

Reference Guide / Social Functions

Copyright ©

1. Applications	1
1.1. List of Portlets in Social	1
1.2. List of Gadgets in Social	1
1.2.1. Activity Stream	1
1.2.2. Social RSS Reader	1
1.2.3. My Connections	2
1.2.4. My Spaces	2
2. Configuration	3
2.1. Components	3
2.1.1. ActivityManager	3
2.1.2. SpaceService	3
2.1.3. IdentityManager	3
2.1.4. ProfileConfig	3
2.1.5. ServiceProviderStore	4
2.2. External component plugins	4
2.2.1. MentionsProcessor	4
2.2.2. PortletPreferenceRequiredPlugin	4
2.2.3. SpaceApplicationConfigPlugin	4
2.2.4. AddNodeTypePlugin	5
2.3. RelationshipManager	5
2.4. SpaceIdentityProvider	5
2.5. SpaceApplicationHandler	5
2.6. ExoPeopleService	5
2.7. Space Service	5
2.7.1. Description	5
2.7.2. Components configuration	5
2.7.3. External plug-in configuration	6
2.8. Activity Manager	8
2.8.1. Description	8
2.8.2. Component plug-in configuration	8
2.8.3. External plug-in configuration	9
2.9. Identity Manager	10
2.9.1. Description	10
2.9.2. Component plug-in configuration	10
2.10. OpenSocial Rest Context Configuration	11
2.10.1. Description	11
2.10.2. Component plug-in configuration	11
2.11. Spaces Template configuration	11
2.12. Configure the oauth 2 legged scenario	12
2.12.1. Generate the certificates	13
2.12.2. Configure the property file	13
3. Developers References	14
3.1. UI Extensions	14
3.1.1. About Activity Plugin	14
3.1.2. How to create activity plugin	14
3.2. Overridable Components	26
3.3. Public Java APIs	27
3.3.1. ActivityManager	27
3.3.2. IdentityManager	30
3.3.3. RelationshipManager	32
3.3.4. SpaceService	34
3.4. Java APIs sample code/ tutorial	48

3.4.1. Activity Stream	48
3.4.2. OpenSocial	53
3.4.3. People	55
3.4.4. Spaces	58
3.4.5. Space widget tutorial	61
3.4.6. How to extend the activities rendering	62
3.4.7. XMLProcessor component	64
3.5. Public REST APIs	66
3.5.1. Activities REST service	67
3.5.2. Apps REST service	68
3.5.3. Identity REST service	69
3.5.4. Linkshare REST service	69
3.5.5. People Rest Service	70
3.5.6. Spaces REST service	70
3.5.7. Widget Rest Service	71
3.5.8. Location	72
3.6. Rest Service APIs (v1-alpha3)	72
3.6.1. Activity Resources	73
3.6.2. Activity Stream Resources	83
3.6.3. Identity Resources	93
3.6.4. Version Resources	95
3.7. Public Javascript APIs	97
3.8. Social JCR Structure	97
3.8.1. Overview	97
3.8.2. soc:providers	98
3.8.3. soc:spaces	102

Chapter 1. Applications

1.1. List of Portlets in Social

All the portlets are in the *social-portlet.war* file.

Portlet name	Description
Members Portlet	Enable users to search for Space members or list Space members in the alphabetical order.
My Spaces Portlet	Display the spaces that user is member or manager.
Space Activity Stream Portlet	Share Spaces activities.
Invitations Portlet	List all people that invite users.
Requests Portlet	List all invitations requested by users.
Invitation Spaces Portlet	Display the spaces that user is invited.
Pending Spaces Portlet	Display the Space requests to join page.
Public Spaces Portlet	Display the Public Spaces page.
User Activity Stream Portlet	Update and share the user's activities and/or status.
People Portlet	Display the People page.
Connections Portlet	Display the Connections page.
User Profile Portlet	Display the User profile page.

1.2. List of Gadgets in Social

All eXo Social gadgets are in *opensocial.war*.

1.2.1. Activity Stream

- **Description:** Manage activities of users: update status, like/unlike activities, comment activities, delete activities and delete comments.
- **Used REST services:** ActivitiesRestServices.

1.2.2. Social RSS Reader

- **Description:** Fetch, parse and display RSS from a specific URL.
- **Description of user preferences:** There are 2 preference fields: URL input box (default value is <http://blog.exoplatform.org/feed/>) and Number of RSS per page selector (default value is 10).

- **Used REST services:** N/A.

1.2.3. My Connections

- **Description:** Get and display information of the current viewer and his connections.
- **Description of user preferences:** The number of connections displayed per page. It is set to '5' by default.
- **Used REST services:** N/A.

1.2.4. My Spaces

- **Description:** Display all spaces that a user has the "member" role.
- **Used REST services:** SpacesRestService.

Chapter 2. Configuration

2.1. Components

2.1.1. ActivityManager

Configuration name	Data type	Possible value	Description
allowedTags	String list	b, i, a, span, em, strong, p, ol, ul, li, br, img	HTML tags

2.1.2. SpaceService

Configuration name	Data type	Possible value	Description
space.homeNodeApp	String	SpaceActivityStreamPortlet	The home application for a space.
space.apps	String list	DashboardPortlet:true, SpaceSettingPortlet:false, MembersPortlet:true	The applications that are used for initializing the application when the space is created.



Note

Deprecated: `org.exoplatform.social.core.space.SpaceApplicationConfigPlugin` Use `external-component-plugins` instead:

2.1.3. IdentityManager

Configuration name	Data type	Possible value	Description
providers	String	org.exoplatform.social.identity.provider.SpaceIdentityProvider	The identity providers.

2.1.4. ProfileConfig

Configuration name	Data type	Possible value	Description
nodetype.emails	String	exo:profileKeyValue	
nodetype.phones	String	exo:profileKeyValue	
nodetype.ims	String	exo:profileKeyValue	
nodetype.urls	String	exo:profileKeyValue	

Configuration name	Data type	Possible value	Description
nodetype.address	String	exo:profileAddress	
nodetype.experiences	String	exo:profileExperience	
nodetype.education	String	exo:profileEducation	
forceMultiValue	String	xxxxxxxxxxxxx	

2.1.5. ServiceProviderStore

Configuration name	Data type	Possible value	Description
sample-provider	properties-params		Sample service provider.

2.2. External component plugins

2.2.1. MentionsProcessor

Configuration name	Data type	Possible value	Description
priority	String	2	The priority of this processor (the lower is executed first).

2.2.2. PortletPreferenceRequiredPlugin

Configuration name	Data type	Possible value	Description
portletsPrefsRequired	String list	SpaceActivityStreamPortlet, SpaceSettingPortlet, MembersPortlet	The portlet name which requires space URL preference.

2.2.3. SpaceApplicationConfigPlugin

Configuration name	Data type	Description
spaceHomeApplication	SpaceApplicationConfigPlugin\$SpaceApplication	The space application for the space home node.
spaceApplicationList	SpaceApplicationConfigPlugin	Space application list configuration.

2.2.4. AddNodeTypePlugin

Configuration name	Data type	Possible value	Description
autoCreatedInNewRepository	String	jar:/conf/portal/core-nodetypes	Node types configuration file.

2.3. RelationshipManager

The Service is used to manipulate user relationships.

2.4. SpaceIdentityProvider

The provider is to give identity space instances.

2.5. SpaceApplicationHandler

The service is to handle all events related to creating and managing all the application spaces.

2.6. ExoPeopleService

The service is to manipulate all data related to people in the Portal.

2.7. Space Service

2.7.1. Description

The service for spaces management includes creating spaces, and installing applications.

2.7.2. Components configuration

Component name	Description
SpaceServiceImpl	Implementation class of Space Service.

```
<component>
  <key>org.exoplatform.social.core.space.spi.SpaceService</key>
  <type>org.exoplatform.social.core.space.impl.SpaceServiceImpl</type>
  <!--Deprecated, Use external-component-plugins instead
  <init-params>
    <values-param>
      <name>space.homeNodeApp</name>
      <value>SpaceActivityStreamPortlet</value>
    </values-param>
    <values-param>
```

```

<name>space.apps</name>
<value>DashboardPortlet:true</value>
<value>SpaceSettingPortlet:false</value>
<value>MembersPortlet:true</value>
</values-param>
</init-params>
-->
</component>

```

Configuration name	Data type	Possible value	Possible value	Description
SpaceActivityStreamPortlet		N/A	SpaceActivityStream	The name of portlet displaying activities of spaces.
space.apps	String list	Portlets' name: true/false	DashboardPortlet:true SpaceSettingPortlet:false MembersPortlet:true	The list of configurations for portlets.

2.7.3. External plug-in configuration

2.7.3.1. PortletPreferenceRequiredPlugin

```

<external-component-plugins>
  <target-component>org.exoplatform.social.core.space.spi.SpaceService</target-component>
  <component-plugin>
    <name>portlets.prefs.required</name>
    <set-method>setPortletsPrefsRequired</set-method>
    <type>org.exoplatform.social.core.application.PortletPreferenceRequiredPlugin</type>
    <init-params>
      <values-param>
        <name>portletsPrefsRequired</name>
        <value>SpaceActivityStreamPortlet</value>
        <value>SpaceSettingPortlet</value>
        <value>MembersPortlet</value>
      </values-param>
    </init-params>
  </component-plugin>
</external-component-plugins>

```

In which:

Name	Set-method	Type	Description
PortletPreferenceRequiredPlugin	setPortletsPrefsRequired	org.exoplatform.social.core.application.PortletPreferenceRequiredPlugin	Configure the list of portlet names which will have portlet preference of space context.

- Init-params:

Name	Possible value	Possible value	Description
portletsPrefsRequired	Portlet names	SpaceActivityStreamPortlet; SpaceSettingPortlet; MembersPortlet	List of portlets which need to be saved and get the space context name.

2.7.3.2. SpaceApplicationConfigPlugin



Note

Since 1.2.0-GA

```
<external-component-plugins>
  <target-component>org.exoplatform.social.core.space.spi.SpaceService</target-component>
  <!-- Default applications to be installed when creating a new space -->
  <component-plugin>
    <name>Space Application Configuration</name>
    <set-method>setSpaceApplicationConfigPlugin</set-method>
    <type>org.exoplatform.social.core.space.SpaceApplicationConfigPlugin</type>
    <init-params>
      <object-param>
        <name>spaceHomeApplication</name>
        <description>Space Home Application</description>
        <object type="org.exoplatform.social.core.space.SpaceApplicationConfigPlugin$SpaceApplication">
          <field name="portletApp">
            <string>social-portlet</string>
          </field>
          <field name="portletName">
            <string>SpaceActivityStreamPortlet</string>
          </field>
          <field name="appTitle">
            <string>Home</string>
          </field>
          <!--<field name="icon"><string>SpaceHomeIcon</string></field> -->
        </object>
      </object-param>
      <object-param>
        <name>spaceApplicationListConfig</name>
        <description>space application list configuration</description>
        <object type="org.exoplatform.social.core.space.SpaceApplicationConfigPlugin">
          <field name="spaceApplicationList">
            <collection type="java.util.ArrayList">
              <value>
                <object type="org.exoplatform.social.core.space.SpaceApplicationConfigPlugin$SpaceApplication">
                  <field name="portletApp">
                    <string>dashboard</string>
                  </field>
                  <field name="portletName">
                    <string>DashboardPortlet</string>
                  </field>
                  <field name="appTitle">
                    <string>Dashboard</string>
                  </field>
                  <field name="removable">
                    <boolean>true</boolean>
                  </field>
                  <field name="order">
                    <int>1</int>
                  </field>
                  <field name="uri">
                    <string>dashboard</string>
                  </field>
                  <!--<field name="icon"><string>SpaceDashboardIcon</string></field> -->
                </object>
              </value>
              <value>
                <object type="org.exoplatform.social.core.space.SpaceApplicationConfigPlugin$SpaceApplication">
                  <field name="portletApp">
                    <string>social-portlet</string>
                  </field>
                  <field name="portletName">
                    <string>SpaceSettingPortlet</string>
                  </field>
                  <field name="appTitle">
                    <string>Space Settings</string>
                  </field>
                  <field name="removable">
                    <boolean>false</boolean>
                  </field>
                </object>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

```

        <field name="order">
          <int>2</int>
        </field>
        <field name="uri">
          <string>settings</string>
        </field>
        <!--<field name="icon"><string>SpaceSettingsIcon</string></field> -->
      </object>
    </value>
    <value>
      <object type="org.exoplatform.social.core.space.SpaceApplicationConfigPlugin$SpaceApplication">
        <field name="portletApp">
          <string>social-portlet</string>
        </field>
        <field name="portletName">
          <string>MembersPortlet</string>
        </field>
        <field name="appTitle">
          <string>Members</string>
        </field>
        <field name="removable">
          <boolean>true</boolean>
        </field>
        <field name="order">
          <int>3</int>
        </field>
        <field name="uri">
          <string>members</string>
        </field>
        <!--<field name="icon"><string>SpaceMembersIcon</string></field> -->
      </object>
    </value>
  </collection>
</field>
</object>
</object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

In which:

Name	Set-method	Type	Description
Space Application Configuration	setSpaceApplicationConfigPlugin	org.exoplatform.social.core.space.SpaceApplicationConfigPlugin	Configuration Portlet Preference Required. space applications to be installed when creating a new space.

2.8. Activity Manager

2.8.1. Description

The service is to manipulate space and user activities.

2.8.2. Component plug-in configuration

```

<component-plugin>
  <name>OSHtmlSanitizer</name>
  <set-method>addProcessorPlugin</set-method>
  <type>org.exoplatform.social.core.processor.OSHtmlSanitizerProcessor</type>
  <init-params>
    <values-param>

```

```

<name>allowedTags</name>
<value>b</value>
<value>i</value>
<value>span</value>
<value>a</value>
<value>em</value>
<value>strong</value>
<value>p</value>
<value>ol</value>
<value>ul</value>
<value>li</value>
<value>br</value>
<value>img</value>
</values-param>
</init-params>
</component-plugin>

```

In which,

Name	Set-method	Type	Description
OSHtmlSanitizerProcessor	addProcessorPlugin	org.exoplatform.social.core.processor.OSHtmlSanitizerProcessor	Process and render valid HTML tags appearing in the Activity body (content).

- Init-params:

Name	Possible value	Possible value	Description
allowedTags	html tags	b, i, a, span, em, strong, p, ol, ul, li, br, img	Process and render HTML tags in the activity content (body).

2.8.3. External plug-in configuration

```

<component-plugin>
  <name>MentionsProcessor</name>
  <set-method>addProcessorPlugin</set-method>
  <type>org.exoplatform.social.core.processor.MentionsProcessor</type>
  <init-params>
    <value-param>
      <name>priority</name>
      <description>priority of this processor (lower number will be executed first)</description>
      <value>2</value>
    </value-param>
  </init-params>
</component-plugin>

```

In which:

Name	Set-method	Type	Description
MentionsProcessor	addProcessorPlugin	org.exoplatform.social.core.processor.MentionsProcessor	Process and substitute @username expressions by a link on the user profile.

- Init-params:

Name	Possible value	Possible value	Description
priority	priority number	2	The priority of this processor (The lower level will be executed first).

2.9. Identity Manager

2.9.1. Description

The service is to manipulate the identity operations, such as creating, getting, deleting or finding a profile.

2.9.2. Component plug-in configuration

```
<component>
  <key>org.exoplatform.social.core.manager.IdentityManager</key>
  <type>org.exoplatform.social.core.manager.IdentityManager</type>
  <component-plugins>
    <component-plugin>
      <name>SpaceIdentityProvider plugin</name>
      <set-method>registerIdentityProviders</set-method>
      <type>org.exoplatform.social.core.identity.IdentityProviderPlugin</type>
      <init-params>
        <values-param>
          <name>providers</name>
          <description>Identity Providers</description>
          <value>org.exoplatform.social.core.identity.provider.SpaceIdentityProvider</value>
        </values-param>
      </init-params>
    </component-plugin>
  </component-plugins>
</component>
```

In which:

Name	Set-method	Type	Description
SpaceIdentityProvider plugin	registerIdentityProviders	org.exoplatform.social.core.identity.IdentityProviderPlugin	The plugin that provides identity for a space.

- Init-params:

Name	Possible value	Possible value	Description
providers	Every other identity providers	org.exoplatform.social.identity.provider.SpaceIdentityProvider	IdentityProvider instances for managing identities.

2.10. OpenSocial Rest Context Configuration

2.10.1. Description

The service is used to configure the portal container name when there is an OpenSocial REST API request. By configuring this service, you can make sure that the right portal container is reached. The default portal container is *portal*.

2.10.2. Component plug-in configuration

This should be used when there is a portal container different than the default *portal* one.

```
<external-component-plugins>
  <target-component>org.exoplatform.social.opensocial.auth.RestPortalContainerNameConfig</target-component>
  <component-plugin>
    <name>set portal container name used for REST service</name>
    <set-method>setRestContainerName</set-method>
    <type>org.exoplatform.social.opensocial.auth.RestPortalContainerNameConfig</type>
    <init-params>
      <value-param>
        <name>rest-container-name</name>
        <value>socialdemo</value>
      </value-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

In which:

Name	Set-method	Type	Description
set portal container name used for REST service	setRestContainerName	org.exoplatform.social.opensocial.auth.RestPortalContainerNameConfig	This plug-in RestPortalContainerNameConfig sets the rest portal container name for the OpenSocial REST API request.

- Init-params:

Name	Possible value	Possible value	Description
rest-container-name	any valid portal container name	N/A	Name of the portal container.

2.11. Spaces Template configuration

In eXo Social, you may have two space types (classic and webos spaces).

For the classic space, you can pre-configure the template. You can configure the layout to select where to display the applications (for example, the application's menu on the left, the selected application is displayed on the right, and more).

Here is an example of the configuration file that displays the menu on the left. The application will be inserted in the container with the ID **Application**:

```
<page>
  <owner-type/>
  <owner-id/>
  <name/>
  <container id="SpacePage" template="system:/groovy/portal/webui/container/UITableColumnContainer.gtmpl">
    <container id="Menu" template="system:/groovy/portal/webui/container/UIContainer.gtmpl">
      <portlet-application>
        <portlet>
          <application-ref>space</application-ref>
          <portlet-ref>SpaceMenuPortlet</portlet-ref>
        </portlet>
        <access-permissions>*:/platform/users</access-permissions>
        <show-info-bar>false</show-info-bar>
      </portlet-application>
    </container>
    <container id="Application" template="system:/groovy/portal/webui/container/UIContainer.gtmpl">
    </container>
  </container>
</page>
```

In this example, the outer container contains two inner containers: one container has id as **Menu** for your Menu and another has id as **Application** containing your applications.

If you want to put the menu on the right and the application on the left, you can swap the declared position of these two containers:

```
<page>
  <owner-type/>
  <owner-id/>
  <name/>
  <container id="SpacePage" template="system:/groovy/portal/webui/container/UITableColumnContainer.gtmpl">
    <container id="Application" template="system:/groovy/portal/webui/container/UIContainer.gtmpl">
      <container id="Menu" template="system:/groovy/portal/webui/container/UIContainer.gtmpl">
        <portlet-application>
          <portlet>
            <application-ref>space</application-ref>
            <portlet-ref>SpaceMenuPortlet</portlet-ref>
          </portlet>
          <access-permissions>*:/platform/users</access-permissions>
          <show-info-bar>false</show-info-bar>
        </portlet-application>
      </container>
    </container>
  </container>
</page>
```

In eXo Social standalone, this configuration file is at:

- `$EXO_TOMCAT/webapps/socialdemo/WEB-INF/conf/portal/template/pages/space/page.xml`

In eXo Platform, this configuration file is at:

- `$EXO_TOMCAT/webapps/{portal-name}/WEB-INF/conf/portal/template/pages/space/page.xml`

2.12. Configure the oauth 2 legged scenario

This section is about configuring the oAuth 2 legs scenario in OpenSocial. (Reference: OpenSocial.org)

For more information, visit [2-legged OAuth for the OpenSocial REST API](#).

2.12.1. Generate the certificates

To generate the key:

```
$ openssl req -newkey rsa:1024 -days 365 -nodes -x509 -keyout testkey.pem  
    -out testkey.pem -subj '/CN=mytestkey'  
$ openssl pkcs8 -in testkey.pem -out oauthkey.pem -topk8 -nocrypt -outform PEM
```

2.12.2. Configure the property file

Edit *container.js* and change the following parameter to point to your private key and key name.

```
"gadgets.signingKeyFile" : "oauth.pem",  
"gadgets.signingKeyName" : "oauthKey",
```

Chapter 3. Developers References

3.1. UI Extensions

3.1.1. About Activity Plugin

The activity plugin feature was introduced to allow using the activity composer extension and the custom UI component for displaying activity by its type.

3.1.2. How to create activity plugin

You should have an idea about the UI Extension Framework. If you have already worked with the UI Extension Framework, it is really easy to create an activity plugin. If not, you have chance to work with it now. You should have a look at [UI Extension Framework](#).

3.1.2.1. Create a custom UI component for displaying the activity based on its type



Note

The Project Code is available [here](#).

When an activity is displayed, UIActivityFactory will look for its registered custom activity display by activity's type. If not found, UIDefaultActivity will be called for displaying that activity.

For example, in eXo Social, there is an activity of type "exosocial:spaces" created by SpaceActivityPublisher, but now, you want to display it without own UI component instead of the default one. To do that, follow these steps.

1. Create a sample project:

```
mvn archetype:generate
[INFO] Scanning for projects...
[INFO] Searching repository for plugin with prefix: 'archetype'.
[INFO] artifact org.apache.maven.plugins:maven-archetype-plugin: checking for updates from central
[INFO] snapshot org.apache.maven.plugins:maven-archetype-plugin:2.0-alpha-6-SNAPSHOT: checking for updates from central
[INFO] snapshot org.apache.maven.archetype:maven-archetype:2.0-alpha-6-SNAPSHOT: checking for updates from central
[INFO] -----
[INFO] Building Maven Default Project
[INFO]   task-segment: [archetype:generate] (aggregator-style)
[INFO] -----
[INFO] Preparing archetype:generate
[INFO] No goals needed for project - skipping
[INFO] snapshot org.apache.maven.archetype:archetype-common:2.0-alpha-6-SNAPSHOT: checking for updates from central
[INFO] snapshot org.apache.maven.archetype:archetype-common:2.0-alpha-6-SNAPSHOT: checking for updates from apache
[INFO] [archetype:generate {execution: default-cli}]
[INFO] Generating project in Interactive mode
[INFO] No archetype defined. Using maven-archetype-quickstart (org.apache.maven.archetypes:maven-archetype-quickstart)
Choose archetype:
1: remote -> docbkx-quickstart-archetype (null)
2: remote -> gquery-archetype (null)
3: remote -> gquery-plugin-archetype (null)
//....

