eXo Mail allows assigning tags to messages. Tags are used as labels which allows filtering or categorizing messages from different folders. A single message can be assigned to many tags at a time. Using tags make it easy for you to find your messages independently of the folder you stored them.Tag in a account storage in : /Users/%username%/ApplicationData/MailApplication/%account*id%/%tag*id%.

Attachment: the attach file of a message is a part of message's body so when we store it to the data we save in /Users/%username%/ApplicationData/MailApplication/%account*id%/%message*id%/%attachment*id%*. *if when the attach was corrupted by any reason we will mark it like broken attach file*

2.3.2. IMAP Support

Note

CS 1.3 and later provides full IMAP support for mail

- When creating an account with a server which supports IMAP, CS synchronizes Imap folders and messages when fetching mails.

- With IMAP support, exo mail provides lazy loading of messages: Initially eXo Mail only fetches message headers and when you click to view details it fetches the message content. The time for loading messages and folders depends on the network speed, the number of messages in the folders and the number folders of account. So it can considerably slowdown the request progress. (Mostly network speed problem).
- eXo Mail allows the user to limit the number of messages by defining a minimum date. Then, only messages received after that date are fetched from the mail server.
- eXo Mail allows to connect the same account at the same time from many clients. However, the mail server the number of concurrent connections, for example : gmail has limitation 10 connections at the same time on one Imap account.

2.3.3. Portlet Integration of eXo API

Import eXo Mail Service to your portlet

Insert this code snippet to pom.xml file of your portlet.

```
<dependency>
  <groupId>org.exoplatform.cs</groupId>
  <artifactId>exo.cs.eXoApplication.mail.service</artifactId>
  <version>1.0</version>
</dependency>
```

API Usage

eXo Mail Service provides all APIs for working with mail as send mail, check mail, store mail to JCR... but maybe other portlets only use send mail function to send an email.

Send mail :You can use the following APIs to send an email:

```
public void sendMessages(List<Message> msgList, ServerConfiguration serverConfig) throws Exception ;

public Message sendMessage(SessionProvider sProvider, String username, String accId, Message message) throws E

public Message sendMessage(SessionProvider sProvider, String username, Message message) throws Exception ;
```

2.4. Chat

2.4.1. Chat JCR Structure

Chat data is saved in eXo-JCR in form of node.

conversations: when user create a conversation, information about this conversation storage in public path : /exo:applicaton/eXoChat/history/conversations/%conversation*id%*.

participants: in a group chat usually contains participants , these participants storage in public path : /exo:applicaton/eXoChat/history/participants/%participant*id%*.

state : state of chat window include width,height,top,left,visible,extra,activeTabId,tabs... storage in private path : /Users/root/ApplicationData/eXoChat/uistate.

2.4.2. Portlet Integration of eXo API

Import eXo Chat Service to your portlet : Insert this code snippet to pom.xml file of your portlet :

```
<dependency>
    <groupId>org.exoplatform.cs</groupId>
    <artifactId>exo.cs.eXoApplication.chat.service</artifactId>
    <version>1.3</version>
  </dependency>
```

API Usage

eXo Chat Service provides some APIs for working with chat as create room, join room...

Create room :You can use the following API to create a room:

```
public Response createRoom(@URIParam("username") String username,
                           @QueryParam("room") String room,
                           @QueryParam("nickname") String nickname)
```

Join room :You can use the following API to join a room:

```
public Response joinRoom(@URIParam("username") String username,
                         @QueryParam("room") String room,
                         @QueryParam("nickname") String nickname,
                         @QueryParam("password") String password)
```

2.4.3. How a chat session works and maintained

2.4.4. Overview

This page explains how a chat session works and maintained.

Chat service utilizes Smack library for communicating with XMPP servers (ex: Openfire). The connection between a portal user and a XMPP server is managed and maintained by **XMPPSessionImpl** class which wraps the class org.jivesoftware.smack.XMPPConnection in Smack library. The detail document about Smack is located at http://www.igniterealtime.org/builds/smack/docs/latest/documentation/

Note

Smack includes built-in debugging consoles that will let you track all XML traffic between the client and XMPP server . Debugging mode can be enabled by set the Java system property smack.debugEnabled to true when start application server :

-Dsmack.debugEnabled=true

The more detail can be found at http://www.igniterealtime.org/builds/smack/docs/latest/documentation/debugging.html

2.4.5. How connection is maintain between user and Openfire server

When a new XMPPSessionImpl is created and destroyed:

2.4.5.1. A new XMPPSessionImpl is created when:

CS 1.3.x:

- After a portal login,
- After using online action in Chat(ChatBar) portlet
- After Chat(ChatBar) portlet is loaded (after switch to Chat portlet, after refresh Chat(ChatBar) portlet, or after switch to use other portlets when using ChatBar).

