Kosmos Reference Manual

Komposite Open Source Monitoring Suite

For the Kosmos 0.2.x branch

Revision: Id

Midori Consulting (http://www.midori.hu)

Aron Gombas

Kosmos Reference Manual: Komposite Open Source Monitoring Suite: For the Kosmos 0.2.x branch: Revision: Id

Midori Consulting (http://www.midori.hu)
Aron Gombas

Table of Contents

List of Figures

Preface

Project motivation & history

Working as developer and later lead engineer on various Java and C++ projects in the recent years, I had to spend a serious amount of time reguarly checking various sources of information: build reports, source code metrics, PMD and CheckStyle reports, the source code itself, project pages of the dependencies, industry news in online mags and such. In other words, I had to filter and manage information coming from different places to get a global picture about the current state of the project. This process was extremely cumbersome and time-consuming, and distracted me from the more enjoyable part of engineering.

Other members of the team suffered from the same problem, however they wanted to have this global 24/7 picture from a slightly different aspect, based on their "roles" in the team. We clearly needed some flexible and easily personalizable solution.

And this is what portals are about, right? Aggregation and customization.

Later I came up with the idea of developing a suite of lightweight, highly customizable portlets backed by a central server mechanism, and then deploying those to a customization-enabled portlet container. This way beside having a "reference" community page, everyone could set-up his own personal portal page.

This was how *Kosmos* has born.

While working on the very first portlets, I contacted JBoss and had explained them my plans. They liked the idea, it was in sync with certain things what they wanted to do themselves, so they invited my project as one of the first projects hosted in their new-born JBoss Labs [http://labs.jboss.com] forge.

Figure 1. Kosmos logo

In the very beginning, the project ran under a different temporary codename, but when incubating it to JBoss Labs, I had to find out the final name. As the guys at JBoss were waiting for me, I had to come up with a fancy name in a couple of hours. I decided to use the word *kosmos*, because of several reasons:

- is there anything more beautyful than our universe?
- the cosmos flower is a real beauty with its simplicity
- "mikrokozmosz" is a wonderful musical piece by Bela Bartok, one of the greatest Hungarian composers
- gives the opportunity to play more with the notions related, like "kosmonaut"
- a name with the length of six characters perfectly fits Java package names, jar filenames and such

Acknowledgments

Huge thanks to my wife, Szilva, for not giving up the struggle for such a long time, and Damon Sicore and everyone else at JBoss Labs [http://labs.jboss.com] for their support. Special thanks to every Kosmos project contributor (you know, guys, who you are).

Contact

You can always find the latest information about the Kosmos project, published by the community, at the JBoss Labs project page: http://labs.jboss.com/projects/kosmos [http://labs.jboss.com/projects/kosmos]. Beside the downloads and documentation, we host also our Wiki and blog here, and this is our primary information resource. Come and visit us regularly.

You can reach me in email at the following address: <aron dot gombas at midori dot hu>.

Aron Gombas

Chapter 1. Overview

Vision

The first releases of Kosmos focuses on integrating a small set of de-facto standard open source tools like Subversion or CruiseControl. These releases will form the base for future development and aspire to reach a production-ready state as soon as possible.

In the long run, we have left the door open with an extensible architecture. We will consider supporting anything demanded. That includes both popular open source and commercial tools if they are accessible through some kind of public web-service, API, or at by page-scraping (in case of web interfaces).

Design goals

We aim to reach the following primary design goals:

1. Lightweight architecture implemented with POJOs.
2. Several independent and simplistic components rather than single monolithic one.
3. Easy and flexible deployment, and so no changes in the monitored resources.
4. Full transparency for the monitored resources and no extra burden on them.
5. Maximum vendor-indepence: no proprietary features of servlet containers, application servers, portlet containers or WebDAV servers.
6. Consistent and intuitive user interface for all portlets.
7. Visualization by using charts and graphics instead of plain textual information, wherever possible.

Implementation goals

We aim to reach the following primary implementation goals:

1. Java 1.5 language compatibility (some of the language features introduced in Java 1.5 are not compatible with Hessian).
2. Portlets that are perfectly JSR-168-compliant.
3. Portlets that are conform with the portal theme by using exclusively the standard JSR-168 CSS styles.

Dependencies

Kosmos is built on the top of:

- Apache Commons [http://jakarta.apache.org/commons] projects: various packages used as utility classes.

- Display tag [http://displaytag.sourceforge.net] library: used for rendering the tables.
- Hessian [http://www.caucho.com/hessian] binary web service protocol: used for implementing the web-services.
- JavaSVN [http://tmate.org/svn] Subversion client library: used for processing Subversion repositories.
- Jakarta Slide [http://jakarta.apache.org/slide]: its client library is used to access the WebDAV-based cache. Additionally, Slide is also our primary WebDAV server implementation.
- JFreeChart [http://www.jfree.org/jfreechart] library: used to generate the chart images.
- JTidy [http://jtidy.sourceforge.net]: used to transform the HTML documents to XML before further processing.
- JSTL [http://jakarta.apache.org/taglibs] tag library: used in the view tier.
- Log4j [http://logging.apache.org/log4j/docs] library: used for general-purpose logging.
- Saxon [http://saxon.sourceforge.net] XSLT and XQuery processor: used to analyze HTML documents.
- Spring Framework [http://www.springframework.org]: used as IoC container and AOP implementation.

Compatibility

Please see the detailed compatibility matrix on the project website.

