$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

Getting Started with OpenEAI
Version 1.2

March, 2003

by

Tod Jackson (tod@openeai.org)
Steve Wheat (steve@openeai.org)

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

Contents

Introduction

This document provides an overview for anyone wanting to get their hands dirty with OpenEAI. It walks through the entire OpenEAI process including analysis, message design, application development, and application deployment. As a part of this guide, an entire sample enterprise, including all analysis artifacts, Java code and deployment descriptors are available for review and reference. This enterprise is based completely on open source software and artifacts.

For the latest version of this document, please check the Core Documentation Suite on the OpenEAI web site.

If you got this document from the OpenEAI web site and do not have the OpenEAI Examples distribution, you may want to download that distribution from the site as well, because this document focuses on things provided in that distribution.

Current State of this Document

This document and the sample enterprise will continue to evolve over time to provide information about getting started with OpenEAI to the various types of people who are involved in choosing integration technology and implementing integrations. The following is a history of modifications made to this document along with a statement about the intended audience that each released revision targets.

- Version 1.1 – 1/29/2003

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

This version just contained the Introduction section that described the vision of what this document will become.

- Version 1.2 – 3/4/2003

  This version adds the content describing the "OpenEAI Sample Enterprise" from a developer's perspective. It picks up where the analysis and message design will leave off. It DOES NOT necessarily follow all the OpenEAI recommended deployment patterns. That treatment will come in future, more advanced, versions of this document. However, it does provide several messaging components and a simple deployment strategy intended to get people up and running quickly with the components included. An overview of those components can be found in the Overview of the Sample Enterprise section of this document.

Overview of the Sample Enterprise

Welcome to the first version of the OpenEAI Sample enterprise! In this version of the Getting Started guide, we'll walk you through a simple deployment of the components currently contained within the sample enterprise. Running these applications and gateways and reviewing their source code and Javadoc should provide you with a good opportunity to learn more about OpenEAI and see how some of the pieces fit together.

This version of the document is generally targeted toward developers or those who wish to see OpenEAI in action. In the future, this document will evolve to include all parts that lead up to this document (analysis, message design, implementation design, etc.) as well as a more thorough treatment of OpenEAI recommended deployment patterns. This is an evolutionary process, and the sample enterprise as well as this document will grow over time. Hopefully, you find it useful and are able to see how all of this fits together to form an integrated enterprise.

Our sample enterprise is based on applications from two fictitious organizations. One organization is an enterprise that has committed itself to building OpenEAI based applications and gateways, the "Any OpenEAI Enterprise." The other organization is an ERP vendor, the "AnyERP Vendor", who provides an ERP system and an OpenEAI-based gateway as one method of integrating with the AnyERP system. To complete the fiction, the Any OpenEAI Enterprise has purchased the AnyERP system from the AnyERP Vendor and is integrating its existing and new systems with the new AnyERP system using the OpenEAI methodology and technology.

Like Java in general and other internet technologies, OpenEAI uses naming and packaging conventions that are based on internet domain names. The internet domain for the Any OpenEAI Enterprise is any-openeai-enterprise.org and the internet domain for the AnyERP Vendor is any-erp-vendor.com.

Below is a brief description over the components deployed. A graphical representation of this enterprise is also included in Appendix 1 of this document and available in electronic format in the distribution and on the web site. A version of this document can be found in the `configs/messaging/Environments/Examples/Documentation` directory included in the archive. Additional details will be mentioned later in this document as we talk about actually starting these components. Javadoc for these components is also included with the distribution and can be re-generated when you compile the supplied sample code.

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

Any ERP Vendor

The AnyERP system is an ERP system that is authoritative for the following enterprise message objects:

- com.any-erp-vendor/Person/BasicPerson
- com.any-erp-vendor/Person/EmergencyContact
- com.any-erp-vendor/Employee/BasicEmployee

It exposes an API for these message objects through the AnyERP Vendor Message Object API (MOA) and corresponding Enterprise Object (EO) documents that have been generated by the OpenEAI MOA Generation Application. By using this MOA along with other OpenEAI foundation components like PointToPointProducers, any enterprise that wishes to integrate with a deployment of the AnyERP system can do so following the OpenEAI patterns and techniques.

The AnyERP Vendor also provides an OpenEAI-based message gateway for their AnyERP system, which consumes request messages sent on behalf of the above objects and performs the action specified in the request in its repository. These actions include: Query, Create, Update and Delete. When a Create, Update or Delete action is requested, this gateway also informs the rest of the enterprise of that action by publishing the corresponding synchronization message. This allows other non-authoritative systems to consume these messages and maintain any data repository they may have that they wish to keep up-to-date with the authoritative system.

Any OpenEAI Enterprise

The Any OpenEAI Enterprise has purchased the AnyERP and wishes to integrate with that system using the supplied MOA and OpenEAI foundation components. They have one gateway and one application that they wish to involve in this integration. Additionally, they've decided to use the OpenEAI *EnterpriseRequestProxy*, *EnterpriseTransRouter* and *EnterpriseLoggingService* reference implementations instead of developing their own infrastructure gateways to perform the tasks of implementing request authorization, synchronization message routing, and message logging. Please refer to the OpenEAI Implementation Strategies document for more information on these reference implementations.

The application that they've developed is a fairly simple Java Swing application. It is a self-service application that will allow their Human Resources department to maintain person information for people within their organization. Since they don't want this application to maintain its own data store, it will be integrated with the ERP system using the supplied MOA and OpenEAI foundation components.

The gateway they need consumes synchronization messages from the ERP vendor when changes occur to the com.any-erp-vendor/Person/BasicPerson enterprise message object. They'll consume those synchronization messages and keep a warehouse database up-to-date with the ERP system. Hypothetically, the data warehouse is used for decision support purposes in the Any OpenEAI Enterprise.

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

Installation of the Sample Enterprise

This section walks you through each step of installing the sample enterprise. Please note, there are several long commands listed in the examples that cause line breaks which can make it a little hard to read. In these cases, we suggest you copy the example from this document into an editor or tool that allows you to see the entire line without breaks, this will hopefully make it easier to read and understand what's going on in the example.

All of the examples in this section assume you'll be running things from the command line. On Windows, this means running them from a "Command Prompt" and on UNIX, it means running them from a terminal window.

Prerequisites

In order to run the supplied `setupOpenJMS`, `setupExamples`, `startOpenJMS`, `start` and `build` scripts, you must establish the following environment variables:

- `PATH` – should include the location to the MySQL executables (e.g. c:\mysql\bin or /usr/local/mysql/bin). Note: if you're planning on using Oracle, SQLPlus should be in your `PATH`.

  If you're using Oracle as your DBMs, you'll need to have the `ORACLE_HOME` environment variable specified in order for SQLPlus to work.

- `JAVA_HOME` – should be set to match the top level directory containing the JVM you want to use. For example:

  ```
  C:\> set JAVA_HOME=C:\jdk1.4.1
  ```

  on Unix:

  ```
  % setenv JAVA_HOME /usr/local/java
  ```
  (csh)
  ```
  > JAVA_HOME=/usr/java; export JAVA_HOME
  ```
  (ksh, bash)

- `OPENEAI_HOME` – should be set to match the directory where you unpacked the OpenEAI Examples archive. For example:

  ```
  C:\> set OPENEAI_HOME=C:\openeai-examples-1.0
  ```

  or on Unix:

  ```
  % setenv OPENEAI_HOME /opt/openeai-examples-1.0
  ```
  (csh)
  ```
  > OPENEAI_HOME=/opt/openeai-examples-1.0; export OPENEAI_HOME
  ```
  (ksh, bash)

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

It is recommended that you make these "system wide" environment variables so each window you open to execute the supplied scripts inherits these variables. This way, you don't have to reset them each time you wish to run a script.

On Windows, you can use the control panel to do this. On UNIX, you can either have your system administrator set these variables or you can set it for yourself by modifying your `.profile` file.

Un-packing the Archive

The archive that you downloaded from the OpenEAI website contains everything you need to run the sample enterprise with the exception of your database of choice (MySQL has been tested so far). This includes:

- A deployment descriptor that includes configuration information for all messaging components contained in the sample enterprise at this point
- Enterprise Object (EO) documents that were originally generated by the OpenEAI MOA Gen application, but have been manually altered to include a few more complex field-level business rules to validate data as its being placed into enterprise objects
- OpenJMS to serve as the message broker (which can be changed if desired)
- Broker configuration files for OpenJMS that contains all administered objects required by the components in the sample enterprise
- Database scripts to create the appropriate structures required by the components deployed in the sample enterprise
- Source code for the components
- TestSuite documents for running the OpenEAI TestSuite Application
- Message tree which contains all enterprise message definitions (DTDs) and all sample XML files provided by Any-ERP-Vendor.

The content of the distribution is described in more detail in the CONTENTS.TXT file included with the distribution.

Installing a Database Management System

We have tested the gateways that require database connections against the MySQL database. The deployment descriptor for the gateways in the sample enterprise are configured by default to go against a MySQL database. Since database installation procedures vary between products, we are not including detailed instructions regarding the installation and general configuration of a database. As the sample enterprise evolves, we will offer additional database scripts for other databases.

MySQL:

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

- Download MySQL from http://www.mysql.com/downloads/mysql-4.0.html
- Follow installation procedures appropriate to your OS.

Oracle:

We have included database scripts for Oracle. However, we have not fully tested the sample enterprise using the Oracle structures. If you'd like to use Oracle, and either have an existing Oracle installation or can get a version installed, you must change the "`dbConnectString`" and "`dbDriverName`" JDBC attributes in the sample deployment descriptor to use the appropriate connect string and driver. This deployment descriptor is located in:

```
OPENEAI_HOME/configs/messaging/Environments/Examples/Deployments/AnyOpenEaiEnterprise.xml.
```

Additionally, if you'd like to use Oracle as the OpenJMS repository, you must modify the "`RdbmsDatabaseConfiguration`" element in the OpenJMS deployment descriptor to use the appropriate driver and connection URL. This file is located in:

```
OPENEAI_HOME/configs/messaging/Environments/Examples/Brokers/OpenJMS/Broker1/AnyOpenEAIEnterprise.xml
```

More information regarding the deployment descriptor changes necessary is given below when we discuss configuring the databases. Oracle JDBC drivers have been included with the sample distribution.