285: remote -> wicket-scala-archetype (Basic setup for a project that combines Scala and Wicket,
    depending on the Wicket-Scala project. Includes an example Specs
    test.)
286: remote -> circumflex-archetype (null)
Choose a number: 76: 76
```

```

Choose version:
1: 1.0
2: 1.0-alpha-1
3: 1.0-alpha-2
4: 1.0-alpha-3
5: 1.0-alpha-4
6: 1.1
Choose a number: : 1
Define value for property 'groupId': : org.exoplatform.social.samples
Define value for property 'artifactId': : exo.social.samples.activity-plugin
Define value for property 'version': 1.0-SNAPSHOT: 1.0.0-SNAPSHOT
Define value for property 'package': org.exoplatform.social.samples: org.exoplatform.social.samples.activityPlugin
Confirm properties configuration:
groupId: org.exoplatform.social.samples
artifactId: exo.social.samples.activity-plugin
version: 1.0.0-SNAPSHOT
package: org.exoplatform.social.samples.activityPlugin
Y: y

```

2. Edit the *pom.xml* file as below.

```

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:sch
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.exoplatform.social</groupId>
    <artifactId>social-project</artifactId>
    <version>1.1.0-GA</version>
  </parent>
  <groupId>org.exoplatform.social.samples</groupId>
  <artifactId>exo.social.samples.activity-plugin</artifactId>
  <packaging>jar</packaging>
  <version>1.1.0-GA</version>
  <name>exo.social.samples.activity-plugin</name>
  <build>
    <sourceDirectory>src/main/java</sourceDirectory>
    <outputDirectory>target/classes</outputDirectory>
    <resources>
      <resource>
        <directory>src/main/resources</directory>
        <includes>
          <include>**/*.gtmpl</include>
        </includes>
      </resource>
      <resource>
        <directory>src/main/java</directory>
        <includes>
          <include>**/*.xml</include>
        </includes>
      </resource>
    </resources>
  </build>
  <dependencies>
    <dependency>
      <groupId>org.exoplatform.portal</groupId>
      <artifactId>exo.portal.webui.core</artifactId>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.exoplatform.portal</groupId>
      <artifactId>exo.portal.webui.portal</artifactId>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.exoplatform.social</groupId>
      <artifactId>exo.social.component.core</artifactId>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.exoplatform.social</groupId>
      <artifactId>exo.social.component.webui</artifactId>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.exoplatform.social</groupId>
      <artifactId>exo.social.component.service</artifactId>

```

```

    <scope>provided</scope>
  </dependency>
</dependencies>
</project>

```

To use the custom UI component for displaying its activity, you need a class that must be a subclass of *BaseUIActivity*.

3. Call *UISpaceSimpleActivity*

```

package org.exoplatform.social.samples.activityplugin;

import org.exoplatform.social.webui.activity.BaseUIActivity;
import org.exoplatform.webui.config.annotation.ComponentConfig;
import org.exoplatform.webui.core.lifecycle.UIFormLifecycle;

@ComponentConfig(
    lifecycle = UIFormLifecycle.class,
    template = "classpath:groovy/social/plugin/space/UISpaceSimpleActivity.gtmpl"
)
public class UISpaceSimpleActivity extends BaseUIActivity {
}

```

The *UISpaceSimpleActivity.gtmpl* template should be created under *main/resources/groovy/social/plugin/space*:

```

<div>This is a space activity UI component displayed for type "exosocial:spaces"</div>

```

An activity builder as [explained later](#) is also needed.

```

package org.exoplatform.social.samples.activityplugin;

import org.exoplatform.social.core.activity.model.Activity;
import org.exoplatform.social.webui.activity.BaseUIActivity;
import org.exoplatform.social.webui.activity.BaseUIActivityBuilder;

public class SimpleSpaceUIActivityBuilder extends BaseUIActivityBuilder {

    @Override
    protected void extendUIActivity(BaseUIActivity uiActivity, Activity activity) {
        // TODO Auto-generated method stub
    }
}

```

4. Create the *configuration.xml* file under *conf/portal*:

```

<configuration xmlns="http://www.exoplaform.org/xml/ns/kernel_1_1.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.exoplaform.org/xml/ns/kernel_1_1.xsd http://www.exoplaform.org/xml/ns/kernel_1_1.xsd">
  <external-component-plugins>
    <target-component>org.exoplatform.webui.ext.UIExtensionManager</target-component>
    <component-plugin>
      <name>add.action</name>
      <set-method>registerUIExtensionPlugin</set-method>
      <type>org.exoplatform.webui.ext.UIExtensionPlugin</type>
      <init-params>
        <object-param>
          <name>Simple Space Activity</name>
          <object type="org.exoplatform.social.webui.activity.UIActivityExtension">
            <field name="type">
              <string>org.exoplatform.social.webui.activity.BaseUIActivity</string>
            </field>
            <field name="name">
              <string>exosocial:spaces</string>
            </field>
          </object>
        </object-param>
      </init-params>
    </component-plugin>
  </external-component-plugins>
</configuration>

```

```

    <field name="component">
      <string>org.exoplatform.social.samples.activityplugin.UISpaceSimpleActivity</string>
    </field>
    <field name="activityBuiderClass">
      <string>org.exoplatform.social.samples.activityplugin.SimpleSpaceUIActivityBuilder</string>
    </field>
  </object>
</object-param>
</init-params>
</component-plugin>
</external-component-plugins>
</configuration>

```

Note that *exosocial:spaces* must have its value matching with the activity's type that you want to display with your UI component.

Assume that you have already built the Social project with version 1.1. If you do not know how to, have a look at [Building from Sources](#) with [Social 1.1.0-CR01](#). Next, build a sample project and copy the jar file to /tomcat/lib. Then, run Social, create a space and access it, you can see the space's activity of type "exosocial:spaces" is displayed by default in Social:

The custom UI component for displaying activity of type "exosocial:spaces" is like below:

5. Make the custom UI activity display have the look, feel and function like the default one.

When displaying an activity, you should make sure that the look and feel of the custom UI component is consistent and match with other activities and have the full functions of **Like**, **Comments**. To create another UI component to display, call *UISpaceLookAndFeelActivity*:

```

package org.exoplatform.social.samples.activityplugin;

import org.exoplatform.social.webui.activity.BaseUIActivity;
import org.exoplatform.webui.config.annotation.ComponentConfig;
import org.exoplatform.webui.config.annotation.EventConfig;
import org.exoplatform.webui.core.lifecycle.UIFormLifecycle;

@ComponentConfig(
  lifecycle = UIFormLifecycle.class,
  template = "classpath:groovy/social/plugin/space/UISpaceLookAndFeelActivity.gtmpl",
  events = {
    @EventConfig(listeners = BaseUIActivity.ToggleDisplayLikesActionListener.class),
    @EventConfig(listeners = BaseUIActivity.ToggleDisplayCommentFormActionListener.class),
    @EventConfig(listeners = BaseUIActivity.LikeActivityActionListener.class),
    @EventConfig(listeners = BaseUIActivity.SetCommentListStatusActionListener.class),
    @EventConfig(listeners = BaseUIActivity.PostCommentActionListener.class),
    @EventConfig(listeners = BaseUIActivity.DeleteActivityActionListener.class, confirm = "UIActivity.msg.Are_You_Sure_Delete_Activity"),
    @EventConfig(listeners = BaseUIActivity.DeleteCommentActionListener.class, confirm = "UIActivity.msg.Are_You_Sure_Delete_Comment")
  }
)
public class UISpaceLookAndFeelActivity extends BaseUIActivity {
}

```

6. Create the *UISpaceLookAndFeelActivity* template. The easiest way is to copy the content of *UIDefaultActivity.gtmpl* to this template file:

```

<%
import org.exoplatform.portal.webui.util.Util;
import org.exoplatform.webui.form.UIFormTextAreaInput;

def pcontext = Util.getPortalRequestContext();
def jsManager = pcontext.getJavascriptManager();
def labelActivityHasBeenDeleted = _ctx.appRes("UIActivity.label.Activity_Has_Been_Deleted");
def activity = uicomponent.getActivity();

```

```

def activityDeletable = uicomponent.isActivityDeletable();
%>

<% if (activity) { //process if not null

jsManager.importJavascript("eXo.social.Util", "/social-resources/javascript");
jsManager.importJavascript("eXo.social.PortalHttpRequest", "/social-resources/javascript");
jsManager.importJavascript("eXo.social.webui.UIForm", "/social-resources/javascript");
jsManager.importJavascript("eXo.social.webui.UIActivity", "/social-resources/javascript");

def labelComment = _ctx.appRes("UIActivity.label.Comment");
def labelLike = _ctx.appRes("UIActivity.label.Like");
def labelUnlike = _ctx.appRes("UIActivity.label.Unlike");
def labelSource = _ctx.appRes("UIActivity.label.Source");
def inputWriteAComment = _ctx.appRes("UIActivity.input.Write_A_Comment");
def labelShowAllComments = _ctx.appRes("UIActivity.label.Show_All_Comments");
def labelHideAllComments = _ctx.appRes("UIActivity.label.Hide_All_Comments");
def labelOnePersonLikeThis = _ctx.appRes("UIActivity.label.One_Person_Like_This");
def labelPeopleLikeThis = _ctx.appRes("UIActivity.label.People_Like_This");
def labelYouLikeThis = _ctx.appRes("UIActivity.label.You_Like_This");
def labelYouAndOnePersonLikeThis = _ctx.appRes("UIActivity.label.You_And_One_Person_Like_This");
def labelYouAndPeopleLikeThis = _ctx.appRes("UIActivity.label.You_And_People_Like_This");

def likeActivityAction = uicomponent.event("LikeActivity", "true");
def unlikeActivityAction = uicomponent.event("LikeActivity", "false");

def commentList = uicomponent.getComments();
def allComments = uicomponent.getAllComments();
if (allComments) {
    labelShowAllComments = labelShowAllComments.replace("{0}", allComments.size() + "");
    labelHideAllComments = labelHideAllComments.replace("{0}", allComments.size() + "");
}
def displayedIdentityLikes = uicomponent.getDisplayedIdentityLikes();
def identityLikesNum = 0;
def labelLikes = null;
def toggleDisplayLikesAction = uicomponent.event("ToggleDisplayLikes");
def startTag = "<a onclick={{{{{}}} $toggleDisplayLikesAction{{{}}} id={{{{{}}} ToggleDisplayListPeopleLikes${act
def endTag = "</a>";
if (displayedIdentityLikes != null) {
    identityLikesNum = displayedIdentityLikes.length;
}
def commentListStatus = uicomponent.getCommentListStatus();
def commentFormDisplayed = uicomponent.isCommentFormDisplayed();
def likesDisplayed = uicomponent.isLikesDisplayed();
//params for init UIActivity javascript object
def params = ""
    {activityId: '${activity.id}',
      inputWriteAComment: '$inputWriteAComment',
      commentMinCharactersAllowed: ${uicomponent.getCommentMinCharactersAllowed()},
      commentMaxCharactersAllowed: ${uicomponent.getCommentMaxCharactersAllowed()},
      commentFormDisplayed: $commentFormDisplayed,
      commentFormFocused: ${uicomponent.isCommentFormFocused()}
    }
""
jsManager.addOnLoadJavascript("initUIActivity${activity.id}");
//make sures commentFormFocused is set to false to prevent any refresh to focus, only focus after post a comme
uicomponent.setCommentFormFocused(false);
def activityUserName, activityUserProfileUri, activityImageSource, activityContentBody, activityPostedTime;
def commentFormBlockClass = "", listPeopleLikeBlockClass = "", listPeopleBGClass = "";
if (!commentFormDisplayed) {
    commentFormBlockClass = "DisplayNone";
}

if (!likesDisplayed) {
    listPeopleLikeBlockClass = "DisplayNone";
}

if (uicomponent.isLiked()) {
    if (identityLikesNum > 1) {
        labelLikes = labelYouAndPeopleLikeThis.replace("{start}", startTag).replace("{end}", endTag).replace("{0}",
    } else if (identityLikesNum == 1) {
        labelLikes = labelYouAndOnePersonLikeThis.replace("{start}", startTag).replace("{end}", endTag);
    } else {
        labelLikes = labelYouLikeThis;
    }
} else {
    if (identityLikesNum > 1) {
        labelLikes = labelPeopleLikeThis.replace("{start}", startTag).replace("{end}", endTag).replace("{0}", id

```

```

    } else if (identityLikesNum == 1) {
        labelLikes = labelOnePersonLikeThis.replace("{start}", startTag).replace("{end}", endTag);
    }
}

if (!labelLikes) {
    //hides displayPeopleBG
    listPeopleBGClass = "DisplayNone";
}

activityContentTitle = activity.title;
activityPostedTime = uicomponent.getPostedTimeString(activity.postedTime);
activityUserName = uicomponent.getUserFullName(activity.userId);
activityUserProfileUri = uicomponent.getUserProfileUri(activity.userId);

activityImageSource = uicomponent.getUserAvatarImageSource(activity.userId);
if (!activityImageSource) {
    activityImageSource = "/social-resources/skin/ShareImages/SpaceImages/SpaceLogoDefault_61x61.gif";
}

%>

<div class="UIActivity">
    <script type="text/javascript">
        function initUIActivity${activity.id}() {
            new exo.social.webui.UIActivity($params);
        }
    </script>

    <% uiForm.begin() %>
    <div class="NormalBox clearfix">
        <a class="Avatar" title="${activityUserName}" href="${activityUserProfileUri}">
            
        </a>
        <div class="ContentBox" id="ContextBox${activity.id}">
            <div class="TitleContent clearfix">
                <div class="Text">
                    <a title="${activityUserName}" href="${activityUserProfileUri}">${activityUserName}</a>
                </div>
                <% if (activityDeletable) %>
                <div onclick="${uicomponent.event("DeleteActivity", uicomponent.getId(), "")}" %> class="CloseContent" %>
                <%}%>
                </div>
                <div class="Content">
                    ${activityContentTitle} (from custom UI component)<br>
                </div>
                <div class="LinkCM">
                    <span class="DateTime">${activityPostedTime} *</span>
                    <% def toggleDisplayCommentAction = uicomponent.event('ToggleDisplayCommentForm', null, false);
                    def commentLink = "";
                    %>
                    <a class="LinkCM ${commentLink}" onclick="${toggleDisplayCommentAction}" id="CommentLink${activity.id}" href="${commentLink}">
                        ${labelComment}
                    </a> |
                    <% if (uicomponent.isLiked()) { %>
                    <a onclick="${unlikeActivityAction}" class="LinkCM" id="UnLikeLink${activity.id}" href="#unlike">
                        ${labelUnlike}
                    </a>
                    <% } else { %>
                    <a onclick="${likeActivityAction}" class="LinkCM" id="LikeLink${activity.id}" href="#like">
                        ${labelLike}
                    </a>
                    <%}%>
                    </div>
                </div>

            <div class="ListPeopleLikeBG ${listPeopleBGClass}">
                <div class="ListPeopleLike">
                    <div class="ListPeopleContent">
                        <% if (!labelLikes) labelLikes = ""; %>
                        <div class="Title">${labelLikes}</div>
                        <div class="${listPeopleLikeBlockClass}">
                            <%
                            //def personLikeFullName, personLikeProfileUri, personLikeAvatarImageSource;

                            displayedIdentityLikes.each({
                                personLikeFullName = uicomponent.getUserFullName(it);
                                personLikeProfileUri = uicomponent.getUserProfileUri(it);
                                personLikeAvatarImageSource = uicomponent.getUserAvatarImageSource(it);

```

```

        if (!personLikeAvatarImageSource) {
            personLikeAvatarImageSource = "/social-resources/skin/ShareImages/activity/AvatarPeople.gif";
        }
        %>
        <a class="AvatarPeopleBG" title="$personLikeFullName" href="$personLikeProfileUri">
            
        <% } %>
    </div>
    <div class="ClearLeft">
        <span></span>
    </div>
</div>
</div>
</div>

<div class="CommentListInfo">
    <% if (uicomponent.commentListToggleable()) {
    def showAllCommentsAction = uicomponent.event("SetCommentListStatus", "all");
    def hideAllCommentsAction = uicomponent.event("SetCommentListStatus", "none");
    %>
    <div class="CommentBlock">
        <div class="CommentContent">
            <div class="CommentBorder">
                <% if (commentListStatus.getStatus().equals("latest") || commentListStatus.getStatus().equals("none"))
                <a onclick="$showAllCommentsAction" href="#show-all-comments">
                    $labelShowAllComments
                </a>
                <% } else if (commentListStatus.getStatus().equals("all")) { %>
                <a onclick="$hideAllCommentsAction" href="#hide-all-comments">
                    $labelHideAllComments
                </a>
                <% } %>
            </div>
        </div>
    </div>
    <% } %>
</div>
<% if (allComments.size() > 0) { %>
    <div class="DownIconCM"><span></span></div>
    <% }%>

    <%
    def commenterFullName, commenterProfileUri, commenterImageSource, commentMessage, commentPostedTime;
    def first = true, commentContentClass;
    commentList.each({
        if (first & !uicomponent.commentListToggleable()) {
            commentContentClass = "CommentContent";
            first = false;
        } else {
            commentContentClass = "";
        }
        commenterFullName = uicomponent.getUserFullName(it.userId);
        commenterProfileUri = uicomponent.getUserProfileUri(it.userId);
        commenterImageSource = uicomponent.getUserAvatarImageSource(it.userId);
        if (!commenterImageSource) {
            commenterImageSource = "/social-resources/skin/ShareImages/activity/AvatarPeople.gif";
        }
        commentMessage = it.title;
        commentPostedTime = uicomponent.getPostedTimeString(it.postedTime);
    %>
    <div id="CommentBlock${activity.id}" class="CommentBox clearfix">
        <a class="AvatarCM" title="$commenterFullName" href="$commenterProfileUri">
            
        </a>
        <div class="ContentBox">
            <div class="Content">
                <a href="$commenterProfileUri"><span class="Commenter">$commenterFullName</span></a><br />
                $commentMessage
            </div>
            <div class="LinkCM">
                <span class="DateCM">$commentPostedTime</span>
            </div>
        </div>
        <%
        if (uicomponent.isCommentDeletable(it.userId)) {
        %>
        <div onclick="<%= uicomponent.event("DeleteComment", uicomponent.id, it.id); %>" class="CloseCMContentHili

```

```

<% } %>
</div>
<% }) %>

<div class="CommentBox $commentFormBlockClass clearfix" id="CommentFormBlock${activity.id}">
  <% uicomponent.renderChild(UIFormTextAreaInput.class); %>
  <input type="button" onclick="<%= uicomponent.event("PostComment") %>" value="$labelComment" class="Commen
</div>

</div>
<% uiiform.end() %>
</div>
<% } else { %> <!-- activity deleted -->
<div class="UIActivity Deleted">$labelActivityHasBeenDeleted</div>
<% }%>

```

And you should make needed modifications for this template:

```

<div class="Content">
  $activityContentTitle (from custom UI component)<br>
</div>

```

7. Reconfigure the *configuration.xml* file:

```

<configuration xmlns="http://www.exoplaform.org/xml/ns/kernel_1_1.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema"
  <external-component-plugins>
    <target-component>org.exoplaform.webui.ext.UIExtensionManager</target-component>
    <component-plugin>
      <name>add.action</name>
      <set-method>registerUIExtensionPlugin</set-method>
      <type>org.exoplaform.webui.ext.UIExtensionPlugin</type>
      <init-params>
        <object-param>
          <name>Look And Feel Space Activity</name>
          <object type="org.exoplaform.social.webui.activity.UIActivityExtension">
            <field name="type">
              <string>org.exoplaform.social.webui.activity.BaseUIActivity</string>
            </field>
            <field name="name">
              <string>exosocial:spaces</string>
            </field>
            <field name="component">
              <string>org.exoplaform.social.samples.activityplugin.UISpaceLookAndFeelActivity</string>
            </field>
            <field name="activityBuiderClass">
              <string>org.exoplaform.social.samples.activityplugin.SimpleSpaceUIActivityBuilder</string>
            </field>
          </object>
        </object-param>
      </init-params>
    </component-plugin>
  </external-component-plugins>
</configuration>

```

8. Rebuild the sample project, copy the **.jar** file to tomcat/lib. Run the server again and see the result:



Note

Currently, you have to copy and paste in the template file. By this way, you have full control of the UI but it is not a good way when there are changes in `UIDefaultActivity`. This will be improved soon so that no copy/paste is needed.

3.1.2.1.1. What is ActivityBuilder?

ActivityBuilder is one class which is used to get values of *ExoSocialActivity* to set to *UIActivity* for displaying. eXo Social provides the *BaseUIActivityBuilder* class for developers to extend and customize their own activity builder easily and properly.

For example, to write your own *UILinkActivityBuilder*, you just need to extend *BaseUIActivityBuilder* and then customize attributes and behaviors of the activity builder as below.

```
public class UILinkActivityBuilder extends BaseUIActivityBuilder {
    private static final Log LOG = ExoLogger.getLogger(UILinkActivityBuilder.class);
    @Override
    protected void extendUIActivity(BaseUIActivity uiActivity, ExoSocialActivity activity) {
        UILinkActivity uiLinkActivity = (UILinkActivity) uiActivity;
        Map<String, String> templateParams = activity.getTemplateParams();
        uiLinkActivity.setLinkSource(templateParams.get(UILinkActivityComposer.LINK_PARAM));
        uiLinkActivity.setLinkTitle(templateParams.get(UILinkActivityComposer.TITLE_PARAM));
        uiLinkActivity.setLinkImage(templateParams.get(UILinkActivityComposer.IMAGE_PARAM));
        uiLinkActivity.setLinkDescription(templateParams.get(UILinkActivityComposer.DESCRPTION_PARAM));
        uiLinkActivity.setLinkComment(templateParams.get(UILinkActivityComposer.COMMENT_PARAM));
    }
}
```



Note

To learn more about ActivityBuilder, refer to the [BaseUIActivity class](#).

3.1.2.2. Create a composer extension for composing activity on the UI composer and display it on the activity stream

The *UIActivityComposer* is an extended class of *UIContainer* that is used to display inputs for users to create their own activities. To write your own activity composer, it is recommended that you use the *UIActivityComposer* available in eXo Social.

For example, to create an input component for inserting video links into your activity, do the following steps:

1. Write *UIVideoActivityComposer* which extends *UIActivityComposer*. The *UIActivityComposer* allows you to input extended activities (for example, adding videos, links or documents) on the UI composer.