CS 2.0:

- After a portal login
- After using online action in Chat(ChatBar) portlet.

2.4.5.2. A XMPPSessionImpl is destroyed when:

CS 1.3.x:

- After a portal logout
- After using offline action in Chat(ChatBar) portlet
- When **unload** event of window is called: in the progression of refreshing Chat(ChatBar) portlet, in the progression of switching from Chat portlet to other portlet, or in the progression of switching to use other portlets when using ChatBar.

CS 2.0:

- After a portal logout
- After using offline action in Chat(ChatBar) portlet.

Two functions: `login2` and `logout` of **XMPPRestService** is responsible for creating a new **XMPPSessionImpl** and destroying an existed XMPPSessionImpl. They can be called by listeners: **AuthenticationLoginListener**, **AuthenticationLogoutListener** or from client(browser) through Rest protocol (`jabberLogin`, `jabberLogout` in **UIMainChatWindow**.js).

Note

If ignoring online, offline actions of Chat(ChatBar) portlet:

CS 1.3.x: The chat session only maintained when standing at chat or chatbar portlet (don't refresh, don't switch portlet).

CS 2.0: The chat session is synchronized with the portal session.

2.4.6. How chat connection and openfire connection works

2.4.6.1. Establishing a Connection

The XMPPConnection class is used to create a connection to an XMPP server. Below are code examples in **XMPPSessionImpl** for making a connection:

```
public class XMPPSessionImpl {
  private final XMPPConnection connection_;
  private MultiUserChatManager multiUserChatManager;
  private FileTransferManager fileTransferManager;
  private final ContinuationServiceDelegate delegate_;
  ...
  connection_ = new XMPPConnection(XMPPMessenger.getConnectionConfiguration());
  connection_.connect();
  connection_.login(username, password, null);
  multiUserChatManager = new MultiUserChatManager();
  fileTransferManager = new FileTransferManager(connection_);
  ...
}
```

2.4.6.2. Reading and Writing Packets

2.4.6.2.1. Writing Packets

Each message to the XMPP server from a client and vice versa is called a packet and is sent as XML. The org.jivesoftware.smack.packet package contains classes that encapsulate the three different basic packet types allowed by XMPP (message, presence, and IQ).

Below are code examples in **XMPPSessionImpl** for sending mesage between 2 user

```
public void sendMessage(Message message) {
  if (connection_.isConnected()) {
    connection_.sendPacket(message);
  }
}
```

Below are code examples in **XMPPSessionImpl** for messages to users in a room.

```
public void sendMessageToMUC(String room, String body) throws XMPPException {
  MultiUserChat chat = multiUserChatManager.getMultiUserChat(room);
  Message message = chat.createMessage();
  message.setBody(body);
  message.setFrom(getUsername());
  chat.sendMessage(message);
}
```

2.4.6.2.2. Reading Packets

Smack provides two ways to read incoming packets: **PacketListener**, and **PacketCollector**. Both use **PacketFilter** instances to determine which packets should be processed. A packet listener is used when you want to take some action whenever a packet happens to come in. Packet listeners can be created using an

XMPPConnection instance.

Below are code examples in **XMPPSessionImpl** to read incoming messages and send to browser through an appropriate cometd channel.

```
MessageFilter msgFilter = new MessageFilter();
connection_.addPacketListener(new PacketListener() {
  public void processPacket(Packet packet) {
  ...
  EventsBean eventsBean = new EventsBean();
  eventsBean.addMessage((Message)packet);
  JsonValue json = generatorImpl.createJsonObject(eventsBean);
  delegate_.sendMessage(username_, CometdChannels.MESSAGE, json.toString(), null);
  ...
  }
}, msgFilter);
```

It's similar like above with other listener: **FileTransferListener** for file transfer, **RosterListener** for users status in contact list, **InvitationListener** for chat room, **ErrorMessageFilter**, **SubscriptionFilter** for request/response to add an user to a contact list...