Chapter 2. Components and features

Kosmos server

The “remote server” component acts like a traditional back-end: it collects, analyzes, stores and caches all the information. It then provides results for the front-end. You might ask, "Is it necessary to complicate the deployment with this additional component?" Having a single layer (portlets only, that access the monitored resources directly), it could be much simpler!

The primary reason is that certain operations performed by the system (like monitoring a remote Subversion repository) can be *very* expensive: traversing the repository content can easily take hours, depending on many factors like the repository complexity, server performance, or network bandwidth. By using a simple caching mechanism built into the server component, if several portlet instances are monitoring the same repository and they fire identical requests, only the very first will result in a new traversal! The other requests will receive the cached result until the first cache-miss (which can be caused also by a time-out, of course). This simple mechanism gives a huge performance boost and puts less burden on the “target box”, the Subversion server in this case.

Server component features:

- Implements the application logic.
- Provides Hessian-based web services for the portlets (view tier).
- Caches service calculation results transparently.
- Stores the generated static content (e.g. chart images) to pluggable storage mechanisms (to a WebDAV repository by default).

Kosmos portlets

The portlets implement the “view tier”, they are responsible for rendering the results computed by the server component. Portlet technology was chosen over “traditional” web-application techniques, because flexible customization was a high-priority project goal.

You can set up any portal page, which can contain any number of the portlets in various layouts. Also, you can mix Kosmos portlets with portlets coming from other projects or vendors, without any restriction. The deployment and configuration process is portlet container-dependent, thus it is out of the scope of this document. Please refer to the technical documentation of your particular container.

Using the portlet user interfaces should be straight-forward. There are a set of common features supported by all the portlets:

- You can minimize, maximize each portlet or get some help by clicking the icons in the titlebar (provided your portlet container supports this and your portal theme doesn't hide those controls).
- You can sort the items in the tables in ascending or descending order by clicking to the column headers.
- You can get detailed information related to a given attribute by clicking the *information* icons.
- You can zoom and unzoom chart images by clicking them.

CruiseControl Monitoring portlet

This portlet monitors the continuous integration build processes managed by CruiseControl, a very well-known continuous build framework. It helps you to track whether the builds of your projects break, without checking email, reports, or web reports.

Portlet features:

- It reports on the build labels, build results, timestamps, and unit test results. You can also get detailed information about the unit tests.
- The status icons denote failed builds or successful builds with failed unit tests.

Please visit http://cruisecontrol.sourceforge.net [http://cruisecontrol.sourceforge.net] if you want to learn more about CruiseControl.

JIRA Monitoring portlet

This portlet monitors projects hosted by JIRA, a popular issue tracking and project management application. It helps you by giving a quick overview about the state of several projects in a single place.

Please note that Kosmos provides two different service implementations to serve data for this portlet. You can choose between these two implementations simply by specifying different `service.url` parameters for the portlet:

1. `http://localhost:8080/kosmos-server/kosmos-services/jira-service`: the data will be served by a service that downloads JIRA webpages and analyzes their HTML content.
2. `http://localhost:8080/kosmos-server/kosmos-services/jirasoap-service`: the data will be served by a service that uses the JIRA SOAP remote API to download information directly from the JIRA web application. Please note that you have to enable the SOAP interface in the JIRA settings, in this case.

Portlet features:

- It reports on project details and issues organized by status, priority and assignee. Some of the statistics are available as graphical charts.
- The status icons denote projects with a significant number of open issues.
- You can jump onto the JIRA page of the given project by clicking the project name, or to the actual project webpage by clicking the URL in project details view.

Please visit http://www.atlassian.com/software/jira [http://www.atlassian.com/software/jira] for a short summary about JIRA.

SourceForge Monitoring portlet

This portlet monitors the file releases on SourceForge, the world's largest development and download repository of open source projects. The main goal of this portlet is to help you to keep your project dependencies and development toolbox up-to-date, without regularly checking the dependency project pages one-by-one.

Portlet features:

- It reports on the latest versions of the packages and their age.
- The status icons denote new releases (projects with fresh file releases) or inactive projects (projects with very old latest release).
- You can jump onto the download pages of the given project or that particular version on SourceForge by clicking the appropriate package name or the version label.

Please visit the main page of Sourceforge at http://www.sourceforge.net [http://www.sourceforge.net] if you haven't done it before.

Subversion Monitoring portlet

This portlet monitors repositories managed by Subversion, one of the most widely used version control systems. It helps you track the activity and complexity of several separate repositories very easily in a single portlet.

Portlet features:

- It can connect both to secure and public repositories.
- It reports on the latest touch, developer activity and repository statistics. Additionally, some of stats are visualized in charts if you click to the `information` icons.
- The status icons denote inactive repositories (repositories with very low activity or very old latest touch).

Please visit http://subversion.tigris.org [http://subversion.tigris.org] if you are interested in Subversion.

Good practices

Here are a couple of additional ideas which can make your life easier when configuring your particular Kosmos portal instance:

- Combining Kosmos portlets with other portlets is a good practice to maximize the information effectively aggregated on your portal page. Other than the standard Kosmos portlets, you should consider using: Forums portlet, RSS portlet, Blog portlet, Poll portlet, CMS portlet and Wiki portlet.

  PortletBridge [http://www.portletbridge.org] is another extremely useful open source component, don't miss it.

  All these (provided you use them in the right way) should improve the communication both inside your community and with the external world.