```
package org.exoplatform.social.plugin.videolink;

import org.exoplatform.social.plugin.videolink.util.VideoEmbedTool;
@ComponentConfig(
    template = "classpath:groovy/social/plugin/videolink/UIVideoActivityComposer.gtmpl",
    events = {
        @EventConfig(listeners = UIVideoActivityComposer.SearchVideo.class),
        @EventConfig(listeners = UIVideoActivityComposer.SelectVideoFromResultList.class),
        @EventConfig(listeners = UIVideoActivityComposer.AttachActionListener.class),
        @EventConfig(listeners = UIVideoActivityComposer.ChangeLinkContentActionListener.class),
        @EventConfig(listeners = UIActivityComposer.CloseActionListener.class),
        @EventConfig(listeners = UIActivityComposer.SubmitContentActionListener.class),
        @EventConfig(listeners = UIActivityComposer.ActivateActionListener.class)
    }
)

public class UIVideoActivityComposer extends UIActivityComposer {

    public static final String LINK_PARAM = "link";
    public static final String IMAGE_PARAM = "image";
    public static final String TITLE_PARAM = "title";
    public static final String HTML_PARAM = "htmlembed";
    public static final String COMMENT_PARAM = "comment";

    private static final String HTTP = "http://";
    private static final String HTTPS = "https://";
    private JSONObject videoJson;
```

```

private boolean linkInfoDisplayed_ = false;
private Map<String, String> templateParams;

/**
 * The constructor.
 */
public UIVideoActivityComposer() {
    setReadyForPostingActivity(false);
    addChild(new UIFormStringInput("InputLink", "InputLink", null));
}

/**
 * Set the link info to be displayed.
 *
 * @param displayed
 */
public void setLinkInfoDisplayed(boolean displayed) {
    linkInfoDisplayed_ = displayed;
}

/**
 * Set the template params.
 *
 * @param templateParams
 */
public void setTemplateParams(Map<String, String> templateParams) {
    this.templateParams = templateParams;
}

/**
 * Get the template params.
 */
public Map<String, String> getTemplateParams() {
    return templateParams;
}

/**
 * Clear the video json.
 */
public void clearVideoJson() {
    videoJson = null;
}

/**
 * Get the video json.
 */
public JSONObject getVideoJson() {
    return videoJson;
}

/**
 * Set the link.
 *
 * @param url
 * @throws Exception
 */
private void setLink(String url) throws Exception {
    if (!(url.contains(HTTP) || url.contains(HTTPS))) {
        url = HTTP + url;
    }

    videoJson = VideoEmbedTool.getoembedData(url);
    templateParams = new HashMap<String, String>();
    templateParams.put(LINK_PARAM, url);
    templateParams.put(TITLE_PARAM, videoJson.getString(VideoEmbedTool.OEMBED_TITLE));
    templateParams.put(HTML_PARAM, videoJson.getString(VideoEmbedTool.OEMBED_HTML));
    setLinkInfoDisplayed(true);
}

static public class AttachActionListener extends EventListener<UIVideoActivityComposer> {

    @Override
    public void execute(Event<UIVideoActivityComposer> event) throws Exception {
        WebuiRequestContext requestContext = event.getRequestContext();
        UIVideoActivityComposer uiComposerLinkExtension = event.getSource();
        String url = requestContext.getRequestParameter(OBJECTID);
        try {
            uiComposerLinkExtension.setLink(url.trim());
        } catch (Exception e) {
            uiComposerLinkExtension.setReadyForPostingActivity(false);
        }
    }
}

```

```

        return;
    }
    requestContext.addComponentToUpdateByAjax(uiComposerLinkExtension);
    event.getSource().setReadyForPostingActivity(true);
}

static public class ChangeLinkContentActionListener extends EventListener<UIVideoActivityComposer>{
    @Override
    public void execute(Event<UIVideoActivityComposer> event) throws Exception {
        WebuiRequestContext requestContext = event.getRequestContext();
        UIVideoActivityComposer uiComposerLinkExtension = event.getSource();

        Map<String, String> tempParams = new HashMap<String, String>();

        uiComposerLinkExtension.setTemplateParams(tempParams);
        requestContext.addComponentToUpdateByAjax(uiComposerLinkExtension);
        UIComponent uiParent = uiComposerLinkExtension.getParent();
        if (uiParent != null) {
            uiParent.broadcast(event, event.getExecutionPhase());
        }
    }
}

public static class SelectVideoFromResultList extends EventListener<UIVideoActivityComposer>{
    @Override
    public void execute(Event<UIVideoActivityComposer> event) throws Exception {
        WebuiRequestContext requestContext = event.getRequestContext();
        UIVideoActivityComposer uiComposerLinkExtension = event.getSource();
    }
}

public static class SearchVideo extends EventListener<UIVideoActivityComposer>{
    @Override
    public void execute(Event<UIVideoActivityComposer> event) throws Exception {
        WebuiRequestContext requestContext = event.getRequestContext();
        UIVideoActivityComposer uiComposerLinkExtension = event.getSource();
    }
}

@Override
public void onPostActivity(PostContext postContext, UIComponent source,
        WebuiRequestContext requestContext, String postedMessage) throws Exception {
    templateParams.put(COMMENT_PARAM, postedMessage);
    setTemplateParams(templateParams);
    if (templateParams.size() == 0) {
        uiApplication.addMessage(new ApplicationMessage("UIComposer.msg.error.Empty_Message",
            null,
            ApplicationMessage.WARNING));
        return;
    }
    String title = "Shared a video: <a href={{{{}}}{{}}}" + LINK_PARAM + "}}}}>${" + TITLE_PARAM +
    ExoSocialActivity activity = new ExoSocialActivityImpl(userIdentity.getId(),
        UIVideoActivity.ACTIVITY_TYPE,
        title,
        null);
    activity.setTemplateParams(templateParams);

    if (postContext == UIComposer.PostContext.SPACE) {
        UIActivitiesContainer activitiesContainer = uiDisplaySpaceActivities.getActivitiesLoader().get
        activitiesContainer.addActivity(activity);
        requestContext.addComponentToUpdateByAjax(activitiesContainer);
        requestContext.addComponentToUpdateByAjax(uiComposer);
    } else if (postContext == PostContext.USER) {
        UIUserActivitiesDisplay uiUserActivitiesDisplay = (UIUserActivitiesDisplay) getActivityDisplay
        String ownerName = uiUserActivitiesDisplay.getOwnerName();
        Identity ownerIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME
            ownerName, false);

        activityManager.saveActivity(ownerIdentity, activity);

        if (uiUserActivitiesDisplay.getSelectedDisplayMode() == UIUserActivitiesDisplay.DisplayMode.MY
            UIActivitiesContainer activitiesContainer = uiUserActivitiesDisplay.getActivitiesLoader().ge

```

```

        if (activitiesContainer.getChildren().size() == 1) {
            uiUserActivitiesDisplay.setSelectedDisplayMode(UIUserActivitiesDisplay.DisplayMode.MY_STATUS);
        } else {
            activitiesContainer.addActivity(activity);
            requestContext.addUIComponentToUpdateByAjax(activitiesContainer);
            requestContext.addUIComponentToUpdateByAjax(uiComposer);
        }
    } else {
        uiUserActivitiesDisplay.setSelectedDisplayMode(UIUserActivitiesDisplay.DisplayMode.MY_STATUS);
    }
}
}
}
}

```

2. Use the BaseUIActivity class to write and customize the UIActivity display as below:

```

package org.exoplatform.social.plugin.videolink;
import org.exoplatform.social.webui.activity.BaseUIActivity;
import org.exoplatform.webui.config.annotation.ComponentConfig;
import org.exoplatform.webui.core.lifecycle.UIFormLifecycle;
import org.exoplatform.webui.config.annotation.EventConfig;
@ComponentConfig(lifecycle = UIFormLifecycle.class, template = "classpath:groovy/social/plugin/videolink/UIVideoActivity.groovy")
@EventConfig(listeners = BaseUIActivity.ToggleDisplayLikesActionListener.class),
@EventConfig(listeners = BaseUIActivity.ToggleDisplayCommentFormActionListener.class),
@EventConfig(listeners = BaseUIActivity.LikeActivityActionListener.class),
@EventConfig(listeners = BaseUIActivity.SetCommentListStatusActionListener.class),
@EventConfig(listeners = BaseUIActivity.PostCommentActionListener.class),
@EventConfig(listeners = BaseUIActivity.DeleteActivityActionListener.class, confirm = "UIActivity.msg.Are_You_Sure_Delete_Activity"),
@EventConfig(listeners = BaseUIActivity.DeleteCommentActionListener.class, confirm = "UIActivity.msg.Are_You_Sure_Delete_Comment")
)
public class UIVideoActivity extends BaseUIActivity {
    public static final String ACTIVITY_TYPE = "VIDEO_ACTIVITY";
    private String linkSource = "";
    private String linkTitle = "";
    private String linkHTML = "";
    private String linkComment = "";

    /**
     * Get the link comment.
     */
    public String getLinkComment() {
        return linkComment;
    }

    /**
     * Set the link comment.
     *
     * @param linkComment
     */
    public void setLinkComment(String linkComment) {
        this.linkComment = linkComment;
    }

    /**
     * Get the link html.
     */
    public String getLinkHTML() {
        return linkHTML;
    }

    /**
     * Set the link html.
     *
     * @param linkHTML
     */
    public void setLinkHTML(String linkHTML) {
        this.linkHTML = linkHTML;
    }

    /**
     * Get the link source.
     */
    public String getLinkSource() {
        return linkSource;
    }
}

```

```

/**
 * Set the link source.
 *
 * @param linkSource
 */
public void setLinkSource(String linkSource) {
    this.linkSource = linkSource;
}

/**
 * Get the link title.
 */
public String getLinkTitle() {
    return linkTitle;
}

/**
 * Set the link title.
 *
 * @param linkTitle
 */
public void setLinkTitle(String linkTitle) {
    this.linkTitle = linkTitle;
}
}

```

3. Use the *UIVideoActivityBuilder* class to get values of *ExoSocialActivity* that are set to *UIVideoActivity* for displaying.

```

package org.exoplatform.social.plugin.videolink;
import java.util.Map;
import org.exoplatform.social.core.activity.model.ExoSocialActivity;

public class UIVideoActivityBuilder extends BaseUIActivityBuilder {
    private static final Log LOG = ExoLogger.getLogger(UIVideoActivityBuilder.class);
    @Override
    protected void extendUIActivity(BaseUIActivity uiActivity, ExoSocialActivity activity) {
        UIVideoActivity uiVideoActivity = (UIVideoActivity) uiActivity;
        Map<String, String> templateParams = activity.getTemplateParams();
        uiVideoActivity.setLinkSource(templateParams.get(UIVideoActivityComposer.LINK_PARAM));
        uiVideoActivity.setLinkTitle(templateParams.get(UIVideoActivityComposer.TITLE_PARAM));
        uiVideoActivity.setLinkImage(templateParams.get(UIVideoActivityComposer.IMAGE_PARAM));
        uiVideoActivity.setLinkHTML(templateParams.get(UIVideoActivityComposer.HTML_PARAM));
        uiVideoActivity.setLinkComment(templateParams.get(UIVideoActivityComposer.COMMENT_PARAM));
    }
}

```



Note

You can check out the [source code](#) to get more details.

3.2. Overridable Components

There are 2 components in eXo Social that can be overridden: Space Application Handler & Space Service.

- **Space Application Handler**

```

<component>
  <key>org.exoplatform.social.core.space.spi.SpaceApplicationHandler</key>
  <type>org.exoplatform.social.core.space.impl.DefaultSpaceApplicationHandler</type>
</component>

```

• Space Service