2.4.7. How messages are sended

2.4.7.1. How messages are sended:

Client(Browser) ->`jabberSendMessage` in UIMainChatWindow.js ~~-Rest protocol--> `sendMessage` or `sendMUCMessage` in RESTXMPPService -> XMPPSessionImpl~~ -XMPP protocol-->XMPP server (Openfire).

2.4.7.2. How messages are received:

XMPP server ~~-XMPP protocol--> a proper PacketListener of XMPPSessionImpl -> a proper channel of Cometd Server~~ -Bayeux protocol--> a proper listener(`messageListener`, `groupChatListener`, `presenceListener`, `rosterListener`, `subscriptionListener`, `fileExchangeListener`) of Client(Browser) (these listeners are in UIMainChatWindow.js)

2.4.8. How Client(Browser) utilizes Cometd

Browser utilize Cometd through Bayeux protocol which is implemented in **CSCometd.js**

Steps:

2.4.8.1. Handshake

1. Bayeux client initiates a connection negotiation by sending a message to the "/meta/handshake" channel.
2. successful handshake response MUST contain the message fields in which has **clientId** : A newly generated unique ID string.

2.4.8.2. Connect

After a Bayeux client has discovered the server's capabilities with a handshake exchange, a connection is established by sending a message to the "/meta/connect" channel. Request contain the **clientId** returned in the handshake response.

2.4.8.3. Subcribe

1. connected Bayeux client may send subscribe messages to register interest in a channel and to request that messages published to that channel are delivered to itself.

2.4.8.4. Publish

1. Bayeux client can publish events on a channel by sending event messages. Application events are published in event messages sent from a Bayeux client to a Bayeux server and are delivered in event messages sent from a Bayeux server to a Bayeux client. Event messages MUST be delivered to clients if the client is subscribed to the channel of the event message.

2.4.8.5. Unsubscribe

1. connected Bayeux client may send unsubscribe messages to cancel interest in a channel and to request that messages published to that channel are not delivered to itself.

2.4.8.6. Disconnect

When a connected client wishes to quit operation it should send a request to the "/meta/disconnect" channel for the server to remove any client-related state.

2.5. RSS

RSS content is saved in eXo-JCR in form of node.

When user add a content node include Id, URL, Description, type.. information of this content is storaged in private path : /Users/%username%/ApplicationData/ContentService/contentNavigation.xml.

Properties of content node are defined in content-nodetypes.xml :

```
<nodeType name="exo:content" isMixin="false" hasOrderableChildNodes="false" primaryItemName="">
    <supertypes>
      <supertype>nt:base</supertype>
    </supertypes>
    <propertyDefinitions>
                          <propertyDefinition name="id" requiredType="String" autoCreated="false" mandatory="fal
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="ownerType" requiredType="String" autoCreated="false" mandato
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="ownerId" requiredType="String" autoCreated="false" mandatory
                                  <valueConstraints/>
                          </propertyDefinition>
                                 <propertyDefinition name="dataType" requiredType="String" autoCreated="false" m
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="data" requiredType="String" autoCreated="false" mandatory="f
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="createdDate" requiredType="Date" autoCreated="false" mandato
                                  <valueConstraints/>
                          </propertyDefinition>
                          <propertyDefinition name="modifiedDate" requiredType="Date" autoCreated="false" mandat
                                  <valueConstraints/>
                          </propertyDefinition>
    </propertyDefinitions>
```

```
  </nodeType>
```

2.5.1. Portlet Integration of eXo API

Import eXo Content Service to your portlet : Insert this code snippet to pom.xml file of your portlet :

```
<dependency>
    <groupId>org.exoplatform.cs</groupId>
    <artifactId>exo.cs.eXoApplication.content.service</artifactId>
    <version>1.3</version>
  </dependency>
```

API Usage

eXo Content Service provides some APIs for working with content as save content, get data...

save content :You can use the following API to save content:

```
public void save(ContentNavigation navigation) throws Exception;
```

navigation : include data to save.

get data :You can use the following API to get data:

```
public ContentData getData(String id) throws Exception;
```

id : unique string to get special data.

2.6. Services Inventory

See ProductsTechOverview:Components_Registry#CS