- All portlets support multiple monitored resources, and you can group them as you wish. For ex-

ample, you can have a separate portlet for monitoring Spring, and another for monitoring ACEGI. It might make sense to group those as related packages together within a single portlet, and keep all the Struts-related packages in another portlet, and Tomcat-related packages in a third one.

Chapter 3. Deployment and configuration

Requirements

You will need to deploy the server component and the portlets separately. Please note that however we listed recommended container implementations below, you can use any other product until that is compliant with the appropriate servlet or portlet specifications. You can learn more about the compatibity issues by studying the compatibility matrix maintained on the project website.

- We offer deployment scripts in form of Ant build scripts, which automate the deployment process. If you want to use these, you will need to install Ant. The default target of each script is `redeploy`, which deletes the old deployment (if there exists) and deploys a new clean one. Of course, you can still deploy manually if Ant is not available in your environment. In any case, it's a good idea to look through the deploy scripts.
- The server component requires standard servlet containers (or a single container) for proper running. The most trivial option is to use the Apache Tomcat [http://jakarta.apache.org/tomcat] container.
- The portlets needs a JSR-168-compliant portlet container. One possible open source choice is JBoss Portal [http://www.jboss.org/products/jbossportal] deployed into a JBoss Application server [http://www.jboss.org/products/jbossas] instance.
- For storing the dynamically generated content (e.g. chart images), you need a WebDAV [http://www.webdav.org/] server implementation. One option is to use Jakarta Slide [http://jakarta.apache.org/slide], but Subversion [http://subversion.tigris.org] provides a WebDAV interface, too.

Deployment models

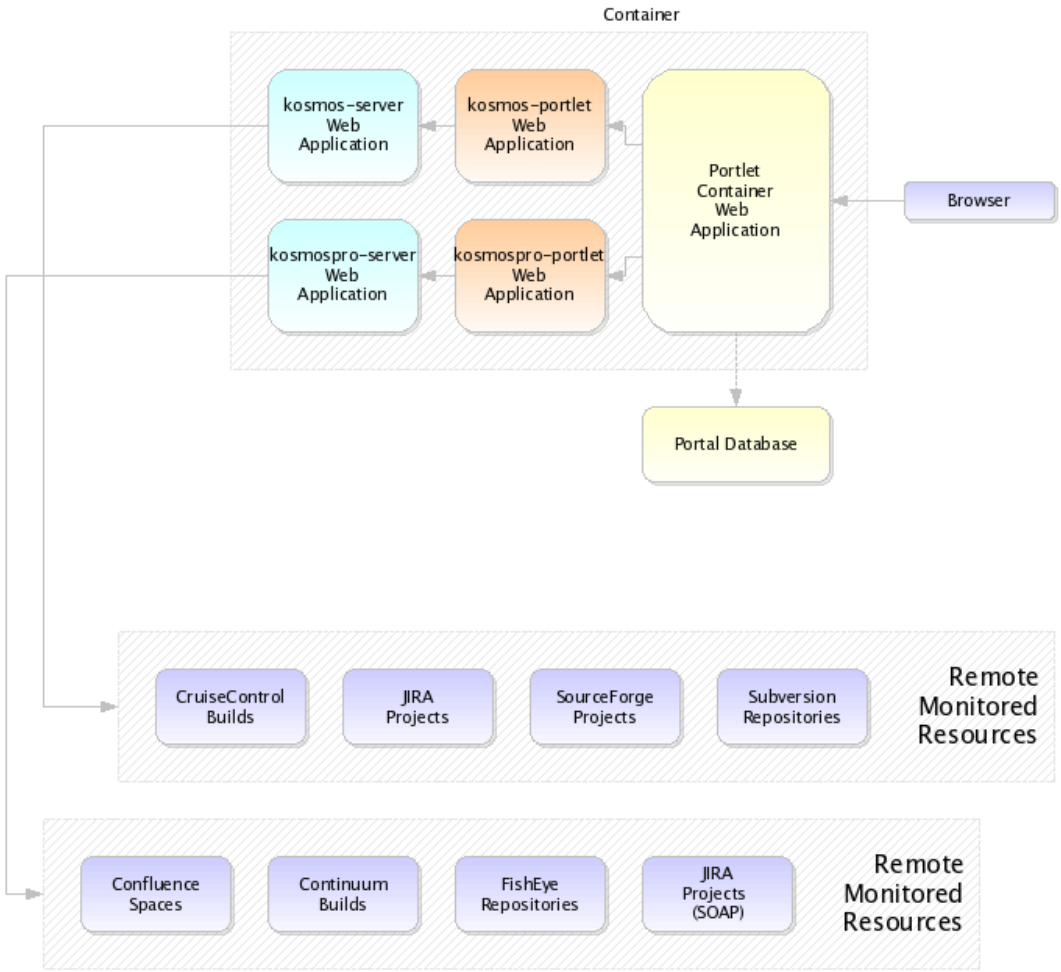
Due to the flexible architecture of Kosmos, it's possible to deploy the system to various environments, for example:

- *minimalistic*: a single container which can act both as servlet-container for Kosmos server, the WebDAV server, and the portlet-container for the portlets. As result, there will be three separate web applications running in the same container. This is the simplest way to deploy Kosmos and it can be an effective setup in many cases.
- *advanced*: separate (even heterogenous!) containers on the same physical node of the network: for example you can use Apache Tomcat as servlet-container and JBoss AS with JBoss Portal as portlet-container. It means that your components will run in separate JVMs which can be useful from a stability or security viewpoint.
- *distributed*: containers on separate nodes. *Server A* can run one instance of JBoss AS to host the server component, *server B* can run another to host the WebDAV repository, while *server C* can run a third one to host the portlet container.