```
<component>
  <key>org.exoplatform.social.core.space.spi.SpaceService</key>
  <type>org.exoplatform.social.core.space.impl.SpaceServiceImpl</type>
  <init-params>
    <!-- Configure the applications to install in a space -->
    <values-param>
      <name>space.homeNodeApp</name>
      <value>SpaceActivityStreamPortlet</value>
    </values-param>
    <!-- Configure removable application or not <value>Application:removable</value> -->
    <values-param>
      <name>space.apps</name>
      <value>DashboardPortlet:true</value>
      <value>SpaceSettingPortlet:false</value>
      <value>MembersPortlet:true</value>
    </values-param>
  </init-params>
</component>
```

3.3. Public Java APIs

3.3.1. ActivityManager

Method	Param	Return	Description
saveActivity (Identity owner, ExoSocialActivity activity) throws ActivityStorageException	owner - the owner of activity stream, activity - the activity which needs to be saved	ExoSocialActivity	Save an activity to the stream of an owner. Note that the Activity.userId will be set to the owner's identity if it has not been already set.
getActivity (String activityId) throws ActivityStorageException	activityId - the id of activity	ExoSocialActivity	Get an activity by its id.
deleteActivity (String activityId) throws ActivityStorageException	activityId - the id of activity	void	Delete an activity by its id.
deleteActivity (ExoSocialActivity activity) throws ActivityStorageException	activity	void	Delete a stored activity (id != null). (Since 1.1.1).
deleteComment (String activityId, String commentId) throws ActivityStorageException	activityId - the id of activity, commentId - the id of comment	void	Delete a comment by its id.
getActivities (Identity identity) throws ActivityStorageException	identity	List<ExoSocialActivity>	Get the latest activities by an identity with the default limit of 20 latest activities.
getActivities (Identity identity, start, limit)	identity, start, limit	List<ExoSocialActivity>	Get the latest activities by

Method	Param	Return	Description
identity, long start, long limit) throws ActivityStorageException			an identity, specifying start that is an offset index and limit .
getActivitiesOfConnections (Identity ownerIdentity) throws ActivityStorageException	Identity	List<ExoSocialActivity>	Get activities of connections from an identity. The activities are returned as a list that is sorted descending by activity posted time. (Since 1.1.1).
getActivitiesOfConnections (Identity ownerIdentity, int offset, int limit) throws ActivityStorageException;	Identity, offset, limit	List<ExoSocialActivity>	Get the activities of connections from an identity by specifying offset and limit. The activities are returned as a list that is sorted starting from the most recent activity.(Since 1.2.0-GA).
getActivitiesOfUserSpaces (Identity ownerIdentity)	Identity	List<ExoSocialActivity>	Get the activities from all spaces of a user. By default, the activity list is composed of all spaces' activities. Each activity list of the space contains maximum 20 activities and are sorted by time. (Since 1.1.1).
getActivityFeed (Identity identity) throws ActivityStorageException	identity	List<ExoSocialActivity>	Get the activity feed of an identity. This feed is the combination of all the activities of his own activities, his connections' activities and spaces' activities which are returned as a list that is sorted starting from the most recent activity.(Since 1.1.2).
saveActivity (ExoSocialActivity activity) throws ActivityStorageException	activity - the activity to save	ExoSocialActivity	Save an activity into the stream for the activity's userId. The userId must be set and this field is used to indicate the owner stream.
saveComment (ExoSocialActivity activity, ExoSocialActivity	activity, comment	void	Save a new comment or updates an existing comment that is an

Method	Param	Return	Description
comment) throws ActivityStorageException			instance of activity with mandatory fields: userId, title.
saveLike (ExoSocialActivity activity, Identity identity) throws ActivityStorageException	activity, identity	void	Save an identity who likes an activity.
removeLike (ExoSocialActivity activity, Identity identity) throws ActivityStorageException	activity, identity - a user who dislikes an activity	void	Remove an identity who likes an activity, if this activity is liked, it will be removed.
getComments (ExoSocialActivity activity) throws ActivityStorageException	activity	List<ExoSocialActivity>	Get the comment list of an activity.
recordActivity (Identity owner, String type, String title) throws ActivityStorageException	owner, type, title	ExoSocialActivity	Record an activity. (Since 1.2.0-GA).
recordActivity (Identity owner, ExoSocialActivity activity) throws Exception	owner, activity	ExoSocialActivity	Save an activity. You should use <code>ActivityManager#saveActivity(org.exoplatform.social.core.activity.model.ExoSocialActivity)</code> instead. It will be removed in eXo Social 1.3.x.
recordActivity (Identity owner, String type, String title, String body) throws ActivityStorageException	owner - the owner of the target stream for this activity, type - the type of an activity which will be used to render a custom UI, title - the title, body - the body	ExoSocialActivity	Record an activity.
addProcessor (ActivityProcessor processor)	processor	void	Add a new processor.
addProcessorPlugin (BaseActivityProcessorPlugin plugin)	plugin	void	Add a new processor plugin.
getActivitiesCount (Identity owner) throws ActivityStorageException	owner	int	Get the number of activities from a stream owner.
processActivity (ExoSocialActivity activity)	activity	void	Pass an activity through the chain of processors.

3.3.2. IdentityManager

Method	Param	Return	Description
registerIdentityProviders (IdentityProviderPlugin plugin)	IdentityProviderPlugin	void	Register one or more IdentityProvider through an IdentityProviderPlugin.
getIdentity (String id)	id can be a social GlobalId or a raw identity such as in Identity.getId()	Identity - null if nothing is found, or the Identity object	Get the identity by ID and loads his profile.
getIdentity (String id, boolean loadProfile)	id can be a social GlobalId or a raw identity such as in Identity.getId(), loadProfile - the value is true if the profile is loaded and false if not loaded	null if nothing is found, or the Identity object	Get the identity by loading id of the profile optionally.
deleteIdentity (Identity identity)	identity	void	Delete an identity.
addIdentityProvider (IdentityProvider idProvider)	IdentityProvider - the id of provider	void	Add the identity provider.
getOrCreateIdentity (String providerId, String remoteId)	providerId - the id of provider, remoteId - the remote id	Identity	Get the identity by remoteId. If the provider can not find any identity by remoteId, the return value is null. If no identity found by identity provider and that identity is still stored on JCR, the stored identity will be deleted and the return value is null.
getOrCreateIdentity (String providerId, String remoteId, boolean loadProfile)	providerId - referring to the name of the Identity provider, remoteId - the identifier that identify the identity in the specific identity provider, loadProfile - true when the profile is loaded	null if nothing is found, or the Identity object improves the performance by specifying what needs to be loaded	This function returns an Identity object that is specific to a special type. For example, if the type is Linked'In, the identifier will be the URL of profile or if it is a CS contact manager, it will be the UID of the contact. A new identity is created if it does not exist.
getIdentitiesByProfileFilter (String providerId, ProfileFilter profileFilter) throws	providerId - the id of provider, profileFilter - the filter of provider	Identity	Get the identities by a profile filter.

Method	Param	Return	Description
Exception			
getIdentitiesByProfileFilter (String providerId, ProfileFilter profileFilter, long offset, long limit) throws Exception	providerId, profileFilter, offset, limit	List<Identity>	Get the identities by a profile filter.
getIdentitiesByProfileFilter (ProfileFilter profileFilter) throws Exception	ProfileFilter - the profile filter	List<Identity>	Get the identities by a profile filter.
getIdentitiesByProfileFilter (ProfileFilter profileFilter, long offset, long limit) throws Exception	ProfileFilter, profileFilter, offset, limit	List<Identity>	Get the identities by a profile filter.
getIdentitiesFilterByAlphabet (String providerId, ProfileFilter profileFilter) throws Exception	String - the id of provider, profileFilter - the profile filter	List<Identity>	Get the identities filter by alphabet.
getIdentitiesFilterByAlphabet (String providerId, ProfileFilter profileFilter, long offset, long limit) throws Exception	String, profileFilter, offset, limit	List<Identity>	Get the identities filter by alphabet with offset and limit.
getIdentitiesFilterByAlphabet (ProfileFilter profileFilter) throws Exception	ProfileFilter - the profile filter	List<Identity>	Get the identities filter by alphabet.
getIdentity (String providerId, String remoteId, boolean loadProfile)	providerId, remoteId, loadProfile	Identity	Get the identity.
getIdentitiesCount (String providerId)	providerId	long	Get the number of identities.
identityExisted (String providerId, String remoteId)	providerId, remoteId	boolean	Check if the identity is already existed or not.
saveIdentity (Identity identity)	identity - the identity	void	Save the identity.
saveProfile (Profile profile)	profile	void	Save a profile.
addOrModifyProfileProperties (Profile profile) throws Exception	Profile	void	Add or modify properties of profile. Profile parameter is a lightweight that contains only the property that you want to

Method	Param	Return	Description
			add or modify. NOTE: The method will not delete the properties of an old profile when the param profile does not have those keys.
updateAvatar (Profile p)	profile	void	Update the avatar.
updateBasicInfo (Profile p) throws Exception	profile	void	Update the basic information.
updateContactSection (Profile p) throws Exception	profile	void	Update the contact section of the profile.
updateExperienceSection (Profile p) throws Exception	profile	void	Update the experience section of the profile.
updateHeaderSection (Profile p) throws Exception	profile	void	Update the header section of the profile.
getIdentities (String providerId) throws Exception	providerId - the id of provider	List<Identity>	Get the identities by the provider id.
getIdentities (String providerId, boolean loadProfile)	providerId - the id of provider, loadProfile - the loaded profile.	List<Identity>	Get the identities by the provider id. If loadProvider is true, loading the profile will be performed.
getConnections (Identity ownerIdentity) throws Exception	ownerIdentity	List<Identity>	Get connections of an identity. (Since 1.1.1).
getIdentityStorage ()	N/A	IdentityStorage	Get the identity storage.
getStorage ()	N/A	IdentityStorage	Get the storage. Deprecated: should use method <code>getIdentityStorage()</code> .
registerProfileListener (ProfileListener listener)	ProfileListener	void	Register the profile listener.
unregisterProfileListener (ProfileListener listener)	ProfileListener	void	Unregister the profile listener.
addProfileListener (ProfileListener plugin)	ProfileListener plugin	void	Register a profile listener component plug-in.

3.3.3. RelationshipManager

Method	Param	Return	Description
getRelationshipById (String id) throws Exception		Relationship	Get the relationship by id. You should use get(String) instead. It will be removed at 1.2.x.
invite (Identity sender, Identity receiver) throws RelationshipStorageException	sender receiver	Relationship	Create a connection invitation between two identities.
saveRelationship (Relationship relationship) throws RelationshipStorageException	relationship - a relationship	void	Save a relationship.
confirm (Relationship relationship) throws RelationshipStorageException	relationship - a pending relationship	void	Mark a relationship as confirmed.
deny (Relationship relationship) throws RelationshipStorageException	relationship - a pending relationship	void	Deny a relationship.
remove (Relationship relationship) throws RelationshipStorageException	relationship - a pending relationship	void	Remove a relationship.
ignore (Relationship relationship) throws RelationshipStorageException	relationship - a pending relationship	void	Mark a relationship as ignored
getPendingRelationships (Identity sender) throws Exception	Identity - an identity	List<Relationship>	Get all the pending relationship of sender.
getPendingRelationships (Identity sender, List<Identity> identities) throws Exception	Identity - an identity, identities - a list of identity	List<Relationship>	Get pending relationships of sender that match with identities.
getRequireValidationRelationships (Identity receiver) throws Exception	Identity - an identity	List<Relationship>	Get list of required validation relationship of receiver.
getRequireValidationRelationships (Identity receiver, List<Identity> identities)	Identity - an identity, identities - a list of identity	List<Relationship>	Get list of required validation relationship of receiver that match with identities.
getConfirmedRelationships (Identity identity)	Identity - an identity	List<Relationship>	Get list of confirmed relationship of identity.
getConfirmedRelationships (Identity identity, List<Identity> identities)	Identity - an identity, identities - a list of identity	List<Relationship>	Get list of confirmed relationship of identity that match with identities.
getAllRelationships (Identity identity)	Identity - an identity	List<Relationship>	Return all the relationship of a given identity with other identity.

Method	Param	Return	Description
getAllRelationships (Identity identity, List<Identity> identities)	identity - an identity, identities - a list of identity	List<Relationship>	Return all the relationship of a given identity with other identity in identities.
getAllRelationships (Identity identity)	identity - an identity	List<Relationship>	Return all the relationship of a given identity with other identity.
getAllRelationships (Identity identity, Relationship.Type type, List<Identity> identities)	identity - an identity, type - a Relationship.Type, identities - a list of identity <Relationship>		Return all the relationship of a given identity with other identity in identities in type.
getRelationship (Identity identity1, Identity identity2)	identity1 and identity2 - identities	Relationship	Get the relationship of two identities.

3.3.4. SpaceService

Method	Param	Return	Description
getSpaceByDisplayName (String spaceDisplayName)	spaceDisplayName	Space	Get a space whose display name matches the string input. (Since 1.2.0-GA).
getSpaceByPrettyName (String spaceName)	spaceName	Space	Get a space whose pretty name matches the string input. (Since 1.2.0-GA).
getSpaceByGroupId (String groupId)	groupId	Space	Get a space that has group Id matching the string input.
getSpaceById (String spaceId)	spaceId	Space	Get a space by its Id.
getSpaceByUrl (String spaceUrl)	spaceUrl	Space	Get a space whose URL matches the string input.
getAllSpaces ()	N/A	ListAccess<Space>	Get a list of spaces with the type of a space list access. (Since 1.3.0-GA).
getAllSpacesWithListAccess ()		ListAccess<Space>	Get a list of spaces with the type of a space list access. (Since 1.2.0-GA).
getAllSpacesByFilter (SpaceFilter spaceFilter)	spaceFilter	ListAccess<Space>	Get a list of spaces with the type of a space list access. These spaces

Method	Param	Return	Description
			matches the space filter. (Since 1.2.0-GA).
getMemberSpaces (String userId)	userId	ListAccess<Space>	Get a list of spaces with the type of list access that contains all the spaces in which a user has the "member" role. (Since 1.2.0-GA).
getMemberSpacesByFilter (String userId, SpaceFilter spaceFilter)	(String, spaceFilter)	ListAccess<Space>	Get a list of spaces with the type of a space list access. The list contains the spaces which a user has the "member" role and match the provided space filter. (Since 1.2.0-GA).
getAccessibleSpaces (String userId)	userId	ListAccess<Space>	Get a list of spaces with the type of a space list access. The list contains all the spaces that a user has the access permission. (Since 1.3.0-GA).
getAccessibleSpacesWithListAccess (String userId)		ListAccess<Space>	Get a list of spaces with a space list access. The list contains all the spaces that a user has the access permission. (Since 1.2.0-GA).
getAccessibleSpacesByFilter (String userId, SpaceFilter spaceFilter)	(String, spaceFilter)	ListAccess<Space>	Get a list of spaces with the type of a space list access. The list contains all the spaces that a user has the access permission and match the provided space filter. (Since 1.2.0-GA).
getSettingableSpaces (String userId)	userId	ListAccess<Space>	Get a list of spaces with the type of a space list access. The list contains all the spaces that a user has the setting permission. (Since 1.2.0-GA).
getSettingabledSpacesByFilter (String userId, SpaceFilter spaceFilter)	(String, spaceFilter)	ListAccess<Space>	Get a list of spaces with the type of a space list

Method	Param	Return	Description
spaceFilter)			access. The list contains all the spaces that a user has the setting permission and match the provided space filter. (Since 1.2.0-GA).
getInvitedSpaces (String userId)	userId	ListAccess<Space>	Get a list of spaces with the type of a space list access. The list contains all the spaces that a user is invited to join. (Since 1.3.0-GA).
getInvitedSpacesWithListAccess (String userId)		ListAccess<Space>	Get a list of spaces with the type of a space list access. The list contains all the spaces that a user is invited to join. (Since 1.2.0-GA).
getInvitedSpacesByFilter (String userId, SpaceFilter spaceFilter)	String, spaceFilter	ListAccess<Space>	Get a list of spaces with the type of a space list access. The list contains all the spaces that a user is invited to join and match the provided space filter. (Since 1.2.0-GA).
getPublicSpaces (String userId)	userId	ListAccess<Space>	Get a list of spaces with the type of a space list access. The list contains all the spaces that a user can request to join. (Since 1.3.0-GA).
getPublicSpacesWithListAccess (String userId)		ListAccess<Space>	Get a list of spaces with the type of a space list access. The list contains all the spaces that a user can request to join. (Since 1.2.0-GA).
getPublicSpacesByFilter (String userId, SpaceFilter spaceFilter)	String, spaceFilter	ListAccess<Space>	Get a list of spaces with the type of a space list access. The list contains all the spaces that a user can request to join and match the provided space filter. (Since 1.2.0-GA).
getPendingSpaces (String userId)	userId	ListAccess<Space>	Get a list of spaces with the type of a space list

Method	Param	Return	Description
			access. The list contains all the spaces that a user sent join-request to a space. (Since 1.3.0-GA).
getPendingSpacesWithListAccess (String userId)	String	ListAccess<Space>	Get a list of spaces with the type of a space list access. The list contains all the spaces that a user sent a request for joining a space. (Since 1.2.0-GA).
getPendingSpacesByFilter (String userId, SpaceFilter spaceFilter)	String, spaceFilter	ListAccess<Space>	Get a list of spaces with the type of a space list access. The list contains all the spaces that a user sent join-request to a space and match the provided space filter. (Since 1.2.0-GA).
createSpace (Space space, String creatorUserId)	space, creatorUserId	Space	Create a new space: create a group, its group navigation with pages for installing space applications.
updateSpace (Space existingSpace)	existingSpace	Space	Update information of a space. (Since 1.2.0-GA).
deleteSpace (Space space)	space	void	Delete a space. When deleting a space, all of its page navigation bars and its group will be deleted.
addPendingUser (Space space, String userId)	space, userId	void	Add a user to the pending list to request to join a space. (Since 1.2.0-GA).
removePendingUser (Space space, String userId)	space, userId	void	Remove a user from the pending list to request to join a space. (Since 1.2.0-GA).
isPendingUser (Space space, String userId)	space, userId	void	Check if a user is in the pending list to request to join a space or not. (Since 1.2.0-GA).
addInvitedUser (Space space, String userId)	space, userId	void	Add a user, who is invited to a space, to the invited list. (Since 1.2.0-GA).

Method	Param	Return	Description
removeInvitedUser (Space space, String userId)	space, userId	void	Remove a user, who is invited to a space, from the invited list. (Since 1.2.0-GA).
isInvitedUser (Space space, String userId)	space, userId	void	Check if a user invited to join a space is in the invited list or not. (Since 1.2.0-GA).
addMember (Space space, String userId)	space, userId	void	Add a user to a space. The user will get the "member" role in that space.
removeMember (Space space, String userId)	space, userId	void	Remove a member from a space.
isMember (Space space, String userId)	space, userId	boolean	Check whether a user is a space's member or not.
setManager (Space space, String userId, boolean isManager)	space, userId, isManager	void	Add a user to have the "manager" role in a space. If <i>isManager</i> is set to "true", a user will get the "manager" role. If false, that user will get the "member" role. (Since 1.2.0-GA).
isManager (Space space, String userId)	space, userId	void	Check if a user has the "manager" role in a space or not. (Since 1.2.0-GA).
isOnlyManager (Space space, String userId)	space, userId	boolean	Check if a user is the only one who has the "manager" role in a space. True if the user Id is the only one who has "manager" role in a space. Otherwise, return false. (Since 1.2.0-GA).
hasAccessPermission (Space space, String userId)	space, userId	boolean	Check if a user can access a space or not. If the user is root or the space's member, return true.
hasSettingPermission (Space space, String userId)	space, userId	boolean	Check if a user can have the setting permission to a space or not. If the user is root or the space's member, return true. (Since 1.2.0-GA).

Method	Param	Return	Description
registerSpaceListenerPlugin (SpaceListenerPlugin spaceListenerPlugin)	SpaceListenerPlugin	void	Register a space listener plugin to listen to space lifecycle events: creating, removing space, activating, deactivating, adding, removing application, promoting, joining, leaving, and revoking. (Since 1.2.0-GA).
unregisterSpaceListenerPlugin (SpaceListenerPlugin spaceListenerPlugin)	SpaceListenerPlugin	void	Unregister an existing space listener plugin. (Since 1.2.0-GA).
setSpaceApplicationConfigPlugin (SpaceApplicationConfigPlugin spaceApplicationConfigPlugin)	SpaceApplicationConfigPlugin		Set a space application configuration plugin to configure the home and space applications. By configuring this, the space service will know how to create a new page node with title, URL, and portlet to use. (Since 1.2.0-GA).
getSpaceApplicationConfigPlugin ()		SpaceApplicationConfigPlugin	Get the configuration of applications to be initialized when creating a new space. (Since 1.2.0-GA).
getAllSpaces() throws SpaceException	N/A	List<Space>	Get all spaces in eXo Social. You should use method <code>getAllSpaceWithListAccess</code> instead of <code>getAllSpaces</code> . It will be removed in eXo Social 1.3.x.
getSpaceByName (String spaceName) throws SpaceException	spaceName	Space	Get a space by its name. You should use <code>SpaceService#getSpaceByPrettyName</code> instead. It will be removed version 1.3.x.
getSpacesByFirstCharacterOfName (String firstCharacterOfName) throws SpaceException	String	List<Space>	Get all spaces whose name starting with the input character.
getSpacesBySearchCondition (String condition) throws Exception	String	List<Space>	Get all spaces which has the name or the description that matches

Method	Param	Return	Description
			the input condition.
getSpaces (String userId) throws SpaceException	userId	List<Space>	Get spaces of a user in which that user is a member. You should use <code>getMemberSpaces(String)</code> instead. It will be removed in eXo Social 1.3.x
getAccessibleSpaces (String userId) throws SpaceException	userId	List<Space>	Get spaces of a user which that user has the access permission. You should use <code>getAccessibleSpacesWithListAccess(String)</code> instead. It will be removed in eXo Social 1.3.x.
getEditableSpaces (String userId) throws SpaceException	userId	List<Space>	Get spaces of a user which that user has the edit permission. You should use <code>getSettingableSpaces(String)</code> instead. It will be removed in eXo Social 1.3.x.
getInvitedSpaces (String userId) throws SpaceException	userId	List<Space>	Get invited spaces of a user and that user can accept or deny the request. You should use <code>getInvitedSpacesWithListAccess(String)</code> instead. It will be removed in eXo Social 1.3.x.
getPublicSpaces (String userId) throws SpaceException	userId - Id of user	List<Space>	Get invited spaces of a user that can be accepted or denied by the user. You should use <code>getPublicSpacesWithListAccess(String)</code> instead. It will be removed in eXo Social 1.3.x.
getPendingSpaces (String userId) throws SpaceException	userId	List<Space>	Get pending spaces of a user and spaces which the user can revoke that request. You should use <code>getPendingSpacesWithListAccess(String)</code> instead. It will be removed in eXo Social

Method	Param	Return	Description
			1.3.x.
createSpace (Space space, String creator, String invitedGroupId) throws SpaceException	space, creator, invitedGroupId	Space	Create a new space and invite all users from invitedGroupId to join this newly created space.
saveSpace (Space space, boolean isNew) throws SpaceException	space, isNew	void	Save a new space or update a space. You should use <code>updateSpace(org.exoplatform.social.core.space.Space)</code> instead. It will be removed in eXo Social 1.3.x.
deleteSpace (String spaceId) throws SpaceException	spaceId	void	Delete a space by its id. You should use <code>deleteSpace(org.exoplatform.social.core.space.Space)</code> instead. It will be removed in eXo Social 1.3.x.
initApp (Space space) throws SpaceException	space	void	It is just for compatibility. Deprecated: it will be removed in eXo Social 1.3.x.
initApps (Space space) throws SpaceException	space	void	It is just for compatibility. Deprecated: it will be removed in eXo Social 1.3.x.
deInitApps (Space space) throws SpaceException	space	void	It is just for compatibility. Deprecated: it will be removed in eXo Social 1.3.x.
addMember (String spaceId, String userId) throws SpaceException	spaceId, userId	void	Add a user to a space, the user will get the "member" role in a space. You should use <code>addMember(org.exoplatform.social.core.space.Space, String)</code> instead. It will be removed in eXo Social 1.3.x.
removeMember (String spaceId, String userId) throws SpaceException	spaceId, userId	void	Remove a member from a space. You should use <code>removeMember(org.exoplatform.social.core.space.Space, String)</code> instead. It will be removed in eXo Social 1.3.x.
getMembers (Space space)	space	List<String>	Get a list of the space

Method	Param	Return	Description
space) throws SpaceException			members from a space. You should use Space#getMembers() instead. It will be removed in eXo Social 1.3.x.
getMembers (String spaceId) throws SpaceException	spaceId	List<String>	Get a list of the space members from a space. You should use Space#getMembers() instead. It will be removed in eXo Social 1.3.x.
setLeader (Space space, String userId, boolean isLeader) throws SpaceException	space, userId, isLeader	void	Set a member of a space as a manager. You should use setManager(org.exoplatform.social.core.SpaceImpl, String, boolean) instead. It will be removed in eXo Social 1.3.x.
setLeader (String spaceId, String userId, boolean isLeader) throws SpaceException	spaceId, userId, isLeader	void	Set a member of a space as a manager. You should use setManager(org.exoplatform.social.core.SpaceImpl, String, boolean) instead. It will be removed in eXo Social 1.3.x.
isLeader (Space space, String userId) throws SpaceException	space, userId	boolean	Check whether a user is a space's leader or not. You should use isManager(org.exoplatform.social.core.SpaceImpl, String) instead. It will be removed in eXo Social 1.3.x.
isLeader (String spaceId, String userId) throws SpaceException	spaceId, userId	boolean	Check whether a user is a space's leader or not. You should use isManager(org.exoplatform.social.core.SpaceImpl, String) instead. It will be removed in eXo Social 1.3.x.
isOnlyLeader (Space space, String userId) throws SpaceException	space, userId	boolean	Check whether a user is the only leader of a space or not. You should use isOnlyManager(org.exoplatform.social.core.SpaceImpl, String) instead. It will be removed in eXo Social 1.3.x.

Method	Param	Return	Description
			1.3.x.
isOnlyLeader (String spaceId, String userId) throws SpaceException	spaceId, userId	boolean	Check whether a user is the only leader of a space or not. You should use <code>isOnlyManager(org.exoplatform.social.core.SpaceImpl space, String userId)</code> instead. It will be removed in eXo Social 1.3.x.
isMember (String spaceId, String userId) throws SpaceException	spaceId, userId,	boolean	Check whether a user is a space's member or not. You should use <code>isMember(org.exoplatform.social.core.SpaceImpl space, String userId)</code> instead. It will be removed in eXo Social 1.3.x.
hasAccessPermission (String spaceId, String userId) throws SpaceException	spaceId, userId	boolean	Check if a user can access a space or not. You should use <code>hasAccessPermission(org.exoplatform.social.core.SpaceImpl space, String userId)</code> instead. It will be removed in eXo Social 1.3.x.
hasEditPermission (Space space, String userId) throws SpaceException	space, userId	Boolean	Check if a user can have the edit permission of a space or not. You should use <code>hasSettingPermission(org.exoplatform.social.core.SpaceImpl space, String userId)</code> instead. It will be removed in eXo Social 1.3.x.
hasEditPermission (String spaceId, String userId) throws SpaceException	spaceId, userId	Boolean	Check if a user can have the edit permission of a space or not. You should use <code>hasSettingPermission(org.exoplatform.social.core.SpaceImpl space, String userId)</code> instead. It will be removed in eXo Social 1.3.x.
isInvited (Space space, String userId) throws SpaceException	space, userId	Boolean	Check if a user is in the invited list of a space or not. You should use <code>isInvitedUser(org.exoplatform.social.core.SpaceImpl space, String userId)</code> instead. It will be removed in eXo Social 1.3.x.
isInvited (String spaceId, String userId)	spaceId, userId	Boolean	Check if a user is in the

Method	Param	Return	Description
String getUserId) throws SpaceException			invited list of a space or not. You should use isInvitedUser(org.exoplatform.social.c String) instead. It will be removed in eXo Social 1.3.x.
isPending (Space space, String userId) throws SpaceException	space, userId	Boolean	Check if a user is in the pending list of a space or not. You should use isPendingUser(org.exoplatform.social. String) instead. It will be removed in eXo Social 1.3.x.
isPending (String spaceId, String userId) throws SpaceException	spaceId, userId	Boolean	Check if a user is in the pending list of a space or not. You should use isPendingUser(org.exoplatform.social. String) instead. It will be removed in eXo Social 1.3.x.
installApplication (String spaceId, String appId) throws SpaceException	spaceId, appId	void	Install an application to a space.
installApplication (Space space, String appId) throws SpaceException	space, appId	void	Install an application to a space.
activateApplication (Space space, String appId) throws SpaceException	space, appId	void	Activate an installed application in a space.
activateApplication (String spaceId, String appId) throws SpaceException	spaceId, appId	void	Activate an installed application in a space.
deactivateApplication (Space space, String appId) throws SpaceException	space, appId	void	Deactivate an installed application in a space.
deactivateApplication (String spaceId, String appId) throws SpaceException	spaceId, appId	void	Deactivate an installed application in a space.
removeApplication (Space space, String appId, String appName) throws SpaceException	space, appId, appName	void	Remove an installed application from a space.
removeApplication (String spaceId, String appId,	space, appId, appName	void	Remove an installed application from a space.

Method	Param	Return	Description
String appName) throws SpaceException			
requestJoin (Space space, String userId) throws SpaceException	space, userId	void	Request users to join a space. The invited users are then added to the pending list of the space. You should use <code>addPendingUser(org.exoplatform.social.core.space.SpaceImpl(String))</code> instead. It will be removed in eXo Social 1.3.x.
requestJoin (String spaceId, String userId) throws SpaceException	spaceId, userId	void	Request users to join a space. The invited users are then added to the pending list of the space.. You should use <code>addPendingUser(org.exoplatform.social.core.space.SpaceImpl(String))</code> instead. It will be removed in eXo Social 1.3.x.
revokeRequestJoin (Space space, String userId) throws SpaceException	space, userId	void	Revoke a join request after users request to join a group and is in the pending status. You should use <code>removePendingUser(org.exoplatform.social.core.space.SpaceImpl(String))</code> instead. It will be removed in eXo Social 1.3.x.
revokeRequestJoin (String spaceId, String userId) throws SpaceException	spaceId, userId	void	Revoke a request to join a space. You should use <code>removePendingUser(org.exoplatform.social.core.space.SpaceImpl(String))</code> instead. It will be removed in eXo Social 1.3.x.
inviteMember (Space space, String userId) throws SpaceException	space, userId	void	Invite a userId to become a member of a space. You should use <code>addInvitedUser(org.exoplatform.social.core.space.SpaceImpl(String))</code> instead. It will be removed in eXo Social 1.3.x.
inviteMember (String spaceId, String userId) throws SpaceException	spaceId, userId	void	Invite a userId to a be member of a space. You should use <code>addInvitedUser(org.exoplatform.social.core.space.SpaceImpl(String))</code> instead. It will be

Method	Param	Return	Description
			removed in eXo Social 1.3.x.
revokeInvitation (Space space, String userId) throws SpaceException	space, userId	void	Revoke an invitation. Remove a user from the invited member list of the space. You should use <code>removeInvitedUser(org.exoplatform.social.core.space.SpaceImpl.removeInvitedUser(String))</code> instead. It will be removed in eXo Social 1.3.x.
revokeInvitation (String spaceId, String userId) throws SpaceException	spaceId, userId	void	Revoke an invitation. Remove a user from the invited member list of the space. You should use <code>removeInvitedUser(org.exoplatform.social.core.space.SpaceImpl.removeInvitedUser(String))</code> instead. It will be removed in eXo Social 1.3.x.
acceptInvitation (Space space, String userId) throws SpaceException	space, userId	void	Accept an invitation and move a user from the invited list to the member list. You should use <code>addMember(org.exoplatform.social.core.space.SpaceImpl.addMember(String))</code> instead. It will be removed in eXo Social 1.3.x.
acceptInvitation (String spaceId, String userId) throws SpaceException	spaceId, userId	void	Accept an invitation and move a user from the invited list to the member list. You should use <code>addMember(org.exoplatform.social.core.space.SpaceImpl.addMember(String))</code> instead. It will be removed in eXo Social 1.3.x.
denyInvitation (Space space, String userId) throws SpaceException	space, userId	void	Deny an invitation and remove a user from the invited list. You should use <code>removeInvitedUser(org.exoplatform.social.core.space.SpaceImpl.removeInvitedUser(String))</code> instead. It will be removed in eXo Social 1.3.x.
denyInvitation (String spaceId, String userId) throws SpaceException	spaceId, userId	void	Deny an invitation and remove a user from the invited list. You should use <code>removeInvitedUser(org.exoplatform.social.core.space.SpaceImpl.removeInvitedUser(String))</code> instead. It will be removed in eXo Social 1.3.x.

Method	Param	Return	Description
			String) instead. It will be removed in eXo Social 1.3.x.
validateRequest (Space space, String userId) throws SpaceException	space, userId	void	Validate a request and move a user from the pending list to the member list. You should use <code>addMember(org.exoplatform.social.core.space.Space, String)</code> instead. It will be removed in eXo Social 1.3.x.
validateRequest (String spaceId, String userId) throws SpaceException	spaceId, userId	void	Validate a request and move a user from the pending list to the member list. You should use <code>addMember(org.exoplatform.social.core.space.Space, String)</code> instead. It will be removed in eXo Social 1.3.x.
declineRequest (Space space, String userId) throws SpaceException	space, userId	void	Decline a request and remove a user from the pending list. You should use <code>removePendingUser(org.exoplatform.social.core.space.Space, String)</code> instead. It will be removed in eXo Social 1.3.x.
declineRequest (String spaceId, String userId) throws SpaceException	spaceId, userId	void	Decline a request and remove a user from the pending list. You should use <code>removePendingUser(org.exoplatform.social.core.space.Space, String)</code> instead. It will be removed in eXo Social 1.3.x.
registerSpaceLifecycleListener (SpaceLifecycleListener listener)	listener	void	Register a space lifecycle listener. Deprecated: it will be removed in eXo Social 1.3.x.
unregisterSpaceLifecycleListener (SpaceLifecycleListener listener)	listener	void	Unregister a space lifecycle listener. Deprecated: it will be removed in eXo Social 1.3.x.

Method	Param	Return	Description
setPortletsPrefsRequired (PortletPreferencesRequiredPlugin portletPrefsRequiredPlugin)			Set the portlet preferences got from the plug-in configuration. You should use <code>SpaceApplicationConfigPlugin(org.exo.social.core.identity.model.IdentityManager)</code> instead. It will be removed in eXo Social 1.3.x.
getPortletsPrefsRequired ()	N/A	String	Get the portlet preferences required to use in creating the portlet application. Deprecated: it will be removed in eXo Social 1.3.x.

3.4. Java APIs sample code/ tutorial

3.4.1. Activity Stream

eXo Social provides users with a way to share their activity information (also known as Activity Stream) and collaborate in spaces (also known as group work). With the API, you can customize the way to display activities or publish new ones. To manipulate activities, you need to use the *ActivityManager* service.

3.4.1.1. Publish an activity

There are two types of activities: activities for a user and activities for a space. The following examples will show you how to publish an activity for each type.

3.4.1.1.1. Publish an activity for a user