Configuring the OpenJMS database

OpenJMS is a good, open-source JMS provider. We include it with our distribution because it is simple to configure, it seems to work well for what we're trying to do here, and it's freely available. One of the fundamental tenants of OpenEAI is that the JMS provider you choose doesn't really matter. All we're trying to do in the sample enterprise distribution is supply everything needed to run this sample enterprise, so you can see how all the pieces fit together and watch them work. We've included a precompiled version of OpenJMS in this distribution mostly for your convenience. You could also simply download and install it from scratch. OpenJMS can be downloaded from www.exolabs.org.

The OpenJMS broker configuration file that we've provided for our sample enterprise is located at:

```
OPENEAI_HOME/configs/messaging/Envioronments/Examples/Brokers/OpenJMS/Broker1/AnyOpenEAIEnterprise.xml
```

This file is used to create the OpenJMS broker database structures and contains information regarding the JMS administered objects that will be used by the JMS components in our sample enterprise. The path to this file is passed as a parameter to the OpenJMS scripts when building its database and when starting the OpenJMS server.

By default, this file is configured to use a MySQL database running on the `localhost` as the broker's repository. If you wish to use a different database as the broker's repository, you need to change this file to point to the appropriate database. Additionally, you may

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

need to add the appropriate JDBC libraries to the CLASSPATH in the `dbtool` and `startjms` scripts that are included in the `openjms-0.7.3.1\bin` directory.

As with most JMS providers, there are a range of options when installing and configuring OpenJMS. The configuration we've provided is solely intended for this sample enterprise and should not be considered a "recommended" way to configure and deploy OpenJMS, or any other JMS provider. Once you've become familiar with the components included in this sample enterprise, you are encouraged to modify the configuration of your broker as well as the components that use it.

Configuring the OpenJMS database for MySQL:

To create the OpenJMS database, grant the appropriate user permissions and create the OpenJMS tables required by the broker, execute the `setupOpenJMS` script that is provided in the `OPENEAI_HOME` directory. For example:

Windows:

```
C:\>%OPENEAI_HOME%\setupOpenJMS.bat mysql
```

Unix:

```
host:/ $OPENEAI_HOME/setupOpenJMS.sh mysql
```

The following steps are the manual steps required to configure the OpenJMS database. They are provided as information so you can get an idea about what's going on. The supplied `setupOpenJMS` script will perform all of these tasks for you so you should NOT need to manually run these instructions. If you're not interested in learning about these manual steps, you can skip to the next section.

Start the "MySQL monitor":

Windows example: **`C:\>`**`mysql --user=root mysql`
Unix example: **`host:/ $`**`mysql --user=root mysql`

Create the database and grant appropriate permissions:

```
mysql> source
OPENEAI_HOME/configs/messaging/Environments/Examples/Databases/openjms/cr
eate_mysql.sql;
```

Create OpenJMS db structures:

Windows example:

```
C:\>cd\OPENEAI_HOME\openjms-0.7.3.1\bin
C:\OPENEAI_HOME\openjms-0.7.3.1\bin\>dbtool.bat -create -config
OPENEAI_HOME\configs\messaging\Environments\Examples\Brokers\OpenJMS\Brok
er1\AnyOpenEAIEnterprise.xml
```

UNIX example:

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

```
host:/ $cd/OPENEAI_HOME/openjms-0.7.3.1/bin
host:/OPENEAI_HOME/openjms-0.7.3.1/bin/ $dbtool.sh -create -config
OPENEAI_HOME/configs/messaging/Environments/Examples/Brokers/OpenJMS/Brok
er1/AnyOpenEAIEnterprise.xml
```

Configuring the OpenJMS database for Oracle:

To create the OpenJMS schema, grant the appropriate user permissions and create the OpenJMS tables required by the broker, execute the setupOpenJMS script that is provided in the `OPENEAI_HOME` directory. **Note: before you can do this you MUST change the "RdbmsDatabaseConfiguration" element in the broker deployment descriptor to use Oracle's JDBC driver.** For example:

```
<RdbmsDatabaseConfiguration
      driver="oracle.jdbc.driver.OracleDriver"
      url="jdbc:oracle:thin:@localhost:1521:OPENJMS"
      user="openjms"
      password="openjms22"/>
```

Again, this file is located in:

`OPENEAI_HOME/configs/messaging/Environments/Examples/Brokers/OpenJMS/Broker1/AnyOpenEAIEnterprise.xml`.

Windows:

```
C:\>%OPENEAI_HOME%\setupOpenJMS.bat oracle user/password@SERVICE
```

Unix:

```
host:/ $OPENEAI_HOME/setupOpenJMS.sh oracle user/password@SERVICE
```

Where "`user/password@SERVICE`" is the fully qualified login information that tells SQLPlus which database you'll be logging into and as what user. This user must have dba privileges.

If you want to run the SQL script manually, you can do that also by executing the OpenJMS `create_oracle.sql` script from SQLPlus just like you could for MySQL. Then, you could execute the OpenJMS `dbtool` script from the command line. However, that is not required if you use the `setupOpenJMS` script provided for you. In order to execute the scripts, you must log into Oracle as a user with DBA privileges. If you've installed Oracle yourself, you should know the credentials necessary for this user. If not, you may need to ask your resident DBA to execute the scripts for you.

Configuring the OpenEAI Sample Enterprise Databases

There are currently four gateways in the sample enterprise that require databases. They are identified in the AnyOpenEaiEnterprise.xml deployment descriptor as:

`com.any-erp-vendor.Gateway` - our ERP vendor's gateway which stores an XML string representation of the enterprise message objects in its repository. The ERP repository is found in the `anyerp.T_ERP_DATA` table.

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

`org.any-openeai-enterprise.EnterpriseWarehouse` - the EnterpriseWarehouse gateway consumes synchronization messages from the ERP gateway and keeps the "warehouse" up-to-date with BasicPerson changes. The warehouse repository is found in the following tables:

- `ewh.T_PERSON`
- `ewh.T_ADDRESS`
- `ewh.T_PHONE`
- `ewh.T_EMAIL`

Like all sync consuming gateways, it also uses the `T_PROCESSED_MESSAGE` table to balance between multiple instances of this gateway.

`org.any-openeai-enterprise.EnterpriseTransRouter` - the OpenEAI reference implementation that routes synchronization messages from the ERP gateway to appropriate targets. One of those targets is the EnterpriseWarehouse gateway mentioned above. The only persistent store required by this gateway is the `etr.T_PROCESSED_MESSAGE` table that it uses to balance between multiple instances of the gateway.

`org.any-openeai-enterprise.EnterpriseLoggingService` – The OpenEAI reference implementation that logs any sync message delivered to it (for historical purposes) as well as logs any org.openeai.CoreMessaging/Sync/Error-Sync message that may be published by sync consuming gateways (like the EnterpriseWarehouse) when errors occur processing the sync messages delivered to that gateway. The repository for the EnterpriseLoggingService is found in the following tables:

- `els.T_SYNC_ERROR`
- `els.T_SYNC_ERROR_MSG`
- `els.T_LOGGED_MESSAGE`
- `els.T_PROCESSED_MESSAGE` (to balance)

The following set of instructions create the appropriate database structures required to fully operate these gateways.

If Running the MySQL database:

To create the sample enterprise databases, grant the appropriate user permissions and create the tables required by each gateway, execute the `setupExamples` script that is provided in the `OPENEAI_HOME` directory. For example:

Windows:

```
C:\>%OPENEAI_HOME%\setupExamples.bat mysql
```

UNIX:

```
host:/ $OPENEAI_HOME/setupExamples.sh mysql
```

The following steps are the manual steps required to configure the sample enterprise databases. They are provided so you can get an idea about what's going on. The supplied `setupExamples` script will perform all of these tasks for you so you should NOT need to manually run these instructions. If you're not interested in learning about these manual steps, you can skip to the next section.

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

Start the "MySQL monitor":

Windows example: `C:\>mysql --user=root mysql`
Unix example: `host:/ $mysql --user=root mysql`

Create the databases and tables required by the sample enterprise:

```
mysql> source
/OPENEAI_HOME/configs/messaging/Environments/Examples/Databases/anyerp/cr
eate_mysql.sql;
mysql> source
/OPENEAI_HOME/configs/messaging/Environments/Examples/Databases/els/creat
e_mysql.sql;
mysql> source
/OPENEAI_HOME/configs/messaging/Environments/Examples/Databases/etr/creat
e_mysql.sql;
mysql> source
/OPENEAI_HOME/configs/messaging/Environments/Examples/Databases/ewh/creat
e_mysql.sql;
```

If Running the Oracle database:

To create the sample enterprise schemas, grant the appropriate user permissions and create the tables required by each gateway, execute the `setupExamples` script that is provided in the `OPENEAI_HOME` directory.

Note: if you use Oracle, you MUST change the database connections listed in the sample enterprise OpenEAI deployment descriptor to use Oracle's JDBC driver. To accomplish this, you must change all "**dbConnectString**" and "**dbDriverName**" JDBC attributes appropriately. For example:

```
<DbConnectionPoolConfig
       name="AevRequestCommandDbPool"
       dbDriverName="oracle.jdbc.driver.OracleDriver"
       dbConnectString="jdbc:oracle:thin:@localhost:1521:ANYERP"
       dbConnectUserId="anyerp"
       dbConnectPassword="anyerp22"
       dbPoolSize="2"/>
```

It is generally easiest to just do a "search and replace" on all `dbDriverName` and `dbConnectString` attributes in the deployment descriptor.