  For advanced users it is possible to deploy each service of the server component to different nodes if that's necessary! You can fine-tune the performance of the system this way.

Since all the information is exposed as standard Hessian web-services, it is possible and perfectly legal to develop other types of front-end for the system: web applications, applets or desktop applications and completely avoid using a portlet container!

Figure 3.1. Minimalistic deployment



You can complicate all the models by another important decision: what RDBMS to use to support the portlet container and how to deploy it. Again, it's possible to use the same physical node or nodes which host the containers, or to have separate database servers for this purporse. It's all up to you, your needs and possibilities.

In the following sections, we give detailed step-by-step instructions for various deployment models, but because of their huge variety, we will cover just some of those. After reading these, it should be relatively easy to find out what to do in situations not listed here.

Deployment step-by-step for Apache Tomcat

and eXo Platform

1. Install eXo Platform [http://www.exoplatform.com] as written in its documentation [http://www.exoplatform.com/portal/faces/public/exo/home/community/wiki]. We recommend using the bundle distribution, because that contains both the Apache Tomcat servlet container and the eXo Platform portlet container in a single package.

2. Deploy Jakarta Slide [http://jakarta.apache.org/slide] to the Tomcat instance used by eXo as written in the JBoss AS step-by-step.

3. You can deploy the server component into the Apache Tomcat instance used by eXo, by running the server deploy script:

   ```
   ant -f deploy-server-tomcat.xml
   ```

   Please don't forget to set the `CATALINA_HOME` environment variable before.

4. Deploy the portlet web application by running the portlet deployment script as:

   ```
   ant -f deploy-portlet-exo.xml
   ```

5. Launch eXo, open the default portal page (e.g. `http://localhost:8080/portal`), and login with the default account (`admin` and `exo`).
   Go to page `edit` mode and add the Kosmos portlets to the page. Change back to `view` mode.

Deployment step-by-step for Apache Tomcat and Gridsphere

Follow these steps:

1. Install Gridsphere [http://www.gridsphere.org] as written in its manual [http://www.gridsphere.org/gridsphere/docs/UsersGuide/UsersGuide.html].

2. Deploy Jakarta Slide [http://jakarta.apache.org/slide] to the Tomcat instance used by Gridsphere as written in the JBoss AS step-by-step.

3. You can deploy the server component into the Apache Tomcat instance used by Gridsphere, by running the server deploy script:

   ```
   ant -f deploy-server-tomcat.xml
   ```

   Please don't forget to set the `CATALINA_HOME` environment variable before.

4. Deploy the portlet web application by running the portlet deploy script as:

   ```
   ant -f deploy-portlet-gridsphere.xml
   ```

   As an additional step, copy `$GRIDSPHERE_HOME/build/lib/gridsphere-ui-tags.jar` to `$CATALINA_HOME/webapps/kosmos-portlet.war/WEB-INF/lib`.

5. Launch Gridsphere, open the default portal page (e.g. `http://localhost:8080/gridsphere`), and login with the default account (`root` and empty

password). After this, there are couple of extra steps that you can do using the Gridsphere admin portlets:

a. Deploy `kosmos-portlet` manually.

b. Create a new public group `kosmos` that contains all the Kosmos portlets and add your user to this new group.

c. Go to your new page and add the Kosmos portlets to this.

Deployment step-by-step for JBoss AS and JBoss Portal

Follow these steps:

1. Install JBoss AS as written in its manual. The current reference documentation both for JBoss AS and JBoss Portal is available from the http://www.jboss.org site.

2. Deploy Jakarta Slide [http://jakarta.apache.org/slide] to JBoss AS and configure it as written in the appropriate manuals. We have included the Slide web application archive (`slide.war`) in the binary distribution package of Kosmos, under the `/etc` folder. In most situations, it's enough to copy this file to the deployment folder of the servlet container or the application server.

   As a quick test, check whether you can access the Slide repositories through a WebDAV navigator. If you use Windows XP, it is able to map a WebDAV repository as a folder to your filesystem. Otherwise try to open the repository in your browser using the `http://localhost:8080/slide` URL.

3. Install JBoss Portal as written in its manual. Test if there are no error messages in the JBoss AS logfile.

4. Run the server deploy script as:

   ```
   ant -f deploy-server-jboss-as.xml
   ```

5. Deploy the portlet web application by running the portlet deploy script as:

   ```
   ant -f deploy-portlet-jboss-portal.xml
   ```

6. Launch JBoss AS and check the default portal page (e.g. `http://localhost:8080/portal`), whether you can see the `Kosmos` page in the page list.

It is possible also to hot-deploy the components while JBoss AS is running.

Deployment step-by-step for Apache Tomcat and Liferay Portal

1. Install Liferay Portal [http://www.liferay.com/web/guest/products] as written in its manual. We recommend using the bundle distribution, because that contains both the Apache Tomcat servlet con-

tainer and the Liferay Portal portlet container in a single package.

2. Deploy Jakarta Slide [http://jakarta.apache.org/slide] to the Tomcat instance used by Liferay as written in the JBoss AS step-by-step.