```
package org.exoplatform.publish.user;

import org.exoplatform.services.log.ExoLogger;
import org.exoplatform.services.log.Log;

import org.exoplatform.container.PortalContainer;
import org.exoplatform.social.core.manager.ActivityManager;
import org.exoplatform.social.core.manager.IdentityManager;
import org.exoplatform.social.core.identity.model.Identity;
import org.exoplatform.social.core.activity.model.ExoSocialActivity;
import org.exoplatform.social.core.activity.model.ExoSocialActivityImpl;
import org.exoplatform.social.core.identity.provider.OrganizationIdentityProvider;

public class PublishActivityForUser {
    // Exo Log.
    private final Log LOG = ExoLogger.getLogger(PublishActivityForUser.class);

    // Portal container.
    private PortalContainer container;

    // identityManager manages identities.
    private IdentityManager identityManager;

    // activityManager manages activities.
    private ActivityManager activityManager;
```

```

private final static String DEFAULT_USER_NAME = "zun";
private final static String DEFAULT_ACTIVITY_TITLE = "Hello World!";

/**
 * Constructor.
 */
public PublishActivityForUser() {
    // Gets the current container.
    container = PortalContainer.getInstance();

    // Gets identityManager to handle an identity operation.
    identityManager = (IdentityManager) container.getComponentInstance(IdentityManager.class);

    // Gets activityManager to handle an activity operation.
    activityManager = (ActivityManager) container.getComponentInstanceOfType(ActivityManager.class);
}

public void createActivityForUser() {
    try {
        // Gets an existing identity or creates a new one.
        Identity userIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME, DEFAULT_USER_NAME);

        // Creates a new activity for this user.
        ExoSocialActivity activity = new ExoSocialActivityImpl();
        activity.setUserId(userIdentity.getId());
        activity.setTitle(DEFAULT_ACTIVITY_TITLE);
        // Saves an activity into JCR by using ActivityManager.
        activityManager.saveActivity(activity);
    } catch (Exception e) {
        LOG.error("can not save activity.", e);
    }
}
}

```

3.4.1.2. Publish an activity for a space

```

package org.exoplatform.publish.space;

import org.exoplatform.services.log.ExoLogger;
import org.exoplatform.services.log.Log;

import org.exoplatform.container.PortalContainer;
import org.exoplatform.social.core.manager.ActivityManager;
import org.exoplatform.social.core.manager.IdentityManager;
import org.exoplatform.social.core.identity.provider.OrganizationIdentityProvider;
import org.exoplatform.social.core.identity.model.Identity;
import org.exoplatform.social.core.activity.model.ExoSocialActivity;
import org.exoplatform.social.core.activity.model.ExoSocialActivityImpl;

import org.exoplatform.social.core.space.model.Space;
import org.exoplatform.social.core.space.SpaceException;
import org.exoplatform.social.core.space.spi.SpaceService;
import org.exoplatform.social.core.identity.provider.SpaceIdentityProvider;

public class PublishActivityForSpace {
    // Exo Log.
    private final Log LOG = ExoLogger.getLogger(PublishActivityForSpace.class);

    // Portal container.
    private PortalContainer container;

    // identityManager manages identities.
    private IdentityManager identityManager;

    // activityManager manages activities.
    private ActivityManager activityManager;

    // spaceService manages spaces.
    private SpaceService spaceService;

    private final static String DEFAULT_NAME_SPACE = "mySpace";
    private final static String DEFAULT_USER_NAME = "zun";
    private final static String DEFAULT_ACTIVITY_TITLE = "An activity for space";
}

```

```

/**
 * Constructor method.
 */
public PublishActivityForSpace() {
    // Gets the current container.
    container = PortalContainer.getInstance();

    // Gets identityManager to manage identities.
    identityManager = (IdentityManager) container.getComponentInstance(IdentityManager.class);

    // Gets activityManager to manage activities.
    activityManager = (ActivityManager) container.getComponentInstanceOfType(ActivityManager.class);

    // Gets spaceService to handle the operation of a space.
    spaceService = (SpaceService) container.getComponentInstanceOfType(SpaceService.class);
}

public void createActivityForSpace() {
    try {
        // make sure that a space with the name "mySpace" is created.
        Space space = spaceService.getSpaceByDisplayName(DEFAULT_NAME_SPACE);
        if (space != null) {
            // Gets spaceIdentity if it already exists. If not, a new one is created.
            Identity spaceIdentity = identityManager.getOrCreateIdentity(SpaceIdentityProvider.NAME, DEFAULT_NAME_SPACE);
            // Gets an identity if it already exists. If not, a new one is created.
            Identity userIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME, DEFAULT_NAME_SPACE);
            // Creates a new activity for this space.
            ExoSocialActivity activity = new ExoSocialActivityImpl();
            activity.setUserId(userIdentity.getId());
            activity.setTitle(DEFAULT_ACTIVITY_TITLE);
            activityManager.saveActivity(spaceIdentity, activity);
        }
    } catch (SpaceException e) {
        LOG.error("Can not save activity.", e);
    } catch (Exception e) {
        LOG.error("Can not save activity.", e);
    }
}
}

```

3.4.1.2.1. Configure an activity processor

An activity processor is used to modify the content of activities before they are returned from an activity manager. For example, to create an activity processor to replace all the texts representing the smile face ":-)" in the activity title by the smiley icons, do as follows:

Firstly, create the *SmileyProcessor* class by extending the *BaseActivityProcessorPlugin*.

```

package org.exoplatform.social.core.activitystream;

import org.exoplatform.social.core.BaseActivityProcessorPlugin;
import org.exoplatform.container.xml.InitParams;
import org.exoplatform.social.core.activity.model.ExoSocialActivity;

public class SmileyProcessor extends BaseActivityProcessorPlugin {
    private String smiley;

    public SmileyProcessor(InitParams params) {
        super(params);
        this.smiley = "<img src={{{{}}}http://www.tombraider4u.com/pictures/smiley.gif{{{{}}}}/>";
    }

    @Override
    public void processActivity(ExoSocialActivity activity) {
        String title = activity.getTitle();
        activity.setTitle(title.replaceAll(":-)", this.smiley));
    }
}

```

Then, register this processor by editing the *configuration.xml* file:

```
<external-component-plugins>
  <target-component>org.exoplatform.social.core.manager.ActivityManager</target-component>
  <component-plugin>
    <name>SmileyProcessor</name>
    <set-method>addProcessorPlugin</set-method>
    <type>org.exoplatform.social.core.activitystream.SmileyProcessor</type>
    <description/>
    <init-params>
      <values-param>
        <name>priority</name>
        <value>1</value>
      </values-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

The "init-params" contains all the key-value data which a processor will use to initialize. In the above configuration, the **priority** value indicates the order in which this processor is executed. If the value is 1, this processor will be used before all the remaining processors with the lower priority.

3.4.1.2.2. Publish an RSS feed with feedmash

It is really easy to publish an RSS feed to a space's activity stream. eXo Social provides *FeedmashJobPlugin* to publish the RSS feeds. As you can see in the project "exo.social.extras.feedmash", there are the *JiraFeedConsumer* and *HudsonFeedConsumer* samples to post eXo Social project's feeds (jira and hudson) to a pre-defined space named *exosocial* in a specific portal container named *socialdemo* as in the configuration file:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.scheduler.JobSchedulerService</target-component>
  <component-plugin>
    <name>RepubSocialJiraActivityJob</name>
    <set-method>addPeriodJob</set-method>
    <type>org.exoplatform.social.extras.feedmash.FeedmashJobPlugin</type>
    <description/>
    <init-params>
      <properties-param>
        <name>mash.info</name>
        <property name="feedURL" value="http://jira.exoplatform.org/plugins/servlet/streams?key=SOC"/>
        <property name="categoryMatch" value="resolved|created"/>
        <property name="targetActivityStream" value="space:exosocial"/>
        <property name="portalContainer" value="socialdemo"/>
      </properties-param>
      <properties-param>
        <name>job.info</name>
        <description>save the monitor data periodically</description>
        <property name="jobName" value="JIRAFeedConsumer"/>
        <property name="groupName" value="Feedmash"/>
        <property name="job" value="org.exoplatform.social.feedmash.JiraFeedConsumer"/>
        <property name="repeatCount" value="0"/>
        <property name="period" value="60000"/>
        <property name="startTime" value="+45"/>
        <property name="endTime" value=""/>
      </properties-param>
    </init-params>
  </component-plugin>
  <component-plugin>
    <name>WatchSocialBuildStatus</name>
    <set-method>addPeriodJob</set-method>
    <type>org.exoplatform.social.extras.feedmash.FeedmashJobPlugin</type>
    <description/>
    <init-params>
      <properties-param>
        <name>mash.info</name>
        <property name="feedURL" value="http://builder.exoplatform.org/hudson/view/social/job/social-trunk-ci/rs">
        <property name="targetActivityStream" value="space:exosocial"/>
        <property name="portalContainer" value="socialdemo"/>
      </properties-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

```

<properties-param>
  <name>job.info</name>
  <description>save the monitor data periodically</description>
  <property name="jobName" value="HudsonFeedConsumer"/>
  <property name="groupName" value="Feedmash"/>
  <property name="job" value="org.exoplatform.social.feedmash.HudsonFeedConsumer"/>
  <property name="repeatCount" value="0"/>
  <property name="period" value="60000"/>
  <property name="startTime" value="+100"/>
  <property name="endTime" value=""/>
</properties-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

Run eXo Social with the URL: <http://localhost:8080/socialdemo>, then log in and create a space named "exosocial". After creating the "exosocial" space, all the feeds of the eXo Social project on Jira and Hudson will be automatically published to the *exosocial* space.

3.4.1.3. Sample Code

See the following code snippet to know more details about how to publish an activity and add comments to an activity:

```

package org.exoplatform.social.introduction.activitystreamandexosocialactivity;

import org.exoplatform.services.log.ExoLogger;
import org.exoplatform.services.log.Log;

import org.exoplatform.container.PortalContainer;
import org.exoplatform.social.core.activity.model.ActivityStream;
import org.exoplatform.social.core.activity.model.ExoSocialActivity;
import org.exoplatform.social.core.activity.model.ExoSocialActivityImpl;
import org.exoplatform.social.core.manager.ActivityManager;
import org.exoplatform.social.core.manager.IdentityManager;
import org.exoplatform.social.core.identity.model.Identity;
import org.exoplatform.social.core.identity.provider.OrganizationIdentityProvider;

public class IntroduceActivityStreamAndExoSocialActivity {
  // Exo Log.
  private final Log LOG = ExoLogger.getLogger(IntroduceActivityStreamAndExoSocialActivity.class);

  // Demo identity.
  private Identity demoIdentity;

  // John identity.
  private Identity johnIdentity;

  // identityManager manages identities.
  private IdentityManager identityManager;

  // activityManager manages activities.
  private ActivityManager activityManager;

  // Portal container.
  private PortalContainer container;

  private final static String DEMO_NAME = "demo";
  private final static String JOHN_NAME = "john";
  private final static String DEFAULT_ACTIVITY_TITLE = "blabla";
  private final static String DEFAULT_COMMENT_TITLE = "comment blah blah";

  /**
   * Constructor.
   */
  public IntroduceActivityStreamAndExoSocialActivity() {
    // Gets the current container.
    container = PortalContainer.getInstance();

    // Gets IdentityManager to handle an identity operation.
    identityManager = (IdentityManager) container.getComponentInstanceOfType(IdentityManager.class);
  }

```

```

// Gets ActivityManager to handle activity operation.
ActivityManager activityManager = (ActivityManager) container.getComponentInstanceOfType(ActivityManager.class);

// Gets or create demo's identity
demoIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME, DEMO_NAME, false);

// Gets or creates the identity "john".
johnIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME, JOHN_NAME, false);
}

/**
 * Posts activity to activity stream
 */
public void introduceActivityStreamAndExoSocialActivity {

    // Sets a string that specifies the primary text of an activity. This field is REQUIRED by ActivityManager.
    activity.setTitle(DEFAULT_ACTIVITY_TITLE);

    // Sets this activity for demo.
    activity.setUserId(demoIdentity.getId());

    // Saves the activity.
    activityManager.saveActivity(johnIdentity, activity);

    // Gets activity stream.
    ActivityStream activityStream = activity.getActivityStream();

    // Type of the activity stream. It can be organization or space.
    LOG.info("activity stream type: " + activityStream.getType());

    LOG.info("activity stream id: " + activityStream.getId());
    LOG.info("activity stream pretty id: " + activityStream.getPrettyId());
    LOG.info("activity stream perma link: " + activityStream.getPermaLink());

    LOG.info("activity stream id: " + activity.getStreamId());
    LOG.info("activity stream owner: " + activity.getStreamOwner());

    // Comment in Social
    ExoSocialActivity demoActivity = new ExoSocialActivityImpl();
    activity.setTitle(DEFAULT_ACTIVITY_TITLE);
    activityManager.saveActivity(demoIdentity, demoActivity);

    ExoSocialActivity comment = new ExoSocialActivityImpl();
    comment.setTitle(DEFAULT_COMMENT_TITLE);

    //Sets comment of demo
    comment.setUserId(demoIdentity.getId());

    //Saves a comment.
    activityManager.saveComment(activity, comment);
}
}

```

3.4.2. OpenSocial

eXo Social supports the OpenSocial standard. So you can integrate OpenSocial gadgets in your dashboard and use the RPC or REST service to view or publish the social data. With the support for the OpenSocial standard, eXo Social provides a framework for developers to build gadgets that can display and mash up activity information for contacts, social networks, applications and services.

3.4.2.1. Gadget

Gadgets are web-based software components based on HTML, CSS, JavaScript; defined by using an XML declaration syntax. They allow developers to easily write social applications that work on the social networks supporting OpenSocial APIs without modification. See the following links for detailed information:

- [Gadgets Specification](#)

- [OpenSocial Core Gadget Specification v0.8](#)
- [Gadgets APIs](#)

To know how to create an OpenSocial gadget, see [here](#).



Note

Gadgets will work out of the box on Dashboard. In eXo, gadgets are wrapped by GadgetWrapperPortlet so they can work as any other portlet applications. At present, eXo Social supports [OpenSocial v0.8](#).

3.4.2.1.1. Supported APIs

eXo Social leverages [Apache Shindig](#) - an OpenSocial reference implementation to provide and extend OpenSocial APIs which is compatible with the common OpenSocial APIs which is supported by other big social networks like [Ning](#), [Hi5](#), [Orkut](#) and more.

To get more details about Supported APIs, refer to [Opensocial Specs](#).

3.4.2.1.1.1. REST/RPC API

Suppose that you are running the local host at port 8080 (<http://localhost:8080/>), the path of the API will be:

- REST API:

<http://localhost:8080/social/social/rest>

- RPC API:

<http://localhost:8080/social/social/rpc>

To learn what you can do with the APIs, have a look at the [specification](#). If you are developing in Java, you can use the [opensocial-java-client](#).

3.4.2.1.1.2. Configure the security

If you are using OpenSocial, you need to configure the OAuth authentication. With the case of eXo Platform, you need to edit the file: *gatein/conf/portal/portal/configuration.xml* and add the following configuration:

```
<component>
  <key>org.exoplatform.social.opensocial.oauth.ServiceProviderStore</key>
  <type>org.exoplatform.social.opensocial.oauth.ServiceProviderStore</type>
  <init-params>
    <properties-param>
      <name>grails-book-flow</name>
      <description>consumer key and secret for sample oauth provider. </description>
      <property name="consumerKey" value="YOUR_KEY_HERE"/>
      <property name="sharedSecret" value="YOUR_SECRET_KEY_HERE"/>
    </properties-param>
  </init-params>
</component>
```

consumerKey and *sharedSecret* are keys that need to be shared with the application which is doing the request.

3.4.2.1.1.3. Publish an activity into a space

This functionality is not available in the standard OpenSocial APIs.

Instead of publishing your activities to the group `@self` as usual, you will publish them to the group `"space:spaceId"` or `"space:spacePrettyName"`.

After using the OpenSocial Java library and Groovy, your code will look like this:

```
def client = getOpenSocialClient()

//create your new activity
Activity activity = new Activity()
    activity.body = "xx purchased the book xxx"
    activity.title = "BookFlow Purchased"

//prepare the request that will create the activity
Request request = ActivitiesService.createActivity(activity);
//specify that the creation of this new activity is for the space bookflow
request.groupId = "space:bookflow";

client.send(request);
```

In the example above, the `groupId` is set to `"space:bookflow"` and `bookflow` is the name of the space.

3.4.2.2. Tutorial

- See [Grails + eXo Social tutorial](#).

3.4.3. People

eXo Social provides a way to manage profile information, and connections between users. With the APIs provided, you can easily add, manage and customize information and relationships of users.

3.4.3.1. Identity

The identity allows identifying a unique social object. The eXo Social objects can be person, group, application, or whatever related to social interactions, such as connecting, publishing an activity stream or holding a user profile.

3.4.3.1.1. IdentityProvider

eXo Social provides `IdentityProvider` as an identity mechanism for the third parties to extend eXo Social's identity system without any limitation. Here is an example:

```
class SampleIdentityProvider extends OrganizationIdentityProvider{
    public SampleIdentityProvider(OrganizationService organizationService) {
        super(organizationService);
    }

    @Override
    public void populateProfile(Profile profile, User user) {
        profile.setProperty(Profile.FIRST_NAME, "this is first name");
        profile.setProperty(Profile.LAST_NAME, "this is last name");
        profile.setProperty(Profile.USERNAME, "this is user name");
        profile.setProperty(Profile.URL, "/path/to/profile/");
    }
}
```

In this example, the *SampleIdentityProvider* class extends *OrganizationIdentityProvider* which is the *IdentityProvider* used to connect to the portal user's base. In this class, the *populateProfile* method is overridden and some dummy data are added to the profile fields.

3.4.3.1.2. IdentityManager

IdentityManager is the service used to manipulate the identity operations, such as creating, getting, deleting or finding a profile. You can get the *IdentityManager* via the *ExoContainer*. The following code will show how to get an *IdentityManager* instance and create a basic identity instance:

```
import org.exoplatform.container.ExoContainer;
import org.exoplatform.container.ExoContainerContext;
import org.exoplatform.social.core.identity.IdentityManager;
import org.exoplatform.social.core.identity.impl.organization.OrganizationIdentityProvider;
import org.exoplatform.social.core.identity.model.Identity;

//.....

String containerName = "portal";
String username = "zun";

//get container to get other registered components
ExoContainer container = ExoContainerContext.getContainerByName(containerName);

//get IdentityManager to handle identity operation
IdentityManager identityManager = (IdentityManager) container.getComponentInstanceOfType(IdentityManager.class);

//get ActivityManager to handle activity operation
ActivityManager activityManager = (ActivityManager) container.getComponentInstanceOfType(ActivityManager.class);

//create new user with name Zun
Identity userIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME, username);
```

3.4.3.2. ProfileListener

APIs provide notification interfaces which you can implement to create your own handlers for notification when having the profile modifications by extending the *ProfileListenerPlugin* class, or relationship changes by extending [RelationshipListenerPlugin](#).

1. Create the *ProfileLoggerListener* class to log all profile modifications of the systems. The abstract class named *ProfileListenerPlugin* provides the interface to implement this method.