Again, this file is located in:

`OPENEAI_HOME/configs/messaging/Environments/Examples/Deployments/AnyOpenEaiEnterprise.xml`.

Also, since Oracle doesn't support "REPLACE IF EXISTS…" type processing for tables, you may see some errors reported the first time you run the setup scripts because the first thing they attempt to do is DROP tables that won't exist if it's the first time you've run them. These errors can be ignored and should be fairly obvious.

Windows:

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

```
C:\>%OPENEAI_HOME%\setupExamples.bat oracle user/password@SERVICE
```

UNIX:

```
host:/ $OPENEAI_HOME/setupExamples.sh oracle user/password@SERVICE
```

Where "`user/password@SERVICE`" is the fully qualified login information that tells SQLPlus which database you'll be logging into and as what user. This user must have dba privileges.

Just like the OpenJMS database setup for Oracle, you can execute the provided `create_oracle.sql` scripts for the sample enterprise components from SQLPlus if you'd like. However, it is not necessary since the `setupExamples` script executes these SQL scripts for you. In order to execute the scripts, you must log into Oracle as a user with DBA privileges. If you've installed Oracle yourself, you should know the credentials necessary for this user. If not, you may need to ask your resident DBA to execute the scripts for you.

Starting OpenJMS

Before you can start any of the sample applications or gateways included in the distribution, you must start the broker. This will make the JMS administered objects and destinations managed by the broker available to those components that need them when you go to start them. As mentioned before, we've included a precompiled version of OpenJMS with our sample enterprise for convenience sake. If you wish to install OpenJMS yourself, you may do so with little trouble. You just need to add the appropriate libraries to the CLASSPATH built in the `dbtool` and `startjms` scripts.

If you would like to use the pre-compiled version of OpenJMS included in this distribution, you can follow these directions to start OpenJMS.

Windows:

```
C:\>%OPENEAI_HOME%\startOpenJMS.bat
```

UNIX:

```
host:/ $OPENEAI_HOME/startOpenJMS.sh
```

The following steps are the manual steps required to start the OpenJMS broker. They are provided so you can get an idea about what's going on. The supplied `startOpenJMS` script will perform all of these tasks for you so you should NOT need to manually run these instructions. If you're not interested in learning about these manual steps, you can skip to the next section.

Windows:

C:\>cd*OPENEAI_HOME*\openjms-0.7.3.1\bin

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

```
C:\OPENEAI_HOME\openjms-0.7.3.1\bin\>startjms.bat -config
OPENEAI_HOME\configs\messaging\Environments\Examples\Brokers\OpenJMS\Brok
er1\AnyOpenEAIEnterprise.xml
```

UNIX:

```
host:/ $cd/OPENEAI_HOME/openjms-0.7.3.1/bin
host:/OPENEAI_HOME/openjms-0.7.3.1/bin/ $startjms.sh -config
OPENEAI_HOME/configs/messaging/Environments/Examples/Brokers/OpenJMS/Brok
er1/AnyOpenEAIEnterprise.xml
```

Starting the OpenEAI Sample Enterprise Messaging Components

Now that you've got all the pieces in place, you are ready to start things up and play. If you'd like, you can skip down and build the provided source code or generate MOAs. If you've already done that or you just want to get right to it, this is the place. This section assumes you've installed a database and started OpenJMS. By default, all database connections are configured to use MySQL. If you'd like to use a different database, the database configurations listed in the deployment descriptor mentioned below (AnyOpenEaiEnterprise.xml) will have to be changed appropriately.

Additionally, all JMS and database connections are configured by default to use resources on the "localhost". If you're using a database or JMS provider on a different host, you would need to change configuration information for these resources to point to that host.

This version of the sample enterprise uses one deployment descriptor, the `OPENEAI_HOME/configs/messaging/Environments/Examples/Deployments/AnyOpenEaiEnterprise.xml` document. This deployment descriptor includes the configuration information for all components included in the sample enterprise. Each one of these components could be broken into their own deployment descriptor if desired, but we included them in one file here to maintain simplicity in the sample enterprise deployment. It is recommended that you take a look at this file and familiarize yourself with its contents. It contains configuration information for all applications and gateways that will be started in the coming sections. This includes:

- Configuring JMS foundation objects like PubSubConsumers, PointToPointConsumers, PubSubProducers and PointToPointProducers.
- Configuring database connection pools that are used by some of the gateway commands to persist messages etc.
- Configuring ThreadPools that are used by gateways and scheduled apps to execute Commands.
- Configuring Loggers used by all the applications and gateways.
- Configuring general properties associated to the commands executed by the applications and gateways.

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

In some cases, it may be necessary to modify this file to change the default expected locations of certain resources etc. You would also modify this file if you ever decide to try a different JMS provider or if you use different databases for persistence.

This set of instructions will help you start each messaging component included with the sample enterprise. It assumes you've unpacked the archive without modifying the structure. There should be nothing that needs to be changed to start these gateways and applications if you simply unpacked the archive. While all components should be started before you actually start trying to use the self service application, the order in which you start the components doesn't matter.

By running the components in the sample enterprise and watching their logs you should get a good idea of what's going on. However, to really see what's going on under the covers, you are encouraged to review the source code. It has been documented fairly thoroughly with Javadoc as well as inline comments so hopefully you'll get a real good picture as to what's going on within each one of the sample applications and gateways.

In this version of the Getting Started guide, we are providing a very simple way to start these components using Ant from the Apache Jakarta Project. While this does not follow the OpenEAI recommended patterns for starting and managing a production runtime environment completely, it does provide a simple mechanism to get you up and running quickly. The build.xml file that's included in the distribution has targets that can be used to start every component in the sample enterprise. By specifying the appropriate Ant target, you will start the corresponding messaging component. In order to start a gateway or application, you run the `start` script passing the appropriate target as a parameter. This will instruct Ant to start the corresponding component using the properties file associated to that component which is located in the `props` directory off of the distribution root directory. This will instruct Ant to run the `org.openeai.jms.MessageConsumerClient` or `org.openeai.afa.GenericAppRunner` class and use the properties file associated to the component as input to those classes. Since the property files contain the appropriate `providerUrl` to the deployment descriptor used for our sample enterprise it will look for the `messageComponentName` specified in the current properties file and start the gateway that has an `id` matching that name in the deployment descriptor.

Now, we'll walk through starting each component included in the sample enterprise. All gateways and applications included in the sample enterprise are started by opening a command prompt or terminal window, executing the `OPENEAI_HOME/start` script and passing the appropriate target to the script. For example:

Windows:

```
C:\>%OPENEAI_HOME%\start.bat EnterpriseRequestProxy
```

UNIX:

```
host:/ $OPENEAI_HOME/start.sh EnterpriseRequestProxy
```

This will tell Ant to locate the EnterpriseRequestProxy target in the build.xml file and start the `MessageConsumerClient` class passing the `./props/EnterpriseRequestProxy.properties` file to it. Use this same procedure to start each gateway and application provided in this version of the sample enterprise. The

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

only thing that changes is the name of the component you're starting. The following instructions walk you through starting all messaging components included in the sample enterprise and gives you a brief overview of each of them.

EnterpriseRequestProxy

To start the OpenEAI EnterpriseRequestProxy reference implementation, open a command prompt or terminal window and execute the `OPENEAI_HOME/start` script passing `EnterpriseRequestProxy` as an argument.

This gateway is a reference implementation provided by OpenEAI that ensures an application making a request has authority to send that request. It sits between the requesting application and the authoritative source. The SelfServiceApplication and the TestSuite included in this distribution are configured to send requests through the request proxy. So any request they send to the ERP gateway goes through the proxy first. This demonstrates the use of this infrastructure gateway. Currently, the request proxy delivered with this version of the sample enterprise only checks that the requesting application has authority to perform the requested action on the message object contained in the message. It does not perform any "complex" ProxyCommands. For more information regarding the EnterpriseRequestProxy, please refer to the OpenEAI Implementation Strategies document.

The Javadoc for this reference implementation is also included in the distribution and can be found in the `docs/openeai/api` directory. This gateway is implemented in the `org.openeai.implemenations.gateways.requestproxy.EnterpriseRequestProxyCommand` class which is an implementation of the `org.openeai.jms.consumer.commands.RequestCommand` interface.

To stop the gateway, simply perform a `ctrl-c` in the console where the gateway was started.

EnterpriseTransRouter

To start the OpenEAI EnterpriseTransRouter reference implementation, open a command prompt or terminal window, and execute the `OPENEAI_HOME/start` script passing `EnterpriseTransRouter` as an argument.

This gateway is a reference implementation provided by OpenEAI that routes synchronization messages from an authoritative source to all targets that are allowed to consume that message. The sample ERP gateway provided in this distribution publishes synchronization messages to the EnterpriseTransRouter's topic and the router then routes those synchronization messages to the Any OpenEAI Enterprise's EnterpriseWarehouse gateway as well as to the sync consumers used in the OpenEAI TestSuite application. For more information regarding the EnterpriseTransRouter, please refer to the OpenEAI Implementation Strategies document.

The Javadoc for this reference implementation is also included in the distribution and can be found in the `docs/openeai/api` directory. This gateway is implemented in the `org.openeai.implemenations.gateways.transroutergateway.TransRouterCommand`

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

class which is an implementation of the `org.openeai.jms.consumer.commands.SyncCommand` interface.

To stop the gateway, simply perform a `ctrl-c` in the console where the gateway was started.

EnterpriseLoggingService

To start the OpenEAI EnterpriseLoggingService reference implementation, open a command prompt or terminal window, and execute the `OPENEAI_HOME/start` script passing `EnterpriseLoggingService` as an argument.