3. You can deploy the server component into the Apache Tomcat instance used by Liferay, by running the server deploy script:

   ```
   ant -f deploy-server-tomcat.xml
   ```

   Don't forget to set the `CATALINA_HOME` environment variable before.

4. Hot-deploy the portlets by running the portlet deploy script:

   ```
   ant -f deploy-portlet-liferay-portal.xml
   ```

   Edit this file and make sure the properties are correct. If you have CATALINA_HOME set, you shouldn't have to worry about the

   ```
   container.dir
   ```

   property. Ensure

   ```
   container.lib
   ```

   and

   ```
   liferay.home
   ```

   are correct. Hot-deploy the portlets as written at http://www.liferay.com/web/guest/documentation/development/hot_deploy. The portlet distribution package of Kosmos contains a customized version of `portlet-deployer-3.5.0.xml` which you might find convenient as starting point.

   There are a couple of issues you can face while starting up your Liferay, depending on your JVM version:

   a. You have to downgrade the `$LIFERAY_HOME/webapps/kosmos-portlet/WEB-INF/lib/ext/commons-logging-1.0.4.jar` to `commons-logging-1.0.3.jar` (downloadable from http://www.ibiblio.org/maven/commons-logging/jars/) if Liferay throws a `java.lang.NoSuchMethodError: org.apache.log4j.Category.log(Ljava/lang/String;Lorg/apache/log4j/Level;Ljava/lang/Object;Ljava/lang/Throwable;)V`. The same exception might be thrown also for the kosmos-servlet and the kosmos-portlet web-applications, the fix is the same: downgrade the same JAR also in `$LIFERAY_HOME/webapps/kosmos-server/WEB-INF/lib` and `$LIFERAY_HOME/webapps/kosmos-portlet/WEB-INF/lib`, respectively.

   b. You have to delete `$LIFERAY_HOME/common/endorsed/xml-apis.jar` if Liferay throws a `javax.servlet.ServletException: Provider org.apache.xalan.processor.TransformerFactoryImpl not found` exception

   c. You have to copy `$LIFERAY_HOME/liferay/WEB-INF/tld/liferay-portlet.tld` to `$LIFERAY_HOME/webapps/kosmos-portlet/WEB-INF/tld` if Liferay throws a `org.apache.jasper.JasperException: /pages/sf_monitoring.jsp(1,1) File "/`

`WEB-INF/tld/liferay-portlet.tld" not found` or similar exception

d. You have to copy `$LIFERAY_HOME/liferay/WEB-INF/lib/util-taglib.jar` to `$LIFERAY_HOME/webapps/kosmos-portlet/WEB-INF/lib` if Liferay throws a `org.apache.jasper.JasperException: /pages/sf_monitoring.jsp(1,1) Failed to load or instantiate TagExtraInfo class: com.liferay.portlet.taglib.ActionURLTei` or similar exception

5. Launch Liferay, open the default portal page (e.g. `http://localhost:8080`) and login with the default account (`test@liferay.com` and `test`).

   Open the portlet administration portlet and assign some roles to the Kosmos portlets (e.g. `User` and `Guest`). Also, set the JIRA and Subversion monitoring portlets to `wide` style, all the others to `narrow`. Create a new page or go to an existing page, and check whether you can see the Kosmos portlets selectable in the combo boxes at the bottom of the portal pages.

Deployment step-by-step for Apache Tomcat and Apache Pluto

Follow these steps:

1. Install Pluto [http://portals.apache.org/pluto] as written in its manual. We recommend using the bundle distribution, because that contains both the Apache Tomcat servlet container and the Pluto portlet container in a single package.

2. Deploy Jakarta Slide [http://jakarta.apache.org/slide] to the Tomcat instance used by Pluto as written in the JBoss AS step-by-step.

3. You can deploy the server component into the Apache Tomcat instance used by Pluto, by running the server deploy script:

   ```
   ant -f deploy-server-tomcat.xml
   ```

   Please don't forget to set the `CATALINA_HOME` environment variable before.

4. Pluto has a portlet called `Deploy War Admin Portlet` to deploy other portlets. It makes the deployment process extremely easy: just select the portlet WAR and configure the pages, Pluto will take care of all the low-level details. Don't forget to restart Pluto after you've deployed your portlets, otherwise your new portal page won't appear in the menu!

   If you decided to do an automated deployment instead of using the administrative portlet, run the portlet deploy script:

   ```
   ant -f deploy-portlet-pluto.xml
   ```

   . After this, you have to manually update the following Pluto configuration files:

```
pageregistry.xml
portletcontexts.txt
portletentityregistry.xml
```

You can use the sample files found in `/conf/pluto` of the Kosmos portlet distribution package as starting point.

5. Launch Pluto and check the default portal page (e.g. `http://localhost:8080/pluto/portal`), whether you can see your new page in the page list.

General server configuration

The server component can be configured through its Spring application context configuration file `kosmos-services-servlet.xml`. (Please note that former versions were configurable partly through their `web.xml`, but after introducing the pluggable cache store mechanism, all configuration was moved to the Spring XML.) It's absolutely straightforward to modify it, but please note that the configuration changes might require reloading the servlet!