```
import org.exoplatform.services.log.ExoLogger;
import org.exoplatform.services.log.Log;
import org.exoplatform.social.core.identity.lifecycle.ProfileListenerPlugin;
import org.exoplatform.social.core.identity.spi.ProfileLifecycleEvent;

public class ProfileLoggerListener extends ProfileListenerPlugin{
    private static final Log logger = ExoLogger.getExoLogger(ProfileLoggerListener.class);
    @Override
    public void avatarUpdated(ProfileLifecycleEvent event) {
        logger.info("@ " + event.getUsername() + " profile has updated his basic profile info.");
    }

    @Override
    public void basicInfoUpdated(ProfileLifecycleEvent event) {
        logger.info("@ " + event.getUsername() + " profile has updated his basic profile info.");
    }

    @Override
    public void contactSectionUpdated(ProfileLifecycleEvent event) {
        logger.info("@ " + event.getUsername() + " profile has updated his contact info.");
    }

    @Override
```

```

public void experienceSectionUpdated(ProfileLifecycleEvent event) {
    logger.info("@ " + event.getUsername() + " profile has an updated experience section.");
}

@Override
public void headerSectionUpdated(ProfileLifecycleEvent event) {
    logger.info("@ " + event.getUsername() + " has updated his header info.");
}
}

```

2. Add some configurations to this class to the configuration.xml:

```

<external-component-plugins>
  <target-component>org.exoplatform.social.core.identity.IdentityManager</target-component>
  <component-plugin>
    <name>ProfileLoggerListener</name>
    <set-method>addProfileListener</set-method>
    <type>path.to.ProfileLoggerListener</type>
  </component-plugin>
</external-component-plugins>

```

3.4.3.3. Connections

Similarly, you can apply the above steps to implement *RelationshipListenerPlugin* for relationship notifications.

Relationship is the bridge between two identities in eXo Social. There are many types of relationships defined in the Relationship class. With these types, a user can invite another user, confirm invitations or remove relationship.

3.4.3.3.1. Users connection

The following code will show you how to invite a user to connect with another:

```

import org.exoplatform.container.ExoContainer;
import org.exoplatform.container.ExoContainerContext;
import org.exoplatform.social.core.identity.IdentityManager;
import org.exoplatform.social.core.identity.impl.organization.OrganizationIdentityProvider;
import org.exoplatform.social.core.identity.model.Identity;
import org.exoplatform.social.core.relationship.Relationship;
import org.exoplatform.social.core.relationship.RelationshipManager;

public void inviteUser() throws Exception {
    String containerName = "portal";
    ExoContainer container = ExoContainerContext.getContainerByName(containerName);

    IdentityManager identityManager = (IdentityManager) container.getComponentInstanceOfType(IdentityManager.class);

    Identity invitedIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME, "Hoat");
    String invitedUserId = invitedIdentity.getId();

    String currUserId = "Zun";
    Identity currentIdentity = identityManager.getOrCreateIdentity(OrganizationIdentityProvider.NAME, currUserId);

    RelationshipManager relationshipManager = (RelationshipManager) container.getComponentInstanceOfType(RelationshipManager.class);
    Relationship relationship = relationshipManager.invite(currentIdentity, invitedIdentity);
}

```

3.4.3.3.2. RelationshipListener

The following example will show you how to implement one of these interfaces and how to configure this plugin. The following step is similar to the Profile notifications implementation.

1. Create the *RelationshipLoggerListener* class:

```
import org.exoplatform.social.core.relationship.Relationship;
import org.exoplatform.social.core.relationship.lifecycle.RelationshipListenerPlugin;
import org.exoplatform.social.relationship.spi.RelationshipEvent;

public class RelationshipLoggerListener extends RelationshipListenerPlugin{
    private static final Log logger = ExoLogger.getExoLogger(RelationshipLoggerListener.class);

    @Override
    public void confirmed(RelationshipEvent event) {
        String[] names = getUserNamesFromEvent(event);
        logger.info(names[0] + " confirmed the invitation of " + names[1]);
    }

    @Override
    public void ignored(RelationshipEvent event) {
        String[] names = getUserNamesFromEvent(event);
        logger.info(names[0] + " ignored the invitation of " + names[1]);
    }

    @Override
    public void removed(RelationshipEvent event) {
        String[] names = getUserNamesFromEvent(event);
        logger.info(names[0] + " removed the relationship with " + names[1]);
    }

    private String[] getUserNamesFromEvent(RelationshipEvent event){
        Relationship relationship = event.getPayload();

        Identity id1 = relationship.getIdentity1();
        Identity id2 = relationship.getIdentity2();

        String user1 = "@" + id1.getRemoteId();
        String user2 = "@" + id2.getRemoteId();

        return new String[]{user1, user2 };
    }
}
```

2. Add some configurations for this class in the *configuration.xml* file:

```
<external-component-plugins>
  <target-component>org.exoplatform.social.core.relationship.RelationshipManager</target-component>
  <component-plugin>
    <name>RelationshipLoggerListener</name>
    <set-method>addListenerPlugin</set-method>
    <type>classpath.of.your.RelationshipLoggerListener</type>
  </component-plugin>
</external-component-plugins>
```

3.4.4. Spaces

eXo Social provides a way to create groups and to share data and applications by spaces. A space has its own activity stream in which applications or members can publish information. In each space, members use applications together with shared data.

To manipulate the spaces, you will use the *SpaceService*. To get an instance of this class, you need to get the current *PortalContainer* instance.

3.4.4.1. Spaces Management

The following example will show how to create a space:



```

package org.exoplatform.social.sample;

import org.exoplatform.container.PortalContainer;
import org.exoplatform.social.application.impl.DefaultSpaceApplicationHandler;
import org.exoplatform.social.space.Space;
import org.exoplatform.social.space.SpaceException;
import org.exoplatform.social.space.SpaceService;

public class SpaceCreationSample {

    public void createSpace() throws SpaceException {
        String spaceName = "mySpace";
        String creator = "jeremi";
        PortalContainer container = PortalContainer.getInstance();
        SpaceService spaceService = (SpaceService) container.getComponentInstanceOfType(SpaceService.class);
        // verify if there is no space already created
        Space space = spaceService.getSpaceByDisplayName(spaceName);
        if (space == null) {
            space = new Space();
            space.setDisplayName(spaceName);
            space.setRegistration(Space.OPEN);
            space.setDescription("space description");
            //DefaultSpaceApplicationHandler is the default implementation of SpaceApplicationHandler. You can create y
            space.setType(DefaultSpaceApplicationHandler.NAME);
            //create the space
            space = spaceService.createSpace(space, creator);
            //initialize the applications
            spaceService.initApps(space);
        }
    }
}

```

3.4.4.2. Space's applications management

3.4.4.2.1. Add an application to a space

You can add portlet or gadget applications to spaces. Once added, all members of the space can use that application. The following code shows you how to add an application to a space:

```

public void addApplicationToSpace() throws SpaceException {
    //Your portlet name
    String appId = "SocialBannerPortlet";
    String spaceId = "zunSpace";

    //get container to get other registered components
    PortalContainer container = PortalContainer.getInstance();

    //get space service for installing operations
    SpaceService spaceService = (SpaceService) container.getComponentInstanceOfType(SpaceService.class);

    //install application for the space
    spaceService.installApplication(spaceId, appId);

    //you must activate installed application to be able to use it
    spaceService.activateApplication(spaceId, appId);
}

```



Note

appId is the portlet or gadget name as defined in *portlet.xml* or *gadget.xml* in the web-app. You can find it in *social-portlet.war* and *social.war*.

3.4.4.2.2. Remove an application from a space

You can remove portlet or gadget applications from a space and the members of this space will not see them

anymore. The following code shows you how to remove an application from a space:

```
public void removeApplicationFromSpace() throws SpaceException {
    //Your portlet name
    String appId = "SocialBannerPortlet";
    String appName = "SocialBannerPortlet1"
    String spaceId = "zunSpace";

    //get container to get other registered components
    PortalContainer container = PortalContainer.getInstance();

    //get space service for installing operations
    SpaceService spaceService = (SpaceService) container.getComponentInstanceOfType(SpaceService.class);

    //install application for the space
    spaceService.removeApplication(spaceId, appId, appName);
}
```

3.4.4.3. Space's members management

SpaceService allows you to manage the spaces' members. Here is the way to add a new member to a space:

```
String spacePrettyName = "mySpace";

PortalContainer container = PortalContainer.getInstance();
SpaceService spaceService = (SpaceService) container.getComponentInstanceOfType(SpaceService.class);

Space space = service.getSpaceByPrettyName(spacePrettyName);
if (space != null) {
    spaceService.addMember(space, "mary");
}
```

3.4.4.4. Listener to a space lifecycle

To receive notifications of what are happening in spaces, you need to extend *SpaceListenerPlugin* and register it. Every method takes a *SpaceLifeCycleEvent* object as a parameter that contains the information about the event and its context.

The available events are:

- *spaceCreated*: This event is called right after a space is created successfully with its applications.
- *spaceRemoved*: This event is called right after the space is removed. It means all the applications of the space are removed, its group and group navigation are removed.
- *applicationAdded*: This event is called right after an application is added (installed) to the space.
- *applicationActivated*: This event is called right after an application is activated in the space.
- *applicationDeactivated*: This event is called right after an application is deactivated.
- *applicationRemoved*: This event is called right after an application is removed from the space.
- *joined*: This event is called right after a user joins the space.
- *left*: This event is called right after a space member leaves its space.
- *grantedLead*: This event is called right after a user is granted the space's manager role.

- *revokedLead*: This event is called right after the space's manager is revoked his role to be a space member.

```
import org.exoplatform.social.space.lifecycle.SpaceListenerPlugin;

public class MySpaceListenerPlugin extends SpaceListenerPlugin {
    ...
}
```

As an example, see [SpaceActivityPublisher](#) that publishes an activity based on an event that happened in a space.

To register your listener, configure it as a plugin to the SpaceService like this:

```
<external-component-plugins>
  <target-component>org.exoplatform.social.core.space.spi.SpaceService</target-component>
  <component-plugin>
    <name>SpaceActivityPublisher</name>
    <set-method>addSpaceListener</set-method>
    <type>org.mycompany.MySpaceListenerPlugin</type>
  </component-plugin>
</external-component-plugins>
```

3.4.5. Space widget tutorial

The eXo Social widget enables developers to add capabilities of eXo Social to external applications. Since the widget is hosted on your eXo Social server, it can display personalized information. An activity stream of the most recent user actions will display to the group's members.

There are two options of the eXo Social widget that provide different levels of integration and information.

The basic version of this widget is an iFrame. The more advanced version is a button you can insert in a page; this will display a small pop-up with information about the space.

3.4.5.1. Basic version

To insert the basic version, you need to have an iFrame to insert on your site.

```
<iframe scrolling="no" height="180" frameborder="no" width="220" src="http://URL_OF_YOUR_EXO_INSTALLATION.COM/
spaceName=NAME_OF_YOUR_SPACE&description=DESCRIPTION_OF_THE_SPACE"></iframe>
```

To install this version in your application, replace all the uppercase text below:

- *URL_OF_YOUR_EXO_INSTALLATION.COM*: This is the URL of your eXo Social installation. If you are testing on your local computer, the URL may be *localhost:8080*.
- *NAME_OF_YOUR_SPACE*: This is the title of your space in eXo Social. In the URL, it is necessary to avoid special characters. For the space name, you can only use alphanumeric characters and "_", ".", "-" or ":".
- *DESCRIPTION_OF_THE_SPACE*: This will be displayed in the list of spaces.

3.4.5.2. Advanced version

To install an advanced version of the widget, you need to insert a code snippet in your page. This includes some

HTMLs plus some JavaScript. The necessary CSS will be added dynamically.

Next, insert the following code at the position you want the button to be displayed:

```
<div class="exoSpacesContainer"><a href="javascript:void(0);" id="exoSpacesLink" class="exoSpacesLink" target="_blank">Space</a></div>
<script src="/socialWidgetResources/javascript/space.js"></script>
<script>spaces.createPopup("exoSpacesLink", "MyAppName - my social object", "my cool description");</script>
```

The important function here is:

```
spaces.createPopup(link, spaceName, description)
```

In which:

- *link* is the ID or the HTML element where the pop-up will be placed. If you copy and paste the code snippet provided above, you do not need to change this value.
- *spaceName* is the name of space. It is also used to identify the space. You should use the following format: "MyAppName - my social object".
- *description* is a brief introduction about your space.

3.4.5.3. Configure

- *serverURL* (Default: "<http://127.0.0.1:8080>"): The address of your eXo installation. To change it, use *spaces.setServerURL(...)*.
- *spaceServicePath* (Default: */rest/private/spaces/*): The path to the spaces service. It is rare you have to change it; but if needed, use *spaces.setSpaceServicePath(...)*.
- *portalName* (Default: *socialdemo*): The name of portal you are using. To change it, use *spaces.setPortalName(...)*.

If you want to change any part of this configuration, the best way is to change before creating the pop-up. For example:

```
<div class="exoSpacesContainer"><a href="#" id="exoSpacesLink" class="exoSpacesLink" target="_blank">Space</a></div>
<script src="/socialWidgetResources/javascript/space.js"></script>
<script>
  spaces.setServerURL("http://192.168.2.100:8080");
  spaces.createPopup("exoSpacesLink", "My cool new space", "my cool description");
</script>
```

You can see an example of integration at: <http://localhost:8080/socialWidgetResources/test.html>

3.4.6. How to extend the activities rendering

3.4.6.1. Objective

In this section, you will learn how to implement a preprocessor for activities rendering.

3.4.6.2. Requirements

To understand this section, the knowledge of Java and eXo Kernel (component model and its XML configuration) is prerequisite.

3.4.6.3. Why would you need to do this?

A simple activity is made of simple text. An extension point has been created at the level of activities rendering for two cases:

- support more HTML tags.
- support @mentions.

But you may want to support a special syntax, for example:

- #hashtags to feel like Twitter.
- smileys to look like Skype.
- [Markdown](#) to experience Buzz.

You can have more sophisticated cases to process, such as parsing the link that include in the activity's content. Because a process actually has the full access to the Activity, you can very well process based on the owner, app, and media item.

3.4.6.4. Write an ActivityProcessor

eXo Social lets you pre-process an activity before it is returned by the ActivityManager. To do this, you simply need to implement the interface ActivityProcessor:

```
/**
 * An activity processor is responsible to pre-process an activity before it is returned by the {@link ActivityManager}
 */
public interface ActivityProcessor {

    /**
     * Process an activity
     * @param activity the activity. It can be modified
     */
    void processActivity(Activity activity);

    /**
     * Priority of this processor.
     * All activity processors will be executed in ascending priority order
     * @return
     */
    int getPriority();
}
```

For example, the following shows you how to implement a *SmileyProcessor* that will replace text smileys by icons:

```
public class SmileyProcessor implements ActivityProcessor {

    String smiley = "<img src='/images/smiley.gif' />";
```

```

public void processActivity(Activity activity) {
    String title = activity.getTitle();
    activity.setTitle(title.replaceAll(":-)", smiley));
}

public int getPriority() {
    return 100;
}
}

```

3.4.6.5. Configure the processor

Now, you have a nice processor, so need to hook it to the system. At runtime, the processors can be attached to *ActivityManager* via the *addProcessor(ActivityProcessor)* method.

But there is also a component plugin hooked for it: *public void addProcessorPlugin(BaseActivityProcessorPlugin plugin)*.

So, to make your processor easy to hook, you simply need to let him extend the *BaseActivityProcessorPlugin*.

```

public class SmileyProcessor extends BaseActivityProcessorPlugin {

    String smiley = "<img src='/images/smiley.gif' />";

    public SmileyProcessor(InitParams params) {
        super(params);
    }

    public void processActivity(Activity activity) {
        String title = activity.getTitle();
        activity.setTitle(title.replaceAll(":-)", smiley));
    }
}

```

It will have the additional benefit to make the priority field configurable, so you do not need to implement *getPriority()*.

Then your processor can be configured as a component plugin like this:

```

<external-component-plugins>
  <target-component>org.exoplatform.social.core.activitystream.ActivityManager</target-component>
  <component-plugin>
    <name>SmileyProcessor</name>
    <set-method>addProcessorPlugin</set-method>
    <type>org.example.SmileyProcessor</type>
    <init-params>
      <value-param>
        <name>priority</name>
        <description>priority of this processor (lower are executed first)</description>
        <value>2</value>
      </value-param>
    </init-params>
  </component-plugin>
</external-component-plugins>

```

Restart, then place the smiley images on the server and you should see something like that:

3.4.7. XMLProcessor component

3.4.7.1. Objectives

This section shows you the way to change the content of input texts by using and extending the XMLProcessor component and its plugins.

3.4.7.2. XMLProcessor Component

This service processes the input texts in the system by pushing it through a filter (plugin) chain and returns a result as the diagram below:

Each filter is responsible for enriching the content of the input texts. For example, highlight usernames existing in a user's connection or remove the forbidden HTML tags.

The XMLProcessor component is configured in the `config/src/main/java/conf/social/common-configuration.xml` file:

```
<component>
  <key>org.exoplatform.social.common.xmlprocessor.XMLProcessor</key>
  <type>org.exoplatform.social.common.xmlprocessor.XMLProcessorImpl</type>
</component>
```

To manage the chain of the filters in XMLProcessor, you can use the `addFilterPlugin()` and `removeFilterPlugin()` methods. XMLProcessor is initialized by IOC (Inversion of Control) via the configuration files defined in the `/demo/war/src/main/webapp/WEB-INF/conf/socialdemo/social/component-plugins-configuration.xml` path.

Sample code:

```
<external-component-plugins>
  <target-component>org.exoplatform.social.common.xmlprocessor.XMLProcessor</target-component>
  <component-plugin>
    <name>URLConverterFilterPlugin</name>
    <set-method>addFilterPlugin</set-method>
    <type>org.exoplatform.social.common.xmlprocessor.filters.URLConverterFilterPlugin</type>
    <init-params>
      <value-param>
        <name>urlMaxLength</name>
        <description>the max length of URL</description>
        <value>-1</value>
      </value-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

3.4.7.3. Built-in XMLProcessor Plugins

In eXo Social, there are the following built-in XMLProcessor plugins (also known as **filters**) that filter the input texts of users.

Filters	Description
DOMContentEscapeFilter	Process the DOM tree input and escape all text nodes .
DOMLineBreakerFilter	Process the DOM tree input and add <code>
</code> to all text nodes which contain <code>{{{n}}}</code> .
DOMXMLTagFilter	Process the DOM tree input and convert all tag

Filters	Description
	nodes which do not exist in whitelist into text Node .
LineBreakerFilter	Process the String input and replace {{{n}}} to .
XMLBalancer	Process the String input and add missing close tags to input.
XMLTagFilter	Processes the String input and convert all tags which do not exist in whitelist into the escaped String.

The following is the general Class diagram of XMLProcessor in eXo Social:

All of these filters implements the Filter interface as follows:

```
package org.exoplatform.social.common.xmlprocessor;
public interface Filter {
    /**
     * Filters the input data.
     *
     * @param input the input data
     * @return an Object with the result after filtered
     */
    public Object doFilter(Object input);
}
```