This is another reference implementation provided by the OpenEAI project that serves two purposes. First, it stores every message delivered to it for historical and analysis purpose into its repository (T_LOGGED_MESSAGE). Second, it stores org.openeai.CoreMessaging/Sync/Error-Sync messages that are published by sync consuming gateways when they have errors processing a message to its repository (T_SYNC_ERROR and T_SYNC_ERROR_MSG). By using the historical data and the error data, analysts are able to go back and review both any errors that occur and the message being processed when the error occurred.

All PubSubProducers used in the sample enterprise are configured with logging producers to publish messages to the EnterpriseLoggingService when they publish any sync message. Additionally, the `org.any-openeai-enterprise.EnterpriseWarehouse` is configured to publish Error-Sync messages when it encounters errors processing sync messages published by the ERP gateway and routed by the EnterpriseTransRouter. For more information regarding the EnterpriseLoggingService, please refer to the OpenEAI Implementation Strategies document.

The Javadoc for this reference implementation is also included in the distribution and can be found in the `docs/openeai/api` directory. This gateway is implemented in the `org.openeai.implemenations.services.els.commands.EnterpriseSyncErrorLogger` and `EnterrpiseSyncLoggerCommand` classes which are both implementations of the `org.openeai.jms.consumer.commands.SyncCommand` interface.

To stop the gateway, simply perform a `ctrl-c` in the console where the gateway was started.

AevGateway

To start the AnyERP Vendor's gateway, open a command prompt or terminal window and execute the `OPENEAI_HOME/start` script passing `AevGateway` as an argument.

This gateway was developed specifically for this sample enterprise. It is the authoritative source for com.any-erp-vendor.Person/BasicPerson, com.any-erp-vendor.Person/EmergencyContact and com.any-erp-vendor.Employee/BasicEmployee messages. The SelfService application along with the TestSuite application both send requests to this gateway (through the proxy) related to these message objects. When this gateway consumes and successfully processes the requests, it will return a reply to

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

the requesting application (through the proxy) and it may publish synchronization messages depending on the action specified in the message.

Please refer to the Javadoc associated to this gateway for more information about what's going on. Javadoc is included with the distribution in the `docs/examples/api` directory and is generated when you build the source code provided.

The processing associated to this gateway is implemented in the `com.any_erp_vendor.messaging.gateways.AevRequestCommand` class which is an implementation of the `org.openeai.jms.consumer.commands.RequestCommand` interface.

To stop the gateway, simply perform a `ctrl-c` in the console where the gateway was started.

EnterpriseWarehouseGateway

To start the Any OpenEAI Enterprise Warehouse gateway, open a command prompt or terminal window and execute the `OPENEAI_HOME/start` script passing `EnterpriseWarehouse` as an argument.

This gateway was also developed specifically for this sample enterprise. It consumes com.any-erp-vendor.Person/BasicPerson synchronization messages published by the com.any-erp-vendor.Gateway and stores them into a relational database repository (T_PERSON, T_ADDRESS, T_PHONE). Specific pieces of the BasicPerson object are broken into related tables.

Please refer to the Javadoc associated to this gateway for more information about what's going on. Javadoc is included with the distribution in the `docs/examples/api` directory and is generated when you build the source code provided.

The processing associated to this gateway is implemented in the `org.any_openeai_enterprise.gateways.warehouse.EnterpriseWarehouseCommand` class which is an implementation of the `org.openeai.jms.consumer.commands.SyncCommand` interface.

To stop the gateway, simply perform a `ctrl-c` in the console where the gateway was started.

SelfServiceApplication

To start the Any OpenEAI Enterprise SelfService application **as an OpenEAI Scheduled Application**, open a command prompt or terminal window and execute the `OPENEAI_HOME/start` script passing `ScheduledSelfService` as an argument.

This application was developed specifically for this sample enterprise. It provides a user interface that allows you to request certain actions be performed on enterprise message objects. It sends requests to the com.any-erp-vendor.Gateway through the org.openeai.EnterpriseRequestProxy and processes the replies. Currently, it allows you

to Query for, Create, Update and Delete com.any-erp-vendor/Person/BasicPerson and com.any-erp-vendor/Person/EmergencyContact objects in the com.any-erp-vendor.Gateway. The requests sent by this application go through the EnterpriseRequestProxy to ensure the application has authority to make the requests.

The application is runnable using two different approaches. This approach uses the OpenEAI ScheduledApplication foundation to start the application. It is configured as a "triggered" application which means it will execute the ScheduledCommand that displays the GUI immediately and then wait to be triggered before it shuts itself down. In this case, that trigger is when the user closes the user interface.

There are several things going on within the application that take advantage of resources available at runtime. For instance, it uses translations listed in the EnterpriseObject (EO) documents for the Address and Phone objects to populate the Address type and Phone type drop-down lists. Additional rules have been specified in the EO documents for the objects being used by this application. For example, formatting rules have been added to the BasicPerson/InstitutionalId and EmergencyContact/OwnerId fields that say the "datatype" for those fields must be "numeric" and the length of data must be exactly 9 (nine). This means, if you try to put non-numeric data or if you put data that is not exactly 9 characters in length, you will get an error when you attempt to perform an action with that object. This happens automatically when data from the GUI is put in the message objects through their setter methods. You can play with these EO documents and specify different rules to see the behavior change. **Note: changes to EO documents are currently only known when the application is restarted. This means if you want to change rules, you must restart the application before those rules will take effect.**

By default, EO documents used by the application and gateways are located in `configs/messaging/Environments/Examples/EnterpriseObjects/3.0` and are broken apart by reverse domain name. Each object has its own EO document (e.g. – BasicPersonEO.xml, EmergencyContactEO.xml etc.).

Please refer to the Javadoc associated to this application for more information about what's going on. Javadoc is included with the distribution in the `docs/examples/api` directory and is generated when you build the source code provided.

The start up mechanism for this version of the application is implemented in the `org.any_openeai_enterprise.applications.selfservice.SelfServiceAppCommand` class which is an implementation of the `org.openeai.afa.ScheduledCommand` interface.

Walkthrough of an Example SelfServiceApplication Session

Once you've got the GUI up, try the following things just to get a feel for how the application works and to see some of the business rules specified in the EO documents at work:

1. Enter "000000001" in the InstitutionalId field and click on the Query button on the "Person Info" tab. You should get a message in the "Results from action" TextBox saying, "`No rows found matching the query.`"

   Notice in the window you started the EnterpriseRequestProxy, the query performed by the SelfService application went through the proxy before it was allowed through to the ERP gateway.

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

2. Enter some data for the person:
    a. Enter a FirstName of "TOM"
    b. Enter a LastName of "SMITH"
    c. Select "Male" from the Gender drop down
    d. Enter a valid BirthDate in the format mm/dd/yyyy. For example "02/16/1968"

3. Now click on the Create button. You should get a message in the "Results from action" TextBox saying, "`Create was successful.`"

   Notice the EnterpriseRequestProxy window again. This Create-Request was only allowed through after the proxy determined it was okay for the self service application to perform that action on that object. We'll be changing some of these proxy rules in a bit to see how they can affect what actions the application is allowed to perform.

   Also, take a look at the EnterpriseTransRouter and EnterpriseWarehouse windows. Notice how the ERP gateway published a com.any-erp-vendor/Person/BasicPerson/Create-Sync and the router routed it to the EnterpriseWarehouse. Then, the EnterpriseWarehouse consumed that sync message and stored it in its own repository.

   Finally, take a look at the EnterprisLoggingService window. Notice how it logs every sync message published by all the other components. This includes the sync message published by the ERP gateway as well as the messages that are routed by the EnterpriseTransRouter. This is because all PubSubProducers used by these components contain a LoggingProducer that is configured to publish anything sent through that PubSubProducer to the "EnterpriseSyncLogger" topic.

4. Now, click the Query button again. Notice how the First and Last names are now mixed case? This is because there is a "Scrubber" specified on those fields in the `NameEO.xml` document that performs automatic case conversion on the data passed to the `setFirstName` and `setLastName` methods in the `Name` Java object. This EO document can be found in `configs/messaging/Environments/Examples/EnterpriseObjects/3.0/com/any-erp-vendor/1.0/Resouces/NameEO.xml`.

5. Now, click the Delete button and notice the behavior in the other gateways. The Delete-Request is again sent through the proxy to the ERP gateway. It processes the request then publishes a Delete-Sync message. This message is then routed by the EnterpriseTransRouter to the EnterpriseWarehouse which deletes the data from its repository appropriately. The EnterpriseLoggingService logs this sync message to its repository as a historical record of the transaction.

6. Now, click the Clear button. We're going to try to create another person.

7. Enter "123" in the InstitutionalId field.

8. Enter some data for the person:
    a. Enter a FirstName of "SALLY"
    b. Enter a LastName of "JONES"
    c. Select "Female" from the Gender drop down
    d. Enter a valid BirthDate

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

9. Now click the Create button. You should get an error in the Results TextBox that looks something like this:

   ```
   Exception executing create.  FIELD: BasicPerson/InstitutionalId is
   invalid.  Exception: Length of field value is 3.  Length of field
   must be 9 exactly.
   ```

   This is telling you that you attempted to put data in the BasicPerson/InstituttionalId field that was not of "Enterprise Quality." This is because the BasicPersonEO.xml document says the InstitutionalId field in the BasicPerson object MUST be exactly 9 characters in length.

10. Change the InstitutionalId to be something that includes non-numeric characters, like "`123abcdef`" and click the Create button again. This time, you should get an error that looks something like this:

    ```
    Exception executing create.  FIELD: BasicPerson/InstitutionalId is
    invalid.  Exception: Format Error:  Datatype for this field must be
    'numeric'.  Field value is abcdefgge  Exception: For input string:
    "123abcdef"
    ```

    Again, this is telling you that you attempted to put data in the BasicPerson/InstituttionalId field that was not of "Enterprise Quality." This is because the BasicPersonEO.xml document says the InstitutionalId field in the BasicPerson object only supports numeric data.