Here is some basic help for better understanding of the `kosmos-services-servlet.xml`:

- Each service is implemented by a POJO that is exposed as web service by a Spring-based Hessian proxy class. Caching is implemented through a very simple AOP interceptor around the service POJO instance. Consequently, there is one section like this per service:

```
<!-- CC service -->
<bean id="ccService" class="hu.midori.kosmos.server.cc.CcServiceImpl">
  <property name="store" ref="webdavStaticContentStore"/>
</bean>

<bean id="ccServiceProxy" class="org.springframework.aop.framework.ProxyFactoryB
  <property name="targetName" value="ccService"/>
  <property name="interceptorNames">
        <list>
          <value>serviceCachePointCutAdvisor</value>
        </list>
  </property>
</bean>

<bean name="/cc-service" class="org.springframework.remoting.caucho.HessianServi
  <property name="service" ref="ccServiceProxy"/>
  <property name="serviceInterface" value="hu.midori.kosmos.protocol.CcService"/
</bean>
```

  You can activate and deactivate the services by adding or deleting these sections, depending which portlets you're going to use. In the default configuration, all the services are activated and unless there is a special reason to remove them, it's better not to touch these sections. There is hardly any performance penalty or security problem even if you have unused, but active services.

  There is one required property for the services: this is called `store` and it's a reference to the cache store to use. See next section for more details.

- The services can generate and save cached data, mostly images that are later used by the portlets view tier. There is a simple, but flexible store mechanism built into Kosmos. However this is a "pluggable" mechanism, by default, there is only one implementation shipped with Kosmos: a WebDAV-based cache store. (In the very beginning, using WebDAV for this purpose was a requirement, not an option, but the poor quality of the WebDAV client libraries and servers motivated adding an extra level of abstraction to ensure that with some minimal work, any other web-based store can be used here.)

  The cache stores are implemented as POJOs, too. They can have different properties depending on the implementation, we discuss only `WebdavCachedDataStore` here:

```
<!-- WebDAV cached data store -->
```

```
<bean id="webdavCachedDataStore" class="hu.midori.kosmos.server.WebdavStaticCont
  <property name="webdavUrl" value="http://localhost:8080/slide/files"/><!-- Bot
  <property name="webdavUser" value=""/>
  <property name="webdavPassword" value=""/>
  <!-- This URL will be used as base URL for the generated images.
           If you don't specify anything here, the value of "webdavUrl"
           will be used. Uncomment this, if you want to override that.
       <property name="clientUrl" value="http://myserver/my-webdav/kosmos-image
  -->
</bean>
```

The URL is required, but you can leave the user and password empty if your WebDAV is configured to serve unauthenticated requests, too. In the URL you can use both HTTP and HTTPS protocols. The `clientUrl` parameter makes it possible to override the URLs generated for the clients: you can store the images as `https://secure.mydomain.com/mywebdav`, but access them as `http://public.mydomain.com/webdav` if your network environment is configured to match this.

- The trigger for the update scheduler can be configured as a standard Spring trigger bean, either a simple one or a cron-style one:

```
<bean id="serviceResultUpdateTrigger" class="org.springframework.scheduling.quar
  <property name="jobDetail" ref="serviceResultUpdaterJob"/>
  <property name="startDelay" value="7200000"/><!-- start and repeat in every 2
  <property name="repeatInterval" value="7200000"/>
</bean>
<!--
  Alternatively, a cron-style trigger can be used.
  For this, remove the previous bean definition and use this one:

  <bean id="serviceResultUpdateTrigger" class="org.springframework.scheduling.qu
       <property name="jobDetail" ref="serviceResultUpdaterJob"/>
       <property name="cronExpression" value="0 0 6 * * ?"/>
  </bean>
-->
```

All the information delivered by the services will be automatically refreshed in the given time. For example, you can schedule a very expensive Subversion repository traversal at 2 o'clock in the morning every day, so the portal will reflect up-to-date results by the time you get to your browser.

See the Spring Framework documentation: Wiring up jobs using triggers and the SchedulerFactory-Bean [http://static.springframework.org/spring/docs/2.0.x/reference/scheduling.html#d0e19434] for more examples.

- It's possible (and relatively easy) to do more complicated changes (like using separate or even in-homogenous cache stores per service, using more than one instance of the same service, etc.), but please make sure that you know what you do. It's recommended to study the related sections of the Spring Framework documentation [http://www.springframework.org/documentation], too.

Please take a look at the sample configuration files shipped in the distributed package.

General portlet configuration

All the other portlet settings can be configured by specifying `<init-param>` entities in the `portlet.xml` . The common init-parameters supported by every portlet are:

(required)	It is used only for display purposes, does not affect the actual functionality.
`service.url` (required)	It points to the appropriate Hessian-service. For example, in the case of `SfMonitoringPortlet` it can be: `ht-`

.

A quick check to test whether the service is available at the given URL is opening the URL in a normal browser window. You should see a “Hessian requires POST” error message if everything is fine.

`monitored.urls` (required)

Comma-separated list of items to monitor. Depending on the portlet, items are used as:

- `CcMonitoringPortlet`: URLs of the webpages where CruiseControl publishers produce their output including the logfiles related to the projects to monitor. For example: `http://cruisecontrol.jboss.com/cc/buildresults/ejb3-4.0-testsuite,http://cruisecontrol.jboss.com/cc/buildresults/ejb3-head-testsuite`.

- `JiraMonitoringPortlet`: URLs of the JIRA home pages related to the projects to monitor. For example: `http://jira.jboss.com/jira/browse/JBWIKI,http://jira.jboss.com/jira/browse/JBLAB`.