These filters will process the input texts in the *doFilter(Object input)* method and return the result to XMLProcessor. They are initialized via the .xml files found in the */demo/war/src/main/webapp/WEB-INF/conf/socialdemo/social/component-plugins-configuration.xml* path.

```
<external-component-plugins>
  <target-component>org.exoplatform.social.common.xmlprocessor.model.XMLTagFilterPolicy</target-component>
  <component-plugin>
    <name>setAllowedTagPlugin</name>
    <set-method>setAllowedTagPlugin</set-method>
    <type>org.exoplatform.social.common.xmlprocessor.model.XMLTagFilterPolicy$AllowedTagPlugin</type>
    <init-params>
      <object-param>
        <name>img tag</name>
        <object type="org.exoplatform.social.common.xmlprocessor.model.XMLTagFilterPolicy$AllowedTag">
          <field name="tagName"><string>img</string></field>
          <field name="tagAttributes">
            <collection item-type="java.lang.String" type="java.util.HashSet">
              <value><string>src</string></value>
            </collection>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

You can write your own filter by implementing the [Filter interface](#) and add it to XMLProcessor as the [sample code](#) in the **XMLProcessor Component** section.

3.5. Public REST APIs

3.5.1. Activities REST service

Name	Service URL	Location	Description
ActivitiesRestService	{restContextName}/{portalName}	Name}/social/activities Maven groupId: org.exoplatform.social ArtifactId: exo.social.component.service	Provide REST services for activity applications: like/unlike; comment; delete activity.

- **API:**

Name	Service URL	Parameters	Expected Values	Description
destroyActivity	{restContextName}/{portalName}	portalName activityId format	activities/destroy/{activityId} String String String: json or xml	Destroy activity and get the JSON/XML format.
showLikes	{restContextName}/{portalName}	portalName activityId format	activities/{activityId}/likes String String String: json or xml	Show the list of likes by activityId and return the JSON/XML format.
updateLike	{restContextName}/{portalName}	portalName activityId format	activities/{activityId}/likes/update/{likeId} String String String: json or xml	Update the list of likes by the JSON/XML format.
destroyLike	{restContextName}/{portalName}	portalName activityId identityId format	activities/{activityId}/likes/destroy/{likeId} String String String String: json or xml	Destroy like by identityId and get the JSON/XML format return format.
showComments	{restContextName}/{portalName}	portalName	activities/{activityId}/likes/show/{commentId} String	Show the comment list by the

Name	Service Endpoint	URL	Parameters	Expected Values	Description
			activityId	String	JSON/XML format.
			format	String: json or xml	
updateComment	{restContextName}/	{portalName}/social/activities/{activityId}/likes/update. {portalName}	String	String	Update the comment by the JSON/XML format.
			activityId	String	
			format	String: json or xml	
destroyComment	{restContextName}/	{portalName}/social/activities/{activityId}/comments/destroy/{commentId}. {portalName}	String	String	Destroy/destroy the comment and return the JSON/XML format.
			activityId	String	
			commentId	String	
			format	String: json or xml	

Example:

<http://localhost:8080/rest-socialdemo/socialdemo/social/activities/s08d397dg6/likes/destroy/abc.json>

3.5.2. Apps REST service

Name	Service URL	Location	Description
AppsRestService	{restContextName}/social/apps/	Maven groupId: org.exoplatform.social ArtifactId: exo.social.component.service	Provide REST services for the application registry gadget: shows application list.

- **API:**

Name	Service Endpoint	URL	Parameters	Expected Values	Description
showApps	{restContextName}/social/apps/show.	{portalName}	String	String: json or xml	Show applications by the JSON/XML format.

Example:

<http://localhost:8080/rest-socialdemo/social/apps/show.json>

3.5.3. Identity REST service

Name	Service URL	Location	Description
IdentityRestService	{restContextName}/{portalName}	{restContextName}/social/identity/{username}/id/show.json Maven groupId: org.exoplatform.social ArtifactId: exo.social.component.service	Get identity by the username.

• API:

Name	Service Endpoint	Parameters	Expected Values	Description
UserId getId	{restContextName}/{portalName}	{restContextName}/social/identity/{username}/id/show.json username portalName	String String	Get user identity by username and return by the JSON format.

Example:

<http://localhost:8080/rest-socialdemo/socialdemo/social/identity/john/id/show.json>

3.5.4. Linkshare REST service

Name	Service URL	Location	Description
LinkshareRestService	{restContextName}/social/linkshare	{restContextName}/social/linkshare Maven groupId: org.exoplatform.social ArtifactId: exo.social.component.service	Get information from a provided link.

• API:

Name	Service Endpoint	Parameters	Expected Values	Description
getLink	{restContextName}/social/linkshare/show.{format}	link	{format} json or xml	Get the link content by posting a linkShare request.

Example:

<http://localhost:8080/rest-socialdemo/social/linkshare/show.json>

3.5.5. People Rest Service

Name	Service URL	Location	Description
PeopleRestService	{restContextName}/social/people	Maven groupId: org.exoplatform.social ArtifactId: exo.social.component.service	Provide REST services for manipulating jobs related to people.

• **API:**

Name	Service URL Endpoint	Parameters	Expected Values	Description
suggestUsernames	{restContextName}/social/people/suggest.	{format} nameToSearch currentUser typeOfRelation spaceURL format	String String String String String: json or xml	Get and return a list of usernames which match the input string for suggest.

Example: <http://localhost:8080/rest-socialdemo/social/people/suggest.json>

3.5.6. Spaces REST service

Name	Service URL	Location	Description
SpacesRestService	{restContextName}/{portalName}/social/spaces	Maven groupId: org.exoplatform.social ArtifactId: exo.social.component.service	Provide REST services for space gadget to display users' spaces and pending spaces.

• **API:**

Name	Service URL Endpoint	Parameters	Expected Values	Description
showMySpaceList	{restContextName}/social/spaces/mySpaces	portalName format	show.{format} String String: json or xml	Show mySpaceList by the JSON/XML format.
showPendingSpaceList	{restContextName}/social/spaces/pendingSpaces	portalName format	show.{format} String String: json or xml	Show pendingSpaceList by the JSON/XML format.
suggestSpacenames	{restContextName}/social/spaces/spaceNames	portalName conditionToSearch typeOfRelation currentUser format	suggest.{format} String String String String: json or xml	Get and return space's names that match the input string for suggest.

Example:

<http://localhost:8080/rest-socialdemo/social/spaces/mySpaces/show.xml>

3.5.7. Widget Rest Service

Name	Service URL	Location	Description
WidgetRestService	{restContextName}/spaces	{containerName} Maven groupId: org.exoplatform.social ArtifactId: exo.social.extras.widget.rest	Provide REST services for creating spaces or getting spaces'information.

• API:

Name	Service URL Endpoint	Parameters	Expected Values	Description
spaceInfo	{restContextName}/spaces/{containerName}	containerName portalName	space_info String String (default	Return the HTML page for displaying the information of the space. Two

Name	Service Endpoint	URL	Parameters	Expected Values	Description
			spacePrettyName	value: classic)	query parameters needed: <i>spaceName</i> and <i>description</i> .
			description	String	
				String	

Example:

http://localhost:8080/rest-socialdemo/spaces/socialdemo/space_info?name=Social&description=Social

3.5.8. Location

- **Maven groupId:** org.exoplatform.social
- **ArtifactId:** exo.social.component.service

3.6. Rest Service APIs (v1-alpha3)

Objectives:

The third parties want to integrate and extend the eXo Social capability and features and eXo Social REST Services APIs are dedicated for them to complete that. By using these REST APIs, the third parties can write any applications (desktop apps, mobile apps, web apps) to integrate with eXo Social services and business.

Conventions:

The entry point for Social Rest service must be: `/{rest_context_name}/private/api/social/{version}/{portalContainerName}/{social_resources}/`. An example of activities resources: `/rest/private/api/social/v1-alpha1/portal/activity/`.

There are 3 types of parameters on each URL:

- **{server_params}**: this is the unchanged parameter for a specified server.
- **:id**: this param is for a specified data object, for example, activity Id and identity Id.
- **format**: this is the supported data format. It can be JSON, XML, RSS, or ATOM.

Notes:

Currently, only the basic authentication is supported. Please see more details at http://en.wikipedia.org/wiki/Basic_access_authentication.

The JSON support is mandatory. The other format SHOULD be supported too (XML, RSS, ATOM). The supported formats will be specified by the detailed documentation for the REST resources.

The *rest_context_name* and *portal_container_name* parameters are used as follows:

- On eXo Social standalone:
 - *rest_context_name*: *rest-socialdemo*;
 - *portal_container_name*: *socialdemo*;
- On eXo Platform:
 - *rest_context_name*: *rest*;
 - *portal_container_name*: *portal*;

3.6.1. Activity Resources

3.6.1.1. activity

3.6.1.1.1. POST activity.format

Description: Creates an activity to an identity's activity stream. If no *identity_id* is specified, the activity will be created to the authenticated identity's activity stream.

URL:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity.format
```

For example:

- <http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity.json>
- <http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity.json>

Body Data: The data for an activity

For example:

```
{
  "title": "Hello World", //required
  "type": "exosocial:core", //optional
  "priority": 0.5, //optional
  "titleId": "", //optional
  "templateParams": {} //optional
}
```

Supported Format: json

Requires Authentication: true

Parameters:

- **Optional**

- **identity_id**: The identity who posts activities to his stream, to the stream of his connection or the space stream where he is a manager or a member.

Examples

- JSON

Request:

```
POST: http://localhost:8080/rest/private/api/social/v1-alpha3/socialdemo/activity.json
HEADER: Content-Type: application/json
BODY: {"title": "Hello World!!!"}
```

Response: A list of activity objects in the JSON format.

```
{
  "appId": "",
  "identityId": "423c28dac0a80137186cb6150f481c8c",
  "totalNumberOfComments": 0,
  "liked": false,
  "templateParams": {},
  "postedTime": 1315383441216,
  "type": "",
  "posterIdentity": {},
  "activityStream": "",
  "id": "42f6786cc0a80137337a0b28ebe6af73",
  "title": "Hello World!!!",
  "priority": 0,
  "createdAt": "Wed Sep 7 15:17:21 +0700 2011",
  "likedByIdentities": [],
  "titleId": "",
  "body": "",
  "comments": []
}
```

3.6.1.1.2. GET activity/:activityId.format

Description: Gets an activity object from a specified activity Id.

URL:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity/:activityId
```

For example:

- <http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity/1a2b3c4d5e6f7g8h9i.json>
- <http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4d5e6f7g8h9i.json>

Supported Format: json

Requires Authentication: true

Parameters:

- **Required**

- ### Examples:

- Request:**

```
GET: http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4d
```

```
{
  "id": "1a2b3c4d5e6f7g8h9j",
  "title": "Hello World!!!",
  "appId": "",
  "type": "DEFAULT_ACTIVITY",
  "postedTime": 123456789, //timestamp
  "createdAt": "Fri Jun 17 06:42:26 +0000 2011", //The Date follows ISO 8601
  "priority": 0.5, //between 0.0 and 1.0, higher value => higher priority.
  "templateParams": {},
  "titleId": "",
  "identityId": "123456789abcdefghi", //the identity id of the user who created this activity
  "liked": true, //is liked (favorites) by this authenticated identity
  "likedByIdentities": [{"id":1234567, "providerId":"organization", "remoteId":"demo", "profile":{"fullNam
  "totalNumberOfLikes": 20
  "posterIdentity": {"id":1234567, "providerId":"organization", "remoteId":"demo", "profile":{"fullNam
  "comments": [{}], //optional
  "totalNumberOfComments": 1234,
  "activityStream": {
    "type": "user", // or "space"
    "prettyId": "root", // or space_abcde
    "fullName": "Root Root",
    "faviconUrl": "http://demo3.exoplatform.org/favicons/exo-default.jpg",
    "title": "Activity Stream of Root Root",
    "body": "",
    "permalink": "http://localhost:8080/profile/root"
  } //optional
}
```

3.6.1.1.3. DELETE activity/:activityId.format

Description: Deletes an existing activity by its Id using the DELETE method. The deleted activity information will be returned in the specified format.

URL:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity/:activityId.format
```

For example:

- <http://platform35.demo.exoplatfrom.org/rest/private/api/social/v1-alpha3/portal/activity/1a2b3c4d5e6f7g8h9i.json>
- <http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4d5e6f7g8h9i.json>

Supported Format: json

Requires Authentication: true

Parameters:

- **Required**
 - **id:** The Id of the existing activity.

Examples

- **JSON**

Request:

```
DELETE: http://platform35.demo.exoplatfrom.org/rest/private/api/social/v1-alpha3/portal/activity/1a2b3c4d5e6f7g8h9i.json
```

Response:

```
{
  "id": "1a2b3c4d5e6f7g8h9i"
}
```

3.6.1.1.4. POST activity/destroy/:activityId.format

Description: Deletes an existing activity by its Id using the POST method. The deleted activity information will be returned in the specified format. It is recommended to use the DELETE method, except the case that clients cannot make request via this method.

URL:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity/destroy/:activityId.format
```

For example:

<http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity/destroy/1a2b3c4d5e6f7g8h9i.json>

<http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/destroy/1a2b3c4d5e6f7g8h9i.json>

Supported Format: json

Requires Authentication: true

Parameters: No

Examples:

- JSON

Request:

```
POST: http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/destroy/1a2b3c4d5e6f7g8h9i.json
```

Response:

```
{
  "id": "1a2b3c4d5e6f7g8h9i"
}
```

3.6.1.2. activity/:activityId/comments

3.6.1.2.1. GET activity/:activityId/comments.format

Description: Gets all comments on an activity.

URL:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity/:activityId/comments.format
```

For example:

- <http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity/1a2b3c4d5e/comments.json>
- <http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4e5e/comments.json>

Supported Format: json

Requires Authentication: true

Parameters: No

Examples:

- JSON

Request:

```
GET: http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4d5e/comments.
```

Response: A list of comment objects in the JSON format.

```
{
  totalNumberOfComments: 2,
  comments: [
    {
      "id": "123456"
      "posterIdentity": {"id":123456, "providerId":"organization", "remoteId":"demo", "profile":{"fullName":"Demo
      "text": "Comment there!",
      "postedTime": 123456789,
      "createdAt": "Fri Jun 17 06:42:26 +0000 2011"
    },
    {
      "id" : "234567"
      "posterIdentity": {"id":123456, "providerId":"organization", "remoteId":"demo", "profile":{"fullName":"Demo
      "text": "Comment there 2!",
      "postedTime": 123456789,
      "createdAt": "Fri Jun 17 06:42:26 +0000 2011"
    }
  ]
}
```

3.6.1.3. activity/:activityId/comment

3.6.1.3.1. POST activity/:activityId/comment.format

Description: Posts a new comment on an existing activity. The poster of this comment is an authenticated identity.

URL:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity/:activity
```

For example:

- <http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity/1a2b3c4d5e/comment.json>
- <http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4e5e/comment.json>

Body data:

```
{
  "text": "My comment here!!!"
}
```

Supported Format: json

Requires Authentication: true

Parameters: No

Examples:

- JSON

Request:

```
POST: http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4d5e/comment.
HEADER: Content-Type: application/json
BODY: {"text": "My comment here!!!"}
```

Response: The created comment.

```
{
  "id": "123456"
  "posterIdentity": {"id":123456, "providerId":"organization", "remoteId":"demo", "profile":{"fullName":"Demo GTN"
  "text": "My comment here!!!",
  "postedTime": 123456789,
  "createdAt": "Fri Jun 17 06:42:26 +0000 2011"
}
```

3.6.1.3.2. DELETE activity/:activityId/comment/:commentId.format

Description: Deletes an existing comment by its Id.

URL:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity/:activity
```

For example:

- <http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity/1a2b3c4d5e/comment/123456.js>
- <http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4e5e/comment/123456.json>

Supported Format: json

Requires Authentication: true

Parameters: No

Examples:

- JSON

Request:

```
DELETE: http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4d5e/commer
```

Response: The deleted comment.

```
{
  "id": "123456"
}
```

3.6.1.3.3. POST activity/:activityId/comment/destroy/:commentId.format

Description: Deletes an existing comment by its Id using the POST method. The deleted activity information will be returned in a specified format. It is recommended to use the DELETE method, except the case that clients cannot make request via this method.

URL:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity/:activityId
```

For example:

- <http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity/1a2b3c4d5e/comment/destroy/123456>
- <http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4e5e/comment/destroy/123456>

Supported Format: json

Requires Authentication: true

Parameters: No

Examples:

- JSON

Request:

```
POST: http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4e5e/comment/destroy/123456
```

Response: The deleted comment

```
{
  "id": "123456"
}
```

3.6.1.4. activity/:activityId/likes

3.6.1.4.1. GET activity/:activityId/likes.format

Description: Gets all the identities who like an existing activity.

URL:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity/:activityId/likes
```

For example:

- <http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity/1a2b3c4d5e/likes.json>
- <http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4e5e/likes.json>

Supported Format: json

Requires Authentication: true

Parameters: No

Examples:

- JSON

Request:

```
GET: http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4d5e/likes.json
```

Response:

```
{
  totalNumberOfLikes: 2,
  likesByIdentities: [
    {
      "id": 1234567,
      "providerId": "organization",
      "remoteId": "demo",
      "profile": {
        "fullName": "Demo GTN",
        "avatarUrl": "http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
      }
    },
    {
      "id": 23456,
      "providerId": "organization",
      "remoteId": "root",
      "profile": {
        "fullName": "Root GTN",
        "avatarUrl": "http://localhost:8080/profile/u/root/avatar.jpg?u=12345"
      }
    }
  ]
}
```

3.6.1.5. activity/:activityId/like

3.6.1.5.1. POST activity/:activityId/like.format

Description: Allows an authenticated identity to do the "like" action on an existing activity.

URL:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity/:activityId/like.format
```

For example:

- <http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity/1a2b3c4d5e/like.json>
- <http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4e5e/like.json>

Supported Format: json

Requires Authentication: true

Parameters: No

Examples:

- JSON

Request:

```
http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4e5e/like.json
```

Response:

```
{
  "liked": true
}
```

3.6.1.5.2. DELETE activity/:activityId/like.format

Description: Allows an identity to remove his "like" action on an activity.

URL:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity/:activityId/like.format
```

For example:

- <http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity/1a2b3c4d5e/like.json>
- <http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4e5e/like.json>

Supported Format: json

Requires Authentication: true

Parameters: No

Examples:

- JSON

Request:

```
DELETE: http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4e5e/like.json
```

Response:

```
{
  "liked": false
}
```

3.6.1.5.3. POST activity/:activityId/like/destroy.format

Description: Allows an identity to remove his "like" action on an activity. It is recommended to use the DELETE method, except the case that clients cannot make request via this method.

URL:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity/:activity
```

For example:

- <http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity/1a2b3c4d5e/like/destroy.json>
- <http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4e5e/like/destroy.json>

Supported Format: json

Requires Authentication: true

Parameters: No

Examples:

- JSON

Request:

```
POST: http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity/1a2b3c4e5e/like/destroy.json
```

Response:

```
{
  "liked": false
}
```

3.6.2. Activity Stream Resources

3.6.2.1. activity_stream/:identityId

3.6.2.1.1. GET activity_stream/:identityId.format

Description: Gets activities of a defined identity which can be a user identity, a space identity, or any type of identities. There is one special *identityId* called "me" which stands for the authenticated user who makes this request.

Url:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity_stream/:identityId
```

For example:

- http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity_stream/1234567890.json
- http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity_stream/1234567890.json?limit=10

- http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity_stream/f92cd6f0c0a80137102696

Supported Format: json

Requires Authentication: true

Parameters:

- **Optional**

- **limit:** The number of activities retrieved with the default value of 100. This input value must be less than or equal to its default value (100). The number of the returned results is actually less than or equal to the **limit** value.
- **since_id:** Returns the activities having the created timestamps greater than the specified **since_id** 's *created timestamp*.
- **max_id:** Returns the activities having the created timestamps less than the specified **max_id**'s created timestamp. Note that **since_id** and **max_id** must not be defined in one request, if they are, the **since_id** query param is chosen.
- **number_of_comments:** Specifies the number of latest comments to be displayed along with each activity. By default, **number_of_comments=0**. If **number_of_comments** is a positive number, this number is considered as a limit number that must be equal or less than 100. If the total number of comments is less than the provided positive number, the number of actual comments must be returned. If the total number of comments is more than 100, it is recommended to use *activity/:activityId/comments.format* instead.
- **number_of_likes:** Specifies the number of latest detailed likes to be returned along with this activity. By default, **number_of_likes=0**. If **number_of_likes** is a positive number, this number is considered as a limit number that must be equal or less than 100. If the total number of likes is less than the provided positive number, the number of actual likes must be returned. If the total number of likes is more than 100, it is recommended to use *activity/:activityId/likes.format* instead.

Examples:

- **JSON**

Request

Response:

```
GET: http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity_stream/f92cd6f0c0a80137102696
```

- **Response**

```
{
  "activities": [
    {
      "id": "1a2b3c4d5e6f7g8h9j",
      "title": "Hello World!!!",
      "appId": "",
      "type": "DEFAULT_ACTIVITY",
      "postedTime": 123456789,
      "createdAt": "Fri Jun 17 06:42:26 +0000 2011",

```

```

"priority":0.5,
"templateParams":{

},
"titleId": "",
"body": "",
"identityId":"123456789abcdefghi",
"liked":true,
"likedByIdentities":[
  {
    "id":"123456313efghi",
    "providerId":"organization",
    "remoteId":"demo",
    "profile":{
      "fullName":"Demo GTN",
      "avatarUrl":"http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
    }
  }
],
"totalNumberOfLikes":20,
"posterIdentity":{
  "id":"123456313efghi",
  "providerId":"organization",
  "remoteId":"demo",
  "profile":{
    "fullName":"Demo GTN",
    "avatarUrl":"http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
  }
},
"comments":[
  {

  }
],
"totalNumberOfComments":1234,
"activityStream":{
  "type":"user",
  "prettyId":"root",
  "fullName": "Root Root",
  "faviconUrl":"http://demo3.exoplatfrom.org/favicons/exo-default.jpg",
  "title":"Activity Stream of Root Root",
  "permalink":"http://localhost:8080/profile/root"
}
},
{
  "id":"1a210983123f7g8h9j",
  "title":"Hello World 1!!!",
  "appId":"",
  "type":"DEFAULT_ACTIVITY",
  "postedTime":123456789,
  "createdAt":"Fri Jun 19 06:42:26 +0000 2011",
  "priority":0.5,
  "templateParams":{

  },
  "titleId":"",
  "body": "",
  "identityId":"123456789abcdefghi",
  "liked":true,
  "likedByIdentities":[
    {
      "id":"123456313efghi",
      "providerId":"organization",
      "remoteId":"demo",
      "profile":{
        "fullName":"Demo GTN",
        "avatarUrl":"http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
      }
    }
  ]
},
"totalNumberOfLikes":20,
"posterIdentity":{
  "id":"123456313efghi",
  "providerId":"organization",
  "remoteId":"demo",
  "profile":{
    "fullName":"Demo GTN",
    "avatarUrl":"http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
  }
}

```

```

    }
  },
  "comments": [
    {
      }
    ],
    "totalNumberOfComments": 1234,
    "activityStream": {
      "type": "user",
      "prettyId": "root",
      "fullName": "Root Root",
      "faviconUrl": "http://demo3.exoplatform.org/favicons/exo-default.jpg",
      "title": "Activity Stream of Root Root",
      "permalink": "http://localhost:8080/profile/root"
    }
  ]
}

```

3.6.2.2. activity_stream/feed

3.6.2.2.1. GET activity_stream/feed.format

Description: Gets the activity stream feed of the authenticated user identity who makes this request.