    All of these rules are applied at the time the setter methods are called on the `BasicPerson` Java object in the Self Service Application. This means, the Create-Request isn't even sent to the ERP gateway and the business rules aren't coded in the application itself. The business rules are automatically applied when data is put into the BasicPerson object based on information contained in the EO document(s) used by the object. If for some reason the rules were "turned off" in this application, the gateway would enforce these rules and send an appropriate error in the reply stating the same data quality issues. This is because the first thing the gateway does with the Create-Request message it consumes is populate a `BasicPerson` Java object from DataArea of the message sent in using the `buildObjectFromInput` method that all `XmlEnterpriseObject`s inherit. By doing this, the same rules specified in the EO documents are applied to the data as the `BasicPerson` object is populated by the gateway.

    These rules are specified in the `configs/messaging/Environments/Examples/EnterpriseObjects/3.0/com/any-erp-vendor/Person/1.0/BasicPersonEO.xml` file.

Now, let's change the proxy rules in the EnterpriseRequestProxy associated with the SelfService application. These rules specify which actions that application can perform on which message objects.

1. Open up the `configs/messaging/Environments/Examples/Deployments/AnyOpenEaiEnterprise.xml` document.

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

Note: depending on the tools you use to view and edit XML files, this could be a little tricky since the deployment descriptor contains configuration information for all the applications in our enterprise and it is rather large. We'll try to give you searchable items so you can quickly find the location in the file we're referring to. XML editors such as XMLSpy typically make maintaining files such as the OpenEAI deployment descriptors easier, but they aren't required.

2. Find the MessageGateway with an `id` attribute of `org.any-openeai-enterprise.EnterpriseRequestProxy`.

   This element contains the configuration information for the EnterpriseRequestProxy. This includes the consumers that will be used to proxy requests from various applications and the commands that those consumers execute. Do a search for "ProxyP2PConsumerSSA1". This is the proxy consumer that consumes requests from the SelfService Application. This consumer executes one command for every request sent to it, called "UntrustedRequest", that is implemented in the `org.openeai.implementations.gateways.requestproxy.EnterpriseRequestProxyCommand` class. For that command, there is a set of MessagingComponents (applications) from which this proxy expects requests to be sent. Those MessagingComponents will contain configuration information that specifies which message objects may be messaged about and what actions can be performed by that application on any message object.

   The MessageComponent that maps to the SelfService Application is called `org.any-openeai-enterprise.SelfServiceApplication`. The MessageObjects used by the SelfServiceApplication have a SenderAppId that matches this value. This is how the proxy determines which application is making the request. This information is automatically set when the application makes a request using that message object and becomes part of the ControlAreaRequest in the request sent by the Self Service Application.

   We want to eliminate the ability for this application to Delete BasicPerson objects from our ERP system. To do this, we need to find the BasicPerson MessageObjectConfig that maps a BasicPerson request sent from the Self Service Application to the BasicPerson object used by the proxy to determine which actions can be performed by the Self Service app.

   The objects that the SelfServiceApplication can message about are included in the MessageObjectConfigs container for the `org.any-openeai-enterprise.SelfServiceApplication` MessagingComponent. To find this in the deployment descriptor, do a search for the word "Proxied". This will take you to the FullName element given to the self service application component listed in the proxy configuration. Once you've found that section of the deployment document, we want to find the MessageObjectConfig Element associated with the BasicPerson object. This object is named "BasicPerson.v1_0". By default it has 4 (four) PrimedXmlDocuments associated with it. The proxy uses these primed documents to determine which actions can be performed on the object by the Self Service application. To eliminate the ability of the SelfServiceApplication to Delete BasicPerson objects from our ERP system, remove the `PrimedXmlDocument` Element that has a `type` attribute of "delete". Save the file and restart the EnterpriseRequestProxy.

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

Remember, to restart the proxy, `ctrl-c` in the window where the proxy is running, then execute the `OPENEAI_HOME/start` script passing the `EnterpriseRequestProxy` target to that script.

3. Once the proxy is restarted, go back to the self service application and Create a new person:
    a. Click the Clear button
    b. Enter an InstitutionalId: "000000002"
    c. Enter a FirstName of "Cooper"
    d. Enter a LastName of "Jarvis"
    e. Select "Male" from the Gender drop down
    f. Enter a valid BirthDate
    g. Click the Create button.
    h. After you've verified that the create was successful, click the Delete button. You should get an error similar to this:

```
Exception executing delete.  Error processing the delete request for
object com.any_erp_vendor.moa.jmsobjects.person.v1_0.BasicPerson
Exception: Unknown exception processing Delete request.  Exception:
An error occurred processing the Delete request

Error [0] Number: PROXY-1002
Error [0] Description: Application org.any-openeai-
enterprise.SelfServiceApplication does not have authority to perform
Delete actions on BasicPerson objects.
```

The proxy is now configured so that the self service application cannot perform delete actions on the com.any-erp-vendor/Person/BasicPerson object. Therefore, the request was stopped by the proxy and the ERP gateway was not bothered with this request. The ERP gateway doesn't have to be configured to know what applications can perform which actions on which objects. If we wanted to, we could display a prettier error message that just consisted of the errors returned by the proxy. Namely, the information specified in the "`Error [0] Description:…`" line above.

Continue to play with the Self Service application. You may want to go back into the Deployment Descriptor and add the "delete" PrimedXmlDocument back into the BasicPerson.v1_0 MessageObjectConfig. Add Addresses and Phones to a BasicPerson you've already created. Update that BasicPerson and watch what the other gateways do as a result of that action. Refer to the Javadoc for the application to learn more about how it works and exactly what it does.

Also, the "Emergency Contact" Info tab will allow you to maintain EmergencyContact records associated to a person.

In this version of the application, the "Employee Info" tab doesn't have any functionality so you can add functionality there if you'd really like to get your hands dirty.

To stop the application, simply close the GUI.

To start the Any OpenEAI Enterprise self service application as a "traditional" Java Application, open a command prompt or terminal window and execute the `OPENEAI_HOME/start` script passing `SelfService` as an argument.

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

This is the same application as mentioned in the previous bullet. However, it is NOT started as an OpenEAI ScheduledApplication. Instead, it uses a more "traditional" approach to start the application. The functionality in this version of the application is exactly the same as the ScheduledApplication listed above. The only difference is the way it's started.

Please refer to the Javadoc associated to this application for more information about what's going on. Javadoc is included with the distribution in the `docs/examples/api` directory and is generated when you build the source code provided.

The start up mechanism for this version of the application is implemented in the `org.any_openeai_enterprise.applications.selfservice.SelfServiceApplication` class which simply contains a "`main`" routine which builds an `AppConfig` object based on information found in the deployment descriptor referred to in the `./props/SelfService.properties` file.

To stop the application, simply close the GUI.

Running the Test Suite for the AnyERP Gateway

The distribution also includes a TestSuite that can be run against the `com.any-erp-vendor.Gateway`. The instructions executed by the TestSuite Application are located in the `configs/messaging/Environments/Examples/TestSuites/AevGateway.suite1.xml` file. This file is constrained by the `AevGateway.suite1.dtd` definition and provides all the necessary information needed by the OpenEAI TestSuite Application to test all actions currently supported by the `com.any-erp-vendor.Gateway`. The TestSuite application is also included in the distribution in the `openeai_testsuite.jar` file. It is another OpenEAI ScheduledApplication that executes the instructions listed in the test suite document. This includes, querying for, creating, updating and deleting `com.any-erp-vendor/Person/BasicPerson`, `com.any-erp-vendor/Person/EmergencyContact` and `com.any-erp-vendor/Employee/BasicEmployee` message objects in the repository gated by the `com.any-erp-vendor.Gateway`. This also includes consuming the synchronization messages published by the `com.any-erp-vendor.Gateway` and verifying that the appropriate messages are published.

The OpenEAI TestSuite application is a general use application that can be used to test any OpenEAI gateway. The only thing that changes is the actual test suite document that specifies which actions should be performed and the expected results of those actions.

Running the test suite application is NOT required but it will help ensure you have all the necessary messaging components up and running. It also provides a sample use of the OpenEAI TestSuite application that you can play with by modifying the contents of the AevGateway.suite1.xml file.

The test suite is identified in the deployment descriptor and the properties file as `com.any-erp-vendor.TestSuiteApplication-AevGateway`. To run the test suite application, you may need to change one property in the deployment descriptor based on where you have unpacked the archive. If you unpacked it in the root directory of the file system ("c:>\" or "/") you shouldn't need to change the deployment descriptor. However,

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

if you didn't explode it to the root directory, you will need to change the "`TestSuiteDocUri`" property to the appropriate file system location. There should be 4 (four) occurrences of the property in the descriptor. All four occurrences need to be changed.

It is easiest to just do a "search and replace" in the AnyOpenEaiEnterprise.xml deployment descriptor. You will need to change the `PropertyValue` value for the `TestSuiteDocUri PropertyName` in the deployment descriptor to be appropriate for the location you've exploded the archive to.

For example, by default, the `TestSuiteDocUri` property is specified in the deployment descriptor for the `com.any-erp-vendor.TestSuiteApplication-AevGateway` as:

<PropertyValue>file://localhost/openeai-examples-1.0/configs/messaging/Environments/Examples/TestSuites/AevGateway.suite1.xml</PropertyValue>

This assumes you've exploded the archive to the "root" file system into the "openeai-examples-1.0" directory. If you need or want to explode it somewhere else, you simply need to change this PropertyValue to whatever's appropriate. For example:

<PropertyValue>file://localhost/home/jackson4/openeai-examples-1.0/configs/messaging/Environments/Examples/TestSuites/AevGateway.suite1.xml</PropertyValue>

This will point the TestSuite application to the correct TestSuite document where it will find the instructions that it is to carry out. Once you've made this property change, you can start this ScheduledApplication just like you started all the others:

Windows:

```
C:\>%OPENEAI_HOME%\start.bat TestSuite
```

UNIX:

```
host:/ $OPENEAI_HOME/start.sh TestSuite
```

The TestSuiteApplication will create a summary file called `AevGateway.suite1-summary.xml` in the directory where the application was started by default. This file will include summary information for the test suite that was just run. If all goes well, there should be zero failed TestSteps, TestCases or TestSeries. The sample test suite only includes two TestSeries but within those two TestSeries there are 36 TestCases and 60 TestSteps that exercise the com.any-erp-vendor.Gateway fairly thoroughly.

While the TestSuiteApplication is running, notice that all requests it sends are also going through the EnterpriseRequestProxy just like requests sent by the SelfServiceApplication. Also notice that the sync messages being published by the ERP gateway as a result of the requests being sent by the TestSuiteApplication are being routed back to the TestSuiteApplication's consumers for "sync verification" later. Finally, notice that messages are still being routed to our EnterpriseWarehouse gateway. You might not want to route all these messages to the warehouse under normal circumstances if you're just trying to test the ERP gateway. However, this does demonstrate how this test suite can not only test the intended authoritative source (the ERP gateway in this case) but the resulting sync messages can be delivered to other systems at which point their processing can be verified as well.

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

The Javadoc for the TestSuite Application is also included in the distribution and can be found in the `docs/openeai/api` directory. This application is implemented in the `org.openeai.implemenations.applications.testsuite.TestSuiteScheduledCommand` class which is an implementation of the `org.openeai.afa.ScheduledCommand` interface and `org.openeai.implemenations.applications.testsuite.SyncVerificationCommand` class which is an implementation of the `org.openeai.jms.consumer.commands.SyncCommand` interface.

Compiling the Provided Source Code

All of the source code for the components developed specifically for the sample enterprise is included in this distribution. The binaries are included as well so you don't even need to compile the provided source code. However, if you want to, you can. The following commands will compile everything in the `source` directory off of the base distribution root. After you've successfully compiled the source code, you can copy the generated JAR files to the `configs/messaging/Environments/Examples/Jars/3.0` directory to use them when you start the components listed above if you want. This is NOT required however since the distribution includes these binaries as well.

These commands will build the provided ERP gateway, the Any OpenEAI Enterprise Warehouse gateway, the Any OpenEAI Enterprise Self Service application and both organization's MOA.

Windows:

```
C:\>%OPENEAI_HOME%\build.bat
```

UNIX:

```
host:/ $OPENEAI_HOME/build.sh
```

The resulting classes, JAR files and Javadoc will be stored in `OPENEAI_HOME/source/build` by default.

Generating and Compiling the MOAs for the Fictitious Organizations

Our sample enterprise comes complete with a message tree consisting of artifacts provided by:

- the AnyERP Vendor (message/releases/com/any-erp-vendor)
- our Any OpenEAI Enterprise (message/releases/org/any-openeai-enterprise)
- the OpenEAI Software foundation (message/releases/org/openeai)

These artifacts include enterprise message definitions (DTDs) and sample XML messages that are all by-products of the OpenEAI Methodology. Because these artifacts exist, we can use them to generate the Message Object API (MOA) for a particular organization. This includes generated Java code as well as Enterprise Object (EO) documents that consist of basic formatting information for every object defined by the

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

organization. The MOAs for both the ERP vendor and the Any OpenEAI Enterprise are included with the distribution in both binary and source form. However, if you want to get your hands even dirtier, you can use the OpenEAI MOA Generation application to generate the Java code and EO documents using the artifacts supplied in the distribution.

Note: if you generate these MOAs, the EO documents that get generated will be a "basic" version of the ones that are provided in the `configs/messaging/Environments/Examples/EnterpriseObjects/3.0` directory. The EO documents in that location have been altered a little so they actually have additional, more complex business rules specified in them. If you replace these files with the EO documents generated here, you will lose all those rules so if you want to generate the MOAs, you might want to just leave them where they get generated until you become a little more familiar with the process. Future versions of this document will delve deeper into the issue of providing enterprise data rules via the EO documents as well. For now, they've been provided for you just so you can see what they look like and get an idea of how they're used.

The `configs/messaging/Environments/Examples/Deployment/AnyOpenEaiEnterprise.xml` deployment descriptor included with this distribution is configured by default assuming you've exploded the archive to a "root" directory relative to where you're starting the MOA Gen application. On Windows, this would be something like "C:\" and on Unix it would be something like "/". We know that often it is not possible to explode archives of this nature to the root file system. In cases like this, a few modifications to the deployment descriptor are required. They are listed below. It is easiest to just do a "search and replace" in the descriptor. You will need to change the `PropertyValue` value for the following `PropertyName`s in the deployment descriptor to be appropriate for the location you've exploded the archive to.

- `segmentUri (2 occurrences)`
- `baseUri (2 occurrences)`
- `dtdBaseFilePath (2 occurrences)`
- `targetBasePath (4 occurrences)`

These are all properties used by the MOA generation application to determine the location of resources required to perform the generation. For example, by default, the `segmentUri` property is specified in the deployment descriptor for the `org.any-openeai-enterprise.MoaGenApplication` application is specified as:

```
<PropertyValue>file://localhost/openeai-examples-1.0/message/releases/org/any-openeai-enterprise/Resources/1.0/Segments.dtd</PropertyValue>
```

This assumes you've exploded the archive to the "root" file system into the "openeai-examples-1.0" directory. If you need or want to explode it somewhere else, you simply need to change this PropertyValue to whatever's appropriate. For example:

```
<PropertyValue>file://localhost/home/jackson4/openeai-examples-1.0/message/releases/org/any-openeai-enterprise/Resources/1.0/Segments.dtd</PropertyValue>
```

This same type of change would be required for all the properties listed above if you did not explode the archive to the root file system.

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

Since the OpenEAI MOA Generation application is an OpenEAI based Scheduled Application you start it the same way you start the Scheduled Self Service application listed above. This application will then use properties found in the deployment descriptor to determine which Java/EO files to generate, where to store the generated files, copyright information and several other configurable parameters.

To generate the MOA for the **AnyERP Vendor**, execute the `OPENEAI_HOME/start` script passing `AnyErpMoaGen` as an argument.

To generate the MOA for the **Any OpenEAI Enterprise**, execute the `OPENEAI_HOME/start` script passing `AnyOpenEaiMoaGen` as an argument.

Once you've generated the MOA for both organizations, you can build the corresponding JAR files. To do this, execute the following command:

Windows:

```
C:\>%OPENEAI_HOME%\build.bat moa
```

UNIX:

```
host:/ $OPENEAI_HOME/build.sh moa
```

The build.xml file included in the distribution has a target called "moa" that will build the source code generated by the MOA Gen application. It assumes that the source code has been generated into the `OPENEAI_HOME/moagen/source` directory so if you change the `targetBasePath` property for the MOA generation application in the deployment descriptor, you may have to adjust this target as well in the build.xml file.

The Javadoc for the MOA Generation application is also included in the distribution and can be found in the `docs/openeai/api` directory. This application is implemented in the `org.openeai.implemenations.applications.moagen.MoaGenScheduledCommand` class which is an implementation of the `org.openeai.afa.ScheduledCommand` interface.

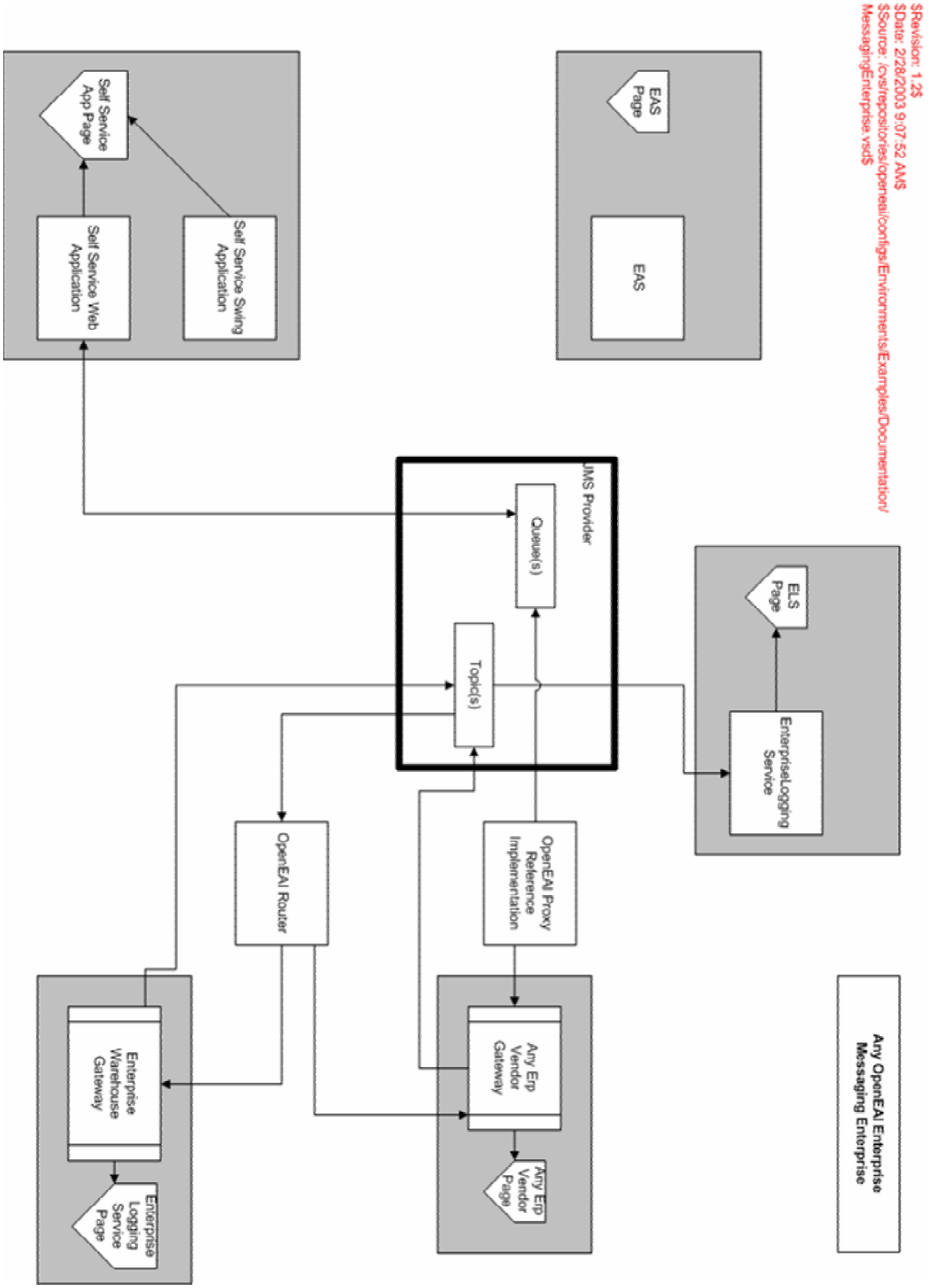
$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

Appendix 1: OpenEAI Sample Enterprise Diagrams

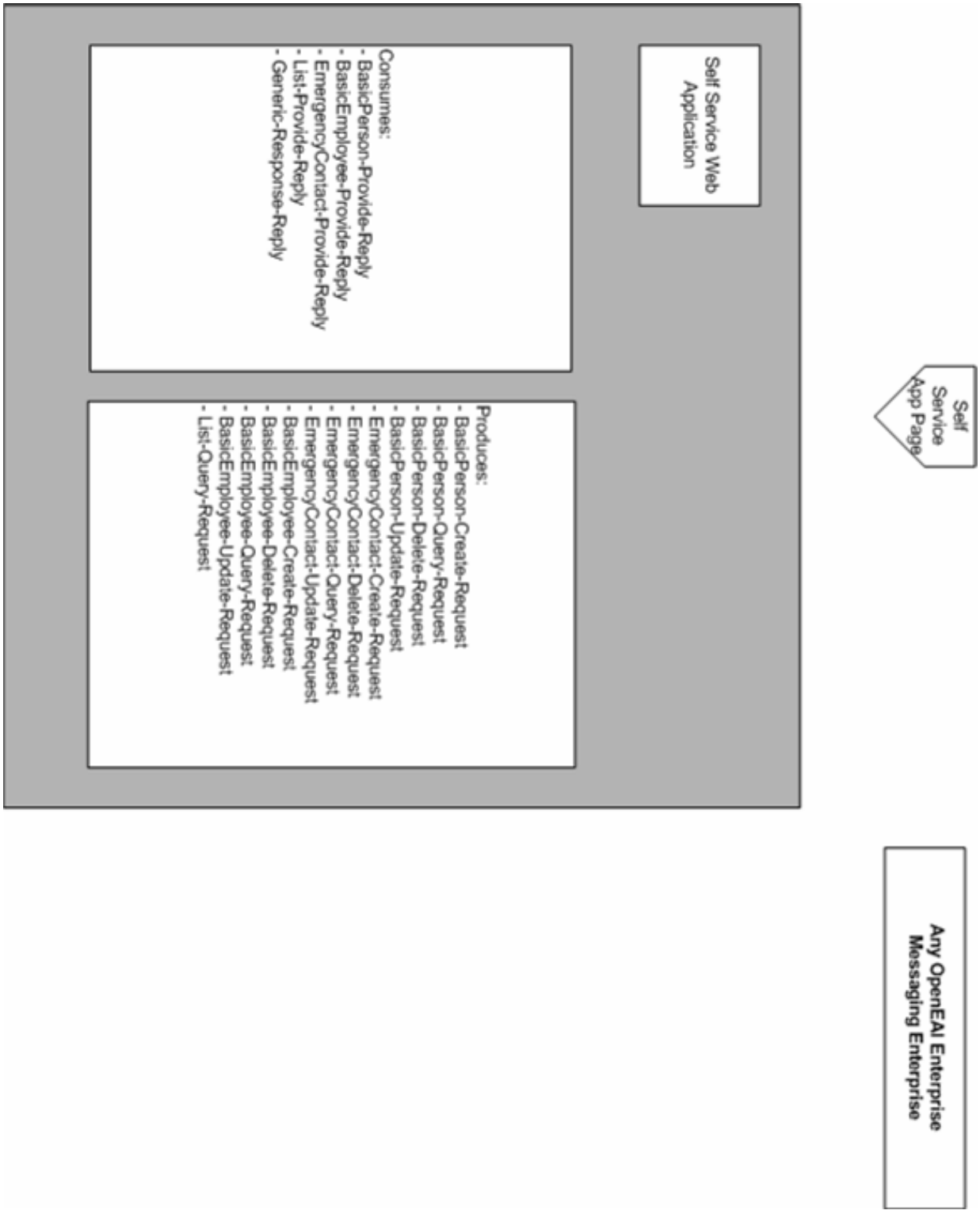
(See next page)

These diagrams are also available in the `OPENEAI_HOME/configs/messaging/Environments/Examples/Documentation` directory but are provided here for convenience.

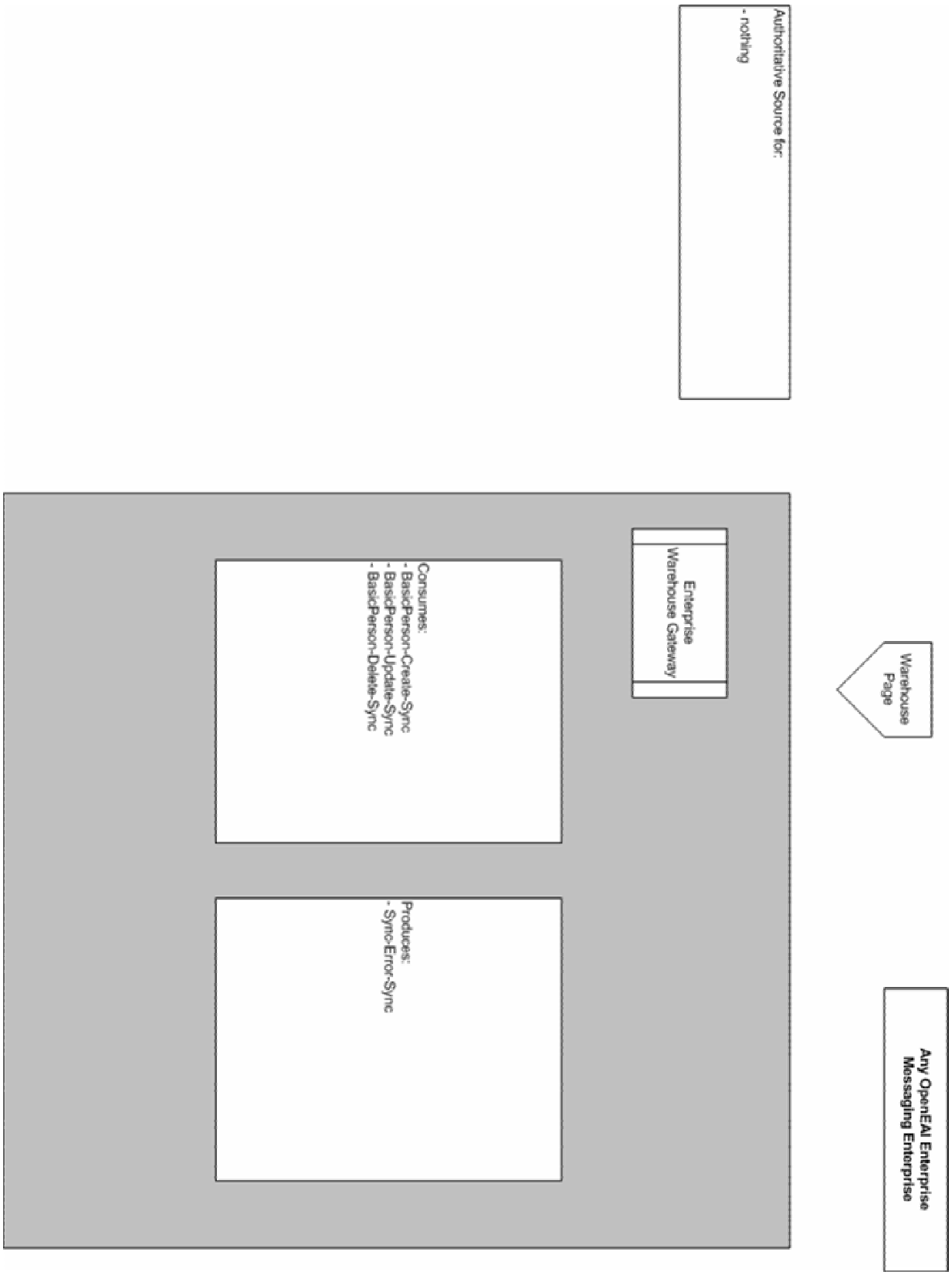
$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$