- `JiraSoapMonitoringPortlet`: list of colon-separated JIRA SOAP service URL and saved filter names related to the projects to monitor. Username and password is required for proper authentication. For example: `http://soaptester:soaptester@jira.atlassian.com/rpc/soap/jirasoapservice-v2?wsdl:Fixed for unreleased versions`.

- `SfMonitoringPortlet`: URLs of the SourceForge pages which host the projects to monitor. For example: `http://www.sourceforge.net/projects/springframework,http://sourceforge.net/projects/aceg isecurity`.

- `SvnMonitoringPortlet`: URLs of the Subversion repositories to monitor. For example: `http://svn.apache.org/repos/asf/commons,http://svn.apache.org/repos/asf/db`. If you have secure repositories, you must include the username and the password in the URL: `http://myusername:mypassword@www.mycompany.com/svn/mysecurerepo`. Please note that the security information will not appear in the user interface, so if you restrict the access to `portlet.xml` and the serverside log, then it's completely safe.

`fixed.view` (optional)

It is possible to restrict the portlets to a certain view and to disable the item listing and the navigation between sub-views. For example, you might want the portlet to display only the commit history of a single Subversion repository. If you don't specify anything for this parameter (this is the default), then the main view will be the item list and each sub-view will be accessible from there.

Depending on the portlet, you can use one of the following self-explanatory values:

- `CcMonitoringPortlet`: `viewTestDetails`.
- `JiraMonitoringPortlet`: `viewAssigneeDetails`, `viewIssueDetails` or `viewProjectDetails`.
- `JiraSoapMonitoringPortlet`: same like the previous one.
- `SfMonitoringPortlet`: does not support fixed views.
- `SvnMonitoringPortlet`: `viewActivityDetails`, `viewCommitterDetails`, `viewRepositoryDetails` or `viewRevisionDetails`.

Here is a section of the default `portlet.xml` file shipped in the distribution:

```
<portlet>
  <portlet-name>KosmosDependenciesSfMonitoringPortlet</portlet-name>
  <portlet-class>hu.midori.kosmos.portlet.sf.SfMonitoringPortlet</portlet-class>
  <supported-locale>en</supported-locale>
  <supported-locale>hu</supported-locale>
  <resource-bundle>hu.midori.kosmos.portlet.sf.sf_monitoring</resource-bundle>
  <init-param>
        <name>monitored.resource</name>
        <value>Kosmos Dependencies</value>
  </init-param>
  <init-param>
        <name>service.url</name>
        <value>http://localhost:8080/kosmos-server/kosmos-services/sf-service</val
  </init-param>
  <init-param>
        <name>monitored.urls</name>
        <value>
                  http://sourceforge.net/projects/cruisecontrol/,
                  http://sourceforge.net/projects/displaytag/,
                  http://sourceforge.net/projects/jfreechart/,
                  http://sourceforge.net/projects/jtidy/,
                  http://sourceforge.net/projects/saxon/,
                  http://www.sourceforge.net/projects/springframework
        </value>
  </init-param>
  <supports>
        <mime-type>text/html</mime-type>
        <portlet-mode>HELP</portlet-mode>
        <portlet-mode>VIEW</portlet-mode>
  </supports>
  <portlet-info>
        <title>SourceForge Monitoring</title>
  </portlet-info>
</portlet>
```

If you have any problems or questions, please study the default `portlet.xml` , and you will probably find the answer there.

I18n

Setting the preferred language for the whole application is a two-step process:

- For the server component, you have to set the context parameter called `locale` in its `web.xml`:

  ```
  <context-param>
    <param-name>locale</param-name>
    <param-value>en</param-value>
  </context-param>
  ```

  This setting will affect mostly the labels on the chart images, as these are the only resources generated by the server component that are language-dependent.

- For the portlets, you have to set the context parameter which configures the JSTL library in the portlet module `web.xml`:

  ```
  <context-param>
    <param-name>javax.servlet.jsp.jstl.fmt.locale</param-name>
    <param-value>en</param-value>
  </context-param>
  ```

  This setting will specify the language for the whole web-based user interface.

Both these parameters accept the standard Java locale signs as value, but please make sure that the property file of the selected language is available in your distribution. Please see the project site for localized versions available.

For advanced users: as you see, one server supports only one language set, which is absolutely sufficient in most of the cases. A multi-language server would require a more complicated caching mechanism (some parts of the content are language-independent thus could be shared between client requests with different locale settings, while others are not), which is necessary only in some infrequent environments. As workaround, for two languages you can launch two separate Kosmos servers with different language settings.

Chapter 4. Developer guide

Building from source

The Kosmos build system use a small set of easy-to-understand Ant scripts and property files. There are three actual build-scripts: one for building the server component (`build-server.xml`), one for building the portlets (`build-portlet.xml`) and one for creating the distribution packages (`build-distro.xml`). The two "real" build scripts use a common template `build/build.xml` and are configured through the property files in the `build` directory.

The variable names are self-explanatory and all the targets are well-documented in the appropriate script:

```
Buildfile: build-portlet.xml
Buildfile: build-portlet.xml
Kosmos Portlet Module build-script
Main targets:

 all       Recompiles all Java source files
 clean     Cleans up temporary files created during previous builds
 compile   Compiles Java source files
 deploy    Deploys the module to the container
 dist-bin  Prepares all binary distributables
 redeploy  Redeploys the module to the container
 undeploy  Undeploys the module from the container
Default target: redeploy
```

For instance, recompiling the full source code is simply:

```
ant -f build-portlet.xml all
```

Please study the scripts themselves for further details.