URL:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity_stream/feed
```

For example:

- http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity_stream/feed.json?limit=20&since_id=123456&max_id=123456
- http://localhost:8080/rest/private/api/social/v1-alpha3/socialdemo/activity_stream/feed.json?limit=20&max_id=123456&since_id=123456

Supported Format: json

Requires Authentication: true

Parameters:

- **Optional**
 - **limit:** Specifies the number of activities to be retrieved. This number must be less than or equal to **100**. The value you pass to **limit** is a maximum number of activities to be returned. the total number of activities you receive maybe less than **limit**. If **limit** is not specified, the default value will be 100.
 - **since_id:** Returns the activities having the created timestamps greater than the specified **since_id**'s created timestamp.
 - **max_id:** Returns the activities having the created timestamps less than the specified **max_id**'s created timestamp. Note that **since_id** and **max_id** must not be defined in one request, if they are, the **since_id** query param is chosen.
 - **number_of_comments:** Specifies the number of latest comments to be displayed along with each activity. By default, **number_of_comments=0**. If **number_of_comments** is a positive number, this number is considered as a limit number that must be equal or less than 100. If the total number of comments is less

than the provided positive number, the number of actual comments must be returned. If the total number of comments is more than 100, it is recommended to use *activity/:activityId/comments.format* instead.

- **number_of_likes**: Specifies the number of latest detailed likes to be returned along with this activity. By default, **number_of_likes=0**. If **number_of_likes** is a positive number, this number is considered as a limit number that must be equal or less than 100. If the total number of likes is less than the provided positive number, the number of actual likes must be returned. If the total number of likes is more than 100, it is recommended to use *activity/:activityId/likes.format* instead.

Examples

- **JSON**

Request:

```
GET: http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity_stream/feed.json?lim
```

- **Response**

```
{
  "activities":[
    {
      "id":"1a2b3c4d5e6f7g8h9j",
      "title":"Hello World!!!",
      "appId":"",
      "type":"DEFAULT_ACTIVITY",
      "postedTime":123456789,
      "createdAt":"Fri Jun 17 06:42:26 +0000 2011",
      "priority":0.5,
      "templateParams":{

      },
      "titleId":"",
      "body": "",
      "identityId":"123456789abcdefghi",
      "liked":true,
      "likedByIdentities":[
        {
          "id":"123456313efghi",
          "providerId":"organization",
          "remoteId":"demo",
          "profile":{
            "fullName":"Demo GTN",
            "avatarUrl":"http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
          }
        }
      ],
      "totalNumberOfLikes":20,
      "posterIdentity":{
        "id":"123456313efghi",
        "providerId":"organization",
        "remoteId":"demo",
        "profile":{
          "fullName":"Demo GTN",
          "avatarUrl":"http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
        }
      },
      "comments":[
        {

        }
      ],
      "totalNumberOfComments":1234,
      "activityStream":{
```

```

        "type": "user",
        "prettyId": "root",
        "fullName": "Root Root",
        "faviconUrl": "http://demo3.exoplatform.org/favicons/exo-default.jpg",
        "title": "Activity Stream of Root Root",
        "permalink": "http://localhost:8080/profile/root"
    }
},
{
    "id": "1a210983123f7g8h9j",
    "title": "Hello World 1!!!",
    "appId": "",
    "type": "DEFAULT_ACTIVITY",
    "postedTime": 123456789,
    "createdAt": "Fri Jun 19 06:42:26 +0000 2011",
    "priority": 0.5,
    "templateParams": {

    },
    "titleId": "",
    "body": "",
    "identityId": "123456789abcdefghi",
    "liked": true,
    "likedByIdentities": [
        {
            "id": "123456313efghi",
            "providerId": "organization",
            "remoteId": "demo",
            "profile": {
                "fullName": "Demo GTN",
                "avatarUrl": "http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
            }
        }
    ],
    "totalNumberOfLikes": 20,
    "posterIdentity": {
        "id": "123456313efghi",
        "providerId": "organization",
        "remoteId": "demo",
        "profile": {
            "fullName": "Demo GTN",
            "avatarUrl": "http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
        }
    },
    "comments": [
        {

        }
    ],
    "totalNumberOfComments": 1234,
    "activityStream": {
        "type": "user",
        "prettyId": "root",
        "fullName": "Root Root",
        "faviconUrl": "http://demo3.exoplatform.org/favicons/exo-default.jpg",
        "title": "Activity Stream of Root Root",
        "permalink": "http://localhost:8080/profile/root"
    }
}
]
}

```

3.6.2.3. activity_stream/connections

3.6.2.3.1. GET activity_stream/connections.format

Description: Gets activities of connections of the authenticated user identity who makes this request.

URL:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity_stream/co
```

For example:

- http://platform35.demo.exoplatfrom.org/rest/private/api/social/v1-alpha3/portal/activity_stream/connections.json
- http://platform35.demo.exoplatfrom.org/rest/private/api/social/v1-alpha3/portal/activity_stream/connections.json?limit=2

Supported Format: json

Requires Authentication: true

Parameters:

- **Optional:**

- **limit:** The number of activities retrieved with the default value of 100. This input value must be less than or equal to its default value (100). The number of the returned results is actually less than or equal to the **limit** value.
- **since_id:** Returns the activities having the created timestamps greater than the specified **since_id**'s created timestamp.
- **max_id:** Returns the activities having the created timestamps less than the specified **max_id**'s created timestamp. Note that **since_id** and **max_id** must not be defined in one request, if they are, the **since_id** query param is chosen.
- **number_of_comments:** Specifies the number of latest comments to be displayed along with each activity. By default, **number_of_comments=0**. If **number_of_comments=0** is a positive number, this number is considered as a limit number that must be equal or less than 100. If the total number of comments is less than the provided positive number, the number of actual comments must be returned. If the total number of comments is more than 100, it is recommended to use *activity/:activityId/comments.format* instead.
- **number_of_likes:** Specifies the number of latest detailed likes to be returned along with this activity. By default, **number_of_likes=0**. If **number_of_likes** is a positive number, this number is considered as a limit number that must be equal or less than 100. If the total number of likes is less than the provided positive number, the number of actual likes must be returned. If the total number of likes is more than 100, it is recommended to use *activity/:activityId/likes.format* instead.

Examples:

- **JSON**

Request:

```
GET: http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity_stream/connections.json
```

Response:

```
{
{
  "activities":[
    {
      "id":"1a2b3c4d5e6f7g8h9j",
      "title":"Hello World!!!",
      "appId":"","
```

```

"type":"DEFAULT_ACTIVITY",
"postedTime":123456789,
"createdAt":"Fri Jun 17 06:42:26 +0000 2011",
"priority":0.5,
"templateParams":{
},
"titleId":"",
"body": "",
"identityId":"123456789abcdefghi",
"liked":true,
"likedByIdentities":[
{
  "id":"123456313efghi",
  "providerId":"organization",
  "remoteId":"demo",
  "profile":{
    "fullName":"Demo GTN",
    "avatarUrl":"http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
  }
}
],
"totalNumberOfLikes":20,
"posterIdentity":{
  "id":"123456313efghi",
  "providerId":"organization",
  "remoteId":"demo",
  "profile":{
    "fullName":"Demo GTN",
    "avatarUrl":"http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
  }
},
"comments":[
{
}
],
"totalNumberOfComments":1234,
"activityStream":{
  "type":"user",
  "prettyId":"root",
  "fullName": "Root Root",
  "faviconUrl":"http://demo3.exoplatform.org/favicons/exo-default.jpg",
  "title":"Activity Stream of Root Root",
  "permalink":"http://localhost:8080/profile/root"
}
},
{
  "id":"1a210983123f7g8h9j",
  "title":"Hello World 1!!!",
  "appId":"",
  "type":"DEFAULT_ACTIVITY",
  "postedTime":123456789,
  "createdAt":"Fri Jun 19 06:42:26 +0000 2011",
  "priority":0.5,
  "templateParams":{
  },
  "titleId":"",
  "body": "",
  "identityId":"123456789abcdefghi",
  "liked":true,
  "likedByIdentities":[
    {
      "id":"123456313efghi",
      "providerId":"organization",
      "remoteId":"demo",
      "profile":{
        "fullName":"Demo GTN",
        "avatarUrl":"http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
      }
    }
  ],
  "totalNumberOfLikes":20,
  "posterIdentity":{
    "id":"123456313efghi",
    "providerId":"organization",
    "remoteId":"demo",
  }
}

```

```

    "profile":{
      "fullName":"Demo GTN",
      "avatarUrl":"http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
    },
    "comments":[
      {
      }
    ],
    "totalNumberOfComments":1234,
    "activityStream":{
      "type":"user",
      "prettyId":"root",
      "fullName": "Root Root",
      "faviconUrl":"http://demo3.exoplatform.org/favicons/exo-default.jpg",
      "title":"Activity Stream of Root Root",
      "permaLink":"http://localhost:8080/profile/root"
    }
  ]
}

```

3.6.2.4. activity_stream/spaces

3.6.2.4.1. GET activity_stream/spaces.format

Description: Gets activities of spaces in which the authenticated user identity is a member or a manager making this request.

URL:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/activity_stream/spaces
```

For example:

- http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity_stream/spaces.json
- http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/activity_stream/spaces.json?limit=20

Supported Format: json

Requires Authentication: true

Parameters:

- **Optional**

- **limit:** The number of activities retrieved with the default value of 100. This input value must be less than or equal to its default value (100). The number of the returned results is actually less than or equal to the **limit** value.
- **since_id:** Returns the activities having the created timestamps greater than the specified *since_id's created timestamp*.
- **max_id:** Returns the activities having the created timestamps less than the specified **max_id's** created timestamp. Note that **since_id** and **max_id** must not be defined in one request, if they are, the **since_id** query param is chosen.

- **number_of_comments**: Specifies the number of latest comments to be displayed along with each activity. By default, **number_of_comments=0**. If **number_of_comments** is a positive number, this number is considered as a limit number that must be equal or less than 100. If the total number of comments is less than the provided positive number, the total number of comments must be returned. If the total number of comments is more than 100, it is recommended to use *activity/:activityId/comments.format* instead.
- **number_of_likes**: Specifies the number of latest detailed likes to be returned along with this activity. By default, **number_of_likes=0**. If **number_of_likes** is a positive number, this number is considered as a limit number that must be equal or less than 100. If the total number of likes is less than the provided positive number, the total number of likes must be returned. If the total number of likes is more than 100, it is recommended to use *activity/:activityId/likes.format* instead.

Examples:**• JSON****Request:**

```
GET: http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/activity_stream/spaces.json?l
```

Response:

```
{
  "activities":[
    {
      "id":"1a2b3c4d5e6f7g8h9j",
      "title":"Hello World!!!",
      "appId":"",
      "type":"DEFAULT_ACTIVITY",
      "postedTime":123456789,
      "createdAt":"Fri Jun 17 06:42:26 +0000 2011",
      "priority":0.5,
      "templateParams":{

      },
      "titleId":"",
      "body": "",
      "identityId":"123456789abcdefghi",
      "liked":true,
      "likedByIdentities":[
        {
          "id":"123456313efghi",
          "providerId":"organization",
          "remoteId":"demo",
          "profile":{
            "fullName":"Demo GTN",
            "avatarUrl":"http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
          }
        }
      ],
      "totalNumberOfLikes":20,
      "posterIdentity":{
        "id":"123456313efghi",
        "providerId":"organization",
        "remoteId":"demo",
        "profile":{
          "fullName":"Demo GTN",
          "avatarUrl":"http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
        }
      },
      "comments":[
        {

        }
      ]
    }
  ]
}
```

```

    ],
    "totalNumberOfComments":1234,
    "activityStream":{
      "type":"user",
      "prettyId":"root",
      "fullName": "Root Root",
      "faviconUrl":"http://demo3.exoplatfrom.org/favicons/exo-default.jpg",
      "title":"Activity Stream of Root Root",
      "permaLink":"http://localhost:8080/profile/root"
    }
  },
  {
    "id":"1a210983123f7g8h9j",
    "title":"Hello World 1!!!",
    "appId":"",
    "type":"DEFAULT_ACTIVITY",
    "postedTime":123456789,
    "createdAt":"Fri Jun 19 06:42:26 +0000 2011",
    "priority":0.5,
    "templateParams":{

    },
    "titleId":"",
    "body": "",
    "identityId":"123456789abcdefghi",
    "liked":true,
    "likedByIdentities":[
      {
        "id":"123456313efghi",
        "providerId":"organization",
        "remoteId":"demo",
        "profile":{
          "fullName":"Demo GTN",
          "avatarUrl":"http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
        }
      }
    ],
    "totalNumberOfLikes":20,
    "posterIdentity":{
      "id":"123456313efghi",
      "providerId":"organization",
      "remoteId":"demo",
      "profile":{
        "fullName":"Demo GTN",
        "avatarUrl":"http://localhost:8080/profile/u/demo/avatar.jpg?u=12345"
      }
    },
    "comments":[
      {

      }
    ],
    "totalNumberOfComments":1234,
    "activityStream":{
      "type":"user",
      "prettyId":"root",
      "fullName": "Root Root",
      "faviconUrl":"http://demo3.exoplatfrom.org/favicons/exo-default.jpg",
      "title":"Activity Stream of Root Root",
      "permaLink":"http://localhost:8080/profile/root"
    }
  }
]
}

```

3.6.3. Identity Resources

3.6.3.1. identity



Note

Avatar link

If the avatar is not uploaded yet by users, the default avatar link will be returned and this link must be an absolute link.

3.6.3.1.1. GET identity/:providerId:remoteId.format

Description: Gets the identity and its associated profile by specifying its *providerId* and *remoteId*. Every identity has its *providerId* and *remoteId*. There could be as many identities as possible. Currently, there are 2 built-in types of identities (user identities and space identities) in eXo Social.

- **user identities:** *providerId* = organization; *remoteId* = portal user name.
- **space identities:** *providerId* = space; *remoteId* = space's pretty name.

Url:

```
http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/identity/:provider
```

For example:

- <http://platform35.demo.exoplatform.org/rest/private/api/social/v1-alpha3/portal/identity/organization/demo.json>
- <http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/identity/organization/demo.json>
- <http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/identity/space/aaaa.json>

Supported Format: json

Requires Authentication: true

Parameters: No

Examples:

- JSON

Request:

```
GET: http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/identity/organization/demo.js
```

Response:

```
{
  "id" : "123456789",
  "providerId": "organization",
  "remoteId": "demo",
  "profile": {
    "fullName": "Demo Gtn",
    "avatarUrl": "http://localhost:8080/socialdemo/demo/profile/avatar/demo.jpg"
  }
}
```

3.6.3.1.2. GET: identity/:identityId.format

Description: Gets the identity and its associated profile by the identityId.

Url:

http://{domain_name}/{rest_context_name}/private/api/social/{version}/{portal_container_name}/identity/:identity

For example:

- <http://platform35.demo.exopatform.org/rest/private/api/social/v1-alpha3/portal/identity/387cab38c0a8013768f5a753041b>
- <http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/identity/387cab38c0a8013768f5a753041b>

Supported Format: json

Requires Authentication: false

Parameters: No

Examples:

- **JSON**

Request:

```
GET: http://localhost:8080/rest-socialdemo/private/api/social/v1-alpha3/socialdemo/identity/123456789.json
```

Response:

```
{
  "id" : "123456789",
  "providerId": "organization",
  "remoteId": "demo",
  "profile": {
    "fullName": "Demo Gtn",
    "avatarUrl": "http://localhost:8080/socialdemo/demo/profile/avatar/demo.jpg"
  }
}
```

3.6.4. Version Resources

3.6.4.1. version/latest

3.6.4.1.1. GET version/latest.format

Description: Gets the latest eXo Social REST API version. This version number should be used as the latest and stable version that is considered to include all new features and updates of eXo Social REST services.

Url:

http://{domain_name}/{rest_context_name}/api/social/version/latest.format

For example:

- <http://platform35.demo.exoplatform.org/rest/api/social/version/latest.json>
- <http://localhost:8080/rest-socialdemo/api/social/version/latest.json>

Supported Format: json

Requires Authentication: false

Parameters: No

For example:

- **JSON**

Request:

```
GET: http://platform35.demo.exoplatform.org/rest/api/social/version/latest.json
```

Response:

```
{"version": "v1-alpha3"}
```

- **XML**

Request:

```
GET: http://platform35.demo.exoplatform.org/rest/api/social/version/latest.xml
```

Response:

```
<version>v1-alpha3</version>
```

3.6.4.2. version/supported

3.6.4.2.1. GET version/supported.format

Description: Gets eXo Social REST API versions that are supported. This is for backward compatibility. If a client application is using an older eXo Social REST APIs version, all APIs of the version still can work. The array **MUST** have the latest to oldest order. For example, [v2-alpha3, v1], but not [v1, v2-alpha3].

Url:

```
http://{domain_name}/{rest_context_name}/api/social/version/supported.format
```

For example:

- <http://platform35.demo.exoplatform.org/rest/api/social/version/supported.json>
- <http://localhost:8080/rest-socialdemo/api/social/version/supported.json>

Supported Format: json, xml

Requires Authentication: false

Parameters: No

Examples:

- JSON

Request:

```
GET: http://platform35.demo.exoplatform.org/rest/api/social/version/supported.json
```

Response:

```
{"versions": ["v1-alpha3"]}
```

- XML

Request:

```
GET: http://platform35.demo.exoplatform.org/rest/api/social/version/supported.xml
```

Response:

```
<versions>
  <version>v1-alpha3</version>
</versions>
```

3.7. Public Javascript APIs

All references for Opensocial Javascript APIs can be viewed [here](#).

3.8. Social JCR Structure

3.8.1. Overview

Like any other eXo's products, eXo Social fully complies the JCR standard to store data (identity, profile, activity, space and relationship). The eXo Social JCR structure is organized to conform to the data storage for the individual purpose of eXo Social. With this structure, it is easy for you to manage and access the data

properly.

See the eXo Social JCR structure in the chart below:

The root node of Social workspace is */production/* which consists of two child nodes named *soc:providers*, and *soc:spaces*.

3.8.2. soc:providers

The *soc:providers* node is used to store the provider entities. In eXo Social, there are two built-in provider entities, including *organization*, and *space*. Its type is *soc:providers* that has child nodes of the *soc:providerdefinition* type.

3.8.2.1. soc:<providername>

The *<providername>* parameter in the *soc:<provider_name>* node depends on the identity providers. eXo Social has two identity providers, including *OrganizationIdentityProvider*, and *SpaceIdentityProvider* that contain organization identities (users) and space identities respectively.

The *soc:<provider_name>* node of the *soc:providerdefinition* type has the *soc:<identityRemoteId>* child nodes that have the *soc:identitydefinition* type.

The *soc:identitydefinition* node type has the following properties:

Property Name	Required Type	Mutiple	Description
soc:providerId	String	false	The provider Id is considered as a namespace for the remote Id.
soc:isDeleted	Boolean	false	Show that if the provider Id is deleted or not via the provider.
soc:remoteId	String	false	The local Id from a provider Id.

And the *soc:identitydefinition* node type has the following child nodes:

Child Nodes	Default Primary Type	Description
soc:profile	soc:profiledefinition	Store the detailed information of an identity.
soc:relationship	soc:relationship	Store all the relationships of an identity that is in connection with other identities.
soc:receiver	soc:relationship	Store all the relationships which contain an identity invited to connect by other identities.
soc:sender	soc:relationship	Store all the relationships which

Child Nodes	Default Primary Type	Description
		contain an identity inviting other identities to connect with himself.
soc:ignored	soc:relationship	Store all the relationships which contain an identity ignored by other identities.
soc:ignore	soc:relationship	Store all the relationships which contain an identity ignoring other identities.
soc:activities	soc:activitylist	Store all activities in the activity stream of an identity.
soc:spacemember	soc:spaceslist	Store all spaces of which an identity is a member.
soc:spacependingmember	soc:spaceslist	Store all spaces which an identity is pending for validation to join.
soc:spaceinvitedmember	soc:spaceslist	Store all spaces which an identity is invited to join.
soc:spacemanagermember	soc:spaceslist	Store all spaces of which an identity is a manager.

The *soc:relationship* node type has the child nodes of the *soc:relationshipdefinition* type that has the following properties:

Property Name	Required Type	Multiple	Description
soc:to	Reference	false	The sender identity. It refers to the <i>soc:identity_node</i> type.
soc:status	String	false	The status of the relationship, including three values: PENDING, CONFIRMED, and IGNORED.
soc:reciprocal	Reference	false	Denotes if the relationship is one way or two ways. It refers to the <i>soc:relationshipdefinition_node</i> type.
soc:from	Reference	false	The receiver identity. It refers to the <i>soc:identity_node</i> type.
soc:createdTime	Long	false	The time when the relationship is created.

The *soc:spaceslist* node type contains the child nodes of the *soc:spaceref* type that has the following properties:

Property Name	Required Type	Multiple	Description
soc:target	Reference	false	Refer to a space entity.

The *soc:profiledefinition* node type contains the *soc:avatar* child node of the *nt:file* type and other child nodes of the *soc:profilexp* type.

The *soc:profiledefinition* node type has the following properties:

Property Name	Required Type	Multiple	Description
soc:parentId	String	false	The identity Id of the profile.
void-Url	undefined	true	The URL to access the profile of an identity.
void-email	undefined	true	The email of an identity in the profile.
void-firstName	undefined	true	The first name of an identity in his profile.
void-fullName	undefined	true	The full name of an identity in his profile.
void-lastName	undefined	true	The last name of an identity in his profile.
void-username	undefined	true	The username of an identity in his profile.

The *soc:profilexp* node type has the following properties:

Property Name	Required Type	Multiple	Description
soc:position	String	false	The job position of an identity at an organization.
soc:company	String	false	The company where an identity works.
soc:skills	String	false	The work skills of an identity.
soc:description	String	false	The description of an identity's position at an organization.
soc:endDate	String	false	The date when an identity stops working at an organization.

Property Name	Required Type	Multiple	Description
soc:startDate	String	false	The date when an identity starts working at an organization.

The *soc:activitylist* node type has the child nodes of the *soc:activityyear*. The *soc:activitylist* node type has the following properties:

Property Name	Required Type	Multiple	Description
soc:number	Long	false	The number of activities in the activities list. The default value is set to 0.

The *soc:activityyear* node type has the child nodes of the *soc:activitymonth* type which has the same properties as its parent node type (*soc:activitylist*).

The *soc:activitymonth* node type has the child nodes of the *soc:activityday* type that has the same properties as its parent node type (*soc:activitymonth*).

The *soc:activityday* node type has the child nodes of the *soc:activity* type that has the following properties:

Property Name	Required Type	Multiple	Description
soc:isComment	Boolean	false	Specify if an activity is a comment or not. The default value is false, meaning that it is a normal activity.
soc:likes	String	true	The list of identity Ids who like the activity.
soc:priority	Float	false	A float number between '0' and '1' which represents the relative priority level of an activity in relation to other activities from the same source.
soc:postedTime	Long	false	The number which specifies the time at which an activity took place in milliseconds since the epoch.
soc:title	String	false	The string which specifies the primary text of an activity.
soc:titleId	String	false	The title Id of an activity.

Property Name	Required Type	Multiple	Description
soc:body	String	false	The string which specifies the body template message Id in the gadget specification. The body is an optional extended version of an activity.
soc:bodyId	String	false	The body Id of an activity.
soc:type	String	false	The application Id which creates an activity.
soc:externalId	String	false	An optional string Id which is generated by the posting application.
soc:url	String	false	The URL to access an activity.
soc:appId	String	false	The application Id which creates an activity.
soc:identity	Reference	false	The identity whose activity stream contains an activity.
soc:posterIdentity	Reference	false	The identity of the user who creates an activity.

The *soc:activity* node type has the *soc:params* child node of the *soc:activityparam* type and other child nodes of the *soc:activity* type.

The *soc:activityparam* node type has the following property:

Property Name	Required Type	Multiple	Description
	String	false	A map of key-values.

3.8.3. soc:spaces

The *soc:spaces* node is used to store data of eXo Social spaces. Its node type is *soc:spaces* that includes child nodes of the *soc:spacedefinition* type.

The *soc:spacedefinition* type has the following properties:

Property Name	Required Type	Multiple	Description
soc:type	String	false	The type of space which is used to run in the Classic or WebOS mode.

Property Name	Required Type	Multiple	Description
soc:membersId	String	true	The list of users which are members of a space.
soc:pendingMembersId	String	true	The list of users who are pending for validation to join a space.
soc:managerMembersId	String	true	The list of users who are managers of a space.
soc:invitedMembersId	String	true	The list of users who are invited to join a space.
soc:visibility	String	false	The space visibility: public, private, and hidden.
soc:name	String	false	The space name.
soc:avatarLastUpdated	Long	false	The last time when the avatar is updated.
soc:displayName	String	false	The display name of a space.
soc:registration	String	false	The space registration status: open, validation, and close.
soc:description	String	false	The description of a space.
soc:priority	String	false	The space priority level that is used to sort spaces in the spaces list. It contains three values: 1, 2 and 3. The smaller value has the higher priority level.
soc:app	String	false	The list of applications with portlet Id, application name, and its state (installed, activated, deactivated).
soc:groupId	String	false	The group associated with the corresponding space.
soc:url	String	false	The link to access a space.