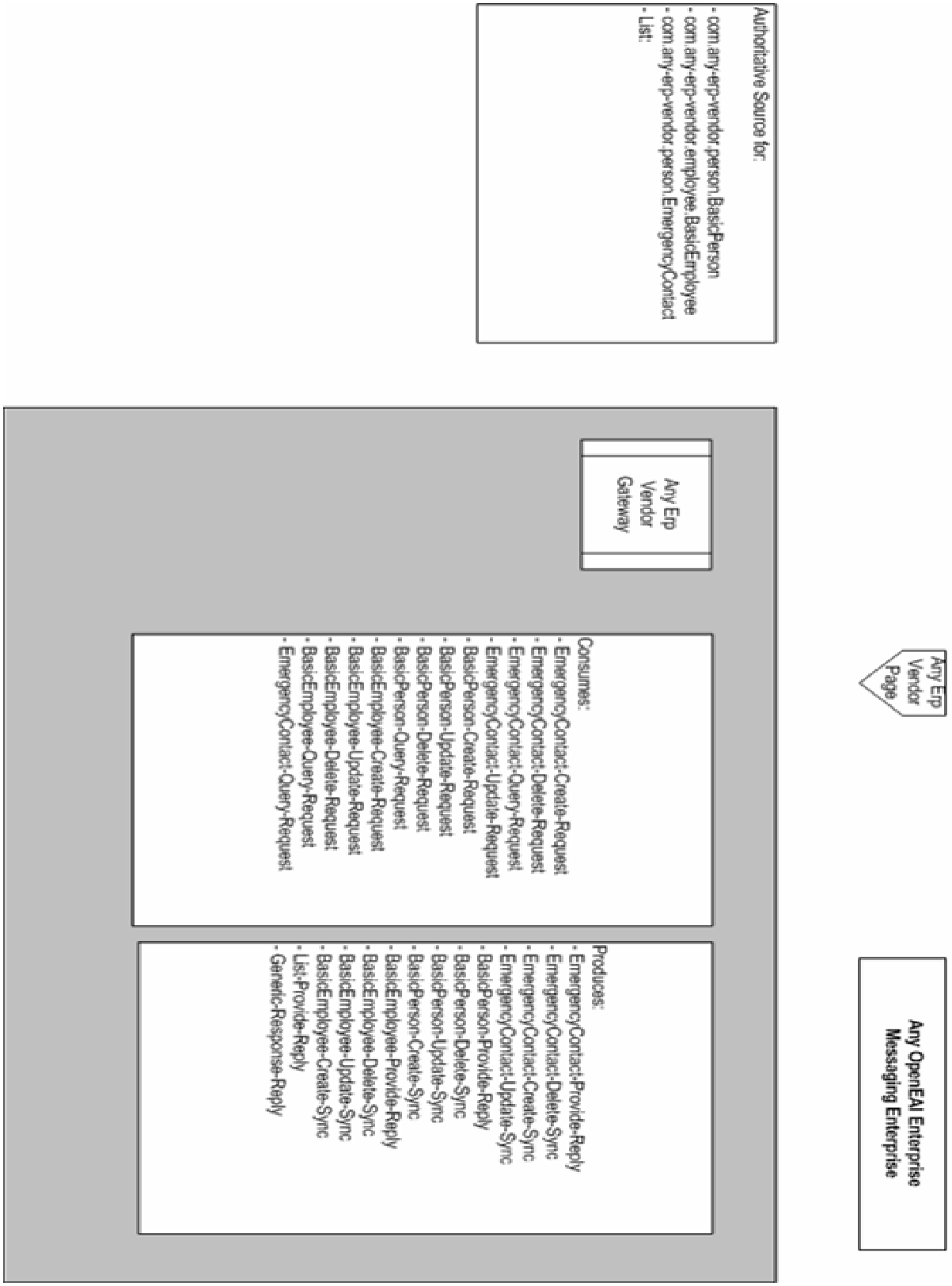
$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$



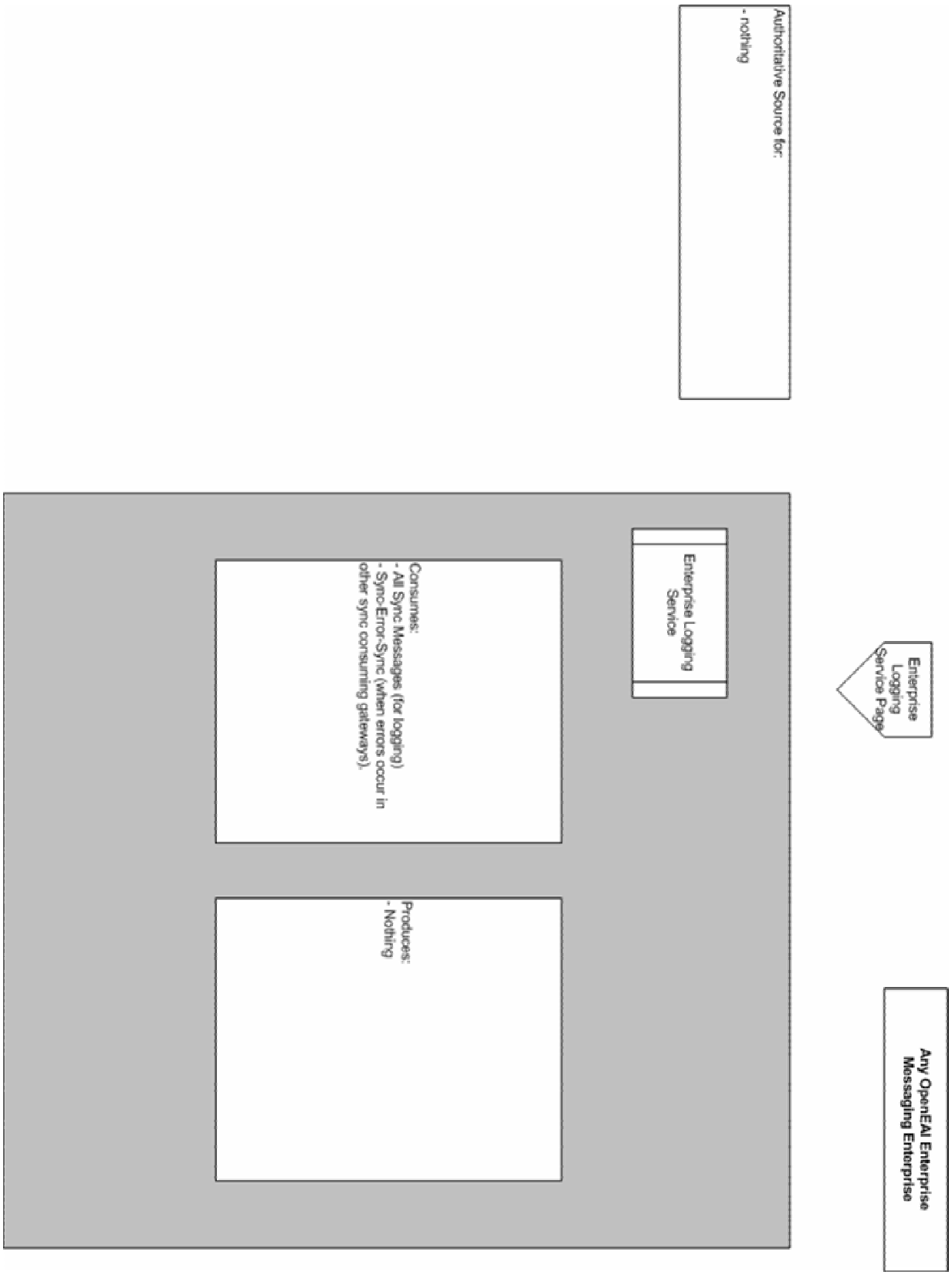
$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$



$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$



$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$



$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

Appendix 2: The GNU Free Documentation License

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document,

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

```
   unaltered in their text and in their titles.  Section numbers
   or the equivalent are not considered part of the section titles.
M. Delete any section entitled "Endorsements".  Such a section
   may not be included in the Modified Version.
N. Do not retitle any existing section as "Endorsements"
   or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or
appendices that qualify as Secondary Sections and contain no material
copied from the Document, you may at your option designate some or all
of these sections as invariant.  To do this, add their titles to the
list of Invariant Sections in the Modified Version's license notice.
These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains
nothing but endorsements of your Modified Version by various
parties--for example, statements of peer review or that the text has
been approved by an organization as the authoritative definition of a
standard.

You may add a passage of up to five words as a Front-Cover Text, and a
passage of up to 25 words as a Back-Cover Text, to the end of the list
of Cover Texts in the Modified Version.  Only one passage of
Front-Cover Text and one of Back-Cover Text may be added by (or
through arrangements made by) any one entity.  If the Document already
includes a cover text for the same cover, previously added by you or
by arrangement made by the same entity you are acting on behalf of,
you may not add another; but you may replace the old one, on explicit
permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License
give permission to use their names for publicity for or to assert or
imply endorsement of any Modified Version.


5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this
License, under the terms defined in section 4 above for modified
versions, provided that you include in the combination all of the
Invariant Sections of all of the original documents, unmodified, and
list them all as Invariant Sections of your combined work in its
license notice.

The combined work need only contain one copy of this License, and
multiple identical Invariant Sections may be replaced with a single
copy.  If there are multiple Invariant Sections with the same name but
different contents, make the title of each such section unique by
adding at the end of it, in parentheses, the name of the original
author or publisher of that section if known, or else a unique number.
Make the same adjustment to the section titles in the list of
Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History"
in the various original documents, forming one section entitled
"History"; likewise combine any sections entitled "Acknowledgements",
and any sections entitled "Dedications".  You must delete all sections
entitled "Endorsements."


6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents
released under this License, and replace the individual copies of this
License in the various documents with a single copy that is included in
the collection, provided that you follow the rules of this License for
```

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document 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.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation 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. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not

$Revision: 1.11$
$Date: 3/11/2003 3:56:42 PM$
$Source: /cvs/repositories/openeai/project/documentation/core/GettingStarted.doc$

```
as a draft) by the Free Software Foundation.
```