Generating the distribution packages

There are a couple of Ant targets for this purpose:

```
Buildfile: build-distro.xml
Kosmos Distro build-file
Main targets:

 clean               Cleans up temporary files created during previous builds
 dist                Prepares all distributables
 dist-bin            Prepares all binary distributables
 dist-src            Prepares all source distributables
 javadocs            Generates the javadocs
 manual              Generates the manual in all formats
 manual-html         Generates the manual in multi-page HTML format
 manual-pdf          Generates the manual in PDF format
 manual-single-html  Generates the manual in single-page HTML format
Default target: dist
```

The only detail you have to care about is setting the correct local paths in `build.properties` for the following dependencies that are not included in the Kosmos distribution package:

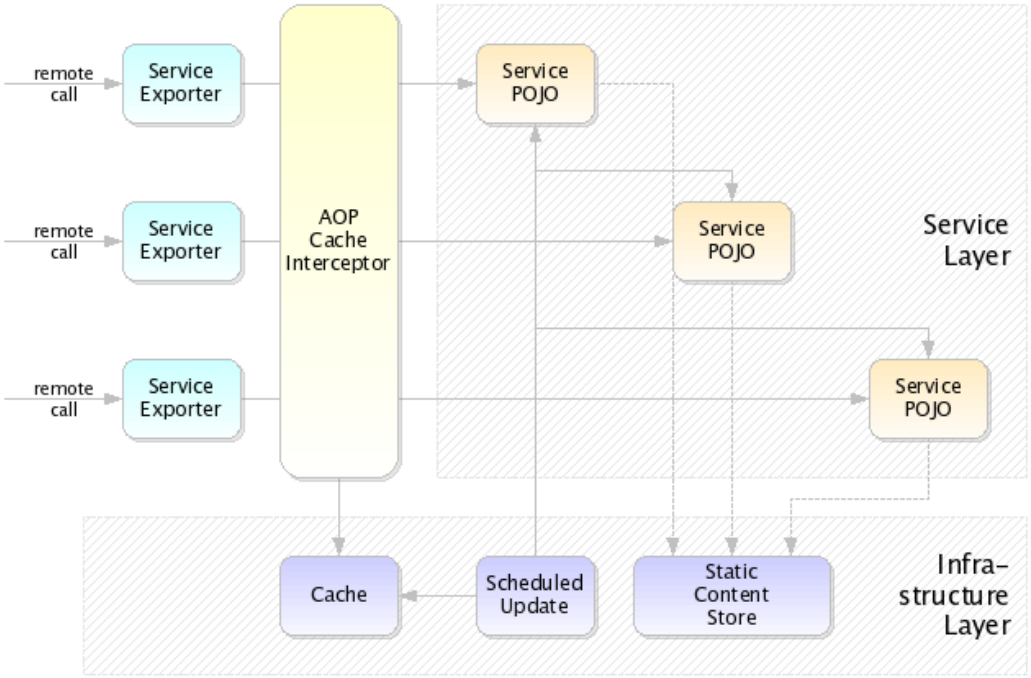
```
docbook.dir=/java/docbook-xsl-1.69.1
fop.dir=/java/fop-0.20.5
```

```
saxon.dir=/java/saxonb-8.5
```

Server component architecture

The Kosmos server component is a loosely coupled network of collaborating POJOs, wired together via the powerful Spring IoC container.

Figure 4.1. Kosmos server architecture



Obviously, the most important POJOs are the services themselves. As the monitored resources and their interfaces vary a lot, there is no uniform way for the concrete services to access them. Services use various techniques to get the information requested, ranging from simple web crawling to using remote APIs. The actual techniques are documented in the javadocs of the appropriate `XxxServiceImpl` implementation classes.

The common features of the services are factored out to the `AbstractKosmosService` class. This where the initialization and static content storage gets done. You should start learning the server component code by studying the javadocs of this class. Also, to understand the static store mechanism, please take a look at the `StaticContentStore` interface, and `WebdavStaticContentStore` as a sample implementation.

The service methods themselves are intercepted by a fairly simple AOP interceptor that stores service method results in an EHCache-based cache, keyed by the service method invocation arguments. If the same service method is invoked again with the same arguments, the interceptor tries to look up the return value in the cache first. It delegates to the service POJO only in case of cache miss. All this is completely transparent for the service POJOs.

There is another useful, infrastructure layer mechanism to achieve performance improvement: scheduled updates. Configuring the update trigger in `kosmos-services-servlet.xml`, you can schedule automatic service reloads at given periods or times. For example, you can schedule a very expensive Subversion repository traversal at 2 o'clock in the morning every day, so the portal will reflect up-to-date results by the time you get to your browser. Scheduling is done using the powerful Quartz library and gives you extreme flexibility.

From the server to the portlets, the data is transferred simply by instantiating the POJO classes in the `hu.midori.kosmos.model` package and sending them over the wire using Hessian.

Again, all this is very simple and lightweight.

Portlets architecture

The portlets are extremely simple: they just connect the appropriate service, download a collection of DTOs and render the JSPs. They do some basic processing of the portlet init parameters and the request parameters, but this is very easy to understand from the source code.

That's it.

Appendix A. Copyright

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of

any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square

root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these

materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this

License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE

LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS