Visual Modeler

Installation Guide

Release 9.1

Copyright

This edition applies to the 9.1 Version of Visual Modeler and to all subsequent releases and modifications until otherwise indicated in new editions.

Before using this information and the product it supports, read the information in *Notices* on page 124.

Contents

Visual Modeler Architecture

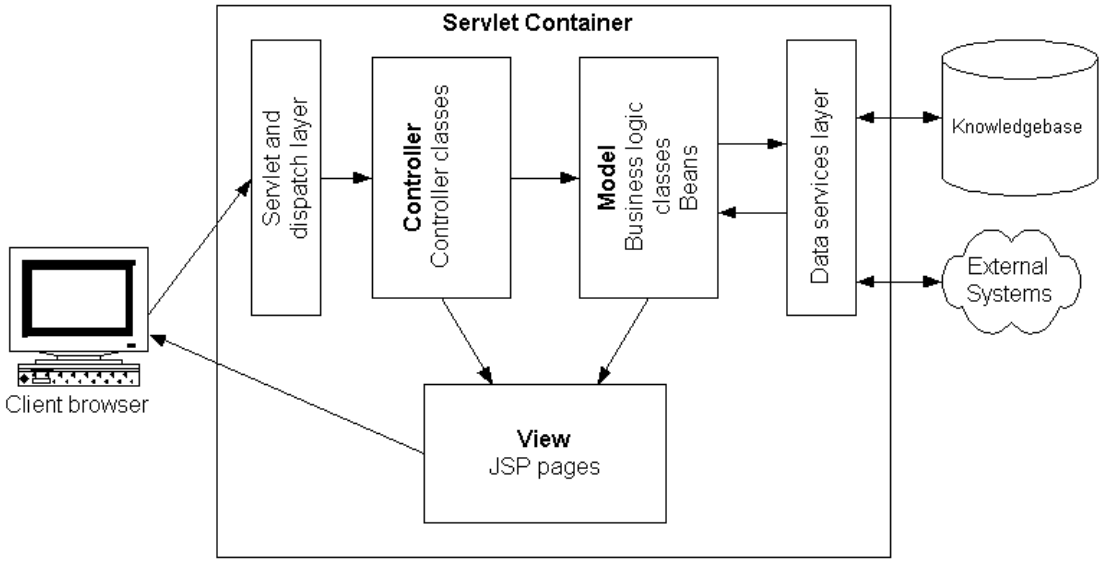
The Visual Modeler is designed to conform to the Java 2 Platform, Enterprise Edition (J2EE) architecture as defined in Java 2 Platform Enterprise Edition Specification, v 1.2 published by Sun Microsystems, Inc. The Visual Modeler server architecture is illustrated schematically in the diagram Logical Representation of the Visual Modeler Server Architecture.

The Visual Modeler is deployed as a Web application that comprises a set of Java classes together with accompanying configuration files, HTML templates, and JSP pages. It must be installed into a servlet container that conforms to the J2EE standard. You can use an existing servlet container that conforms to the standard or deploy the Visual Modeler using the servlet container that we provide as part of the distribution software.

The Visual Modeler is designed to conform to the Model 2 architecture. In this architecture, three functional components referred to as the Model, View, and Controller (MVC) partition the functionality of the server into logically distinct components.

- Model: This component manages the data and business objects that are used by the system.
- View: This component is responsible for generating the content displayed to the user.
- Controller: This component determines the logical flow of the application. It determines what actions are performed on the model and manages the communication between model and view components.

The following figure provides a logical representation of the Visual Modeler server architecture:



The Visual Modeler is designed to be flexible and extensible. You tailor the following components of the Visual Modeler as part of the implementation of your system.

The following table describes the implementation components and their functions:

Component	Function
JSP pages	Customize the JSP pages that determine the look and feel of the Web pages for end-users.
XML schema and data objects	Define the data object schema as a set of XML files. These specify the structure of the data objects and the data sources that provide their content.
Business logic and BizAPI classes	These Java classes determine the business logic that processes requests and messages.
Controller classes	These Java classes handle incoming requests from customer browsers and determine how the responses are displayed.
Configuration files	Use the configuration files to determine the properties of the Visual Modeler and control how incoming requests and messages are processed.

Implementation details are covered in the subsequent sections.

High Availability and Load Balancing

The Visual Modeler supports the ability to distribute request-handling over a number of machines. An enterprise server uses the load-balancing capabilities of the servlet container used to implement the Visual Modeler. Consult your servlet container documentation to see what options are available to you.

Integration Security Issues

Take special care to address security issues. Begin implementation only after you have addressed how users of the Visual Modeler will access data provided by you and your partners.

This discussion covers the following aspects:

- Authentication questions including the use of LDAP
- The use of encryption in storing data in the Knowledgebase
- The use of encryption schemes across your networks and the Internet
- Direct and indirect access to ERP systems
- Your existing firewalls and proxy servers

Installation Worksheet

This topic presents a worksheet to help you gather the information that you need to install and configure the Visual Modeler.

Note: If you do not have this information, then you will not be able to install and run the Visual Modeler.

- Which servlet container are you going to use for the Visual Modeler? What release is this servlet container?
- What version of the Java Servlet Specification does the servlet container support?
- What is the root directory of the servlet container installation? This is referred to as ***container_home*** throughout the documentation.
- What Java Runtime Environment (JRE) are you using? Where is its JAVA_HOME and JDK_HOME?
- What is the database server to be used for the Visual Modeler Knowledgebase?
- What JDBC URL will you use to connect to the Visual Modeler Knowledgebase database server?
  - You must connect to a Universal DB2 Server using a DB2 JDBC driver.
  - You must connect to an Oracle Server using an Oracle JDBC driver.
  - You must connect to a Microsoft SQL Server using a Microsoft SQL Server JDBC driver.
- What is the username and password to be used to connect to the database server?
- What name will you choose for the servlet context to be used for the Visual Modeler?

Hardware Requirements

This section provides a description of the minimum hardware requirements of the Visual Modeler.

Windows 2000

- 512 MB of RAM
- Single or dual Intel processors rated at 400 MHz or faster

UNIX

- 512 MB of RAM
- Single or dual processors rated at 400 MHz or faster

Software Requirements

This section provides a description of the software requirements of the Visual Modeler.

Operating Systems

Hewlett-Packard UNIX

- HP-UX 11.iv3.

  Before deploying the Visual Modeler on HP-UX, you must apply the JavaOOB bundle provided by Hewlett-Packard. See the following URL for further information:

  `http://www.hp.com/products1/unix/java/java2/outofbox/infolibrary/release_notes_java_oob.html`

  JavaOOB is a stand-alone bundle that upon installation, installs startup (RC) scripts, modifies kernel parameters, rebuilds the kernel, and reboots the system. During startup, the startup scripts modify system tunables.

  For the user used to run the servlet container, you must increase the number of files that the user may have open. Do this by running the ulimit command as follows:

  ```
  >ulimit –Sn 1028
  ```

IBM AIX

- AIX 6.11 SP10

Linux

- Red Hat Enterprise Linux 5.5/AP 64-bit Xeon or AMD processor

Microsoft Windows

- Windows 2008 Enterprise /Standard editions.

Sun SPARC Solaris

- Sun SPARC Solaris 10 operating environment or subsequent compatible version.

Java Development Kit

- JDK 6.0.

SDK

You must use JDK 6 to use the SDK to install the Visual Modeler and to create customizations using the SDK.

Servlet Containers

The Visual Modeler has been certified to run in the servlet containers listed in the following table. Install your servlet container before installing the Visual Modeler. Follow and complete the installation instructions for your selected servlet container. We recommend using Tomcat to test your implementation of the Visual Modeler before deploying it to your production system.

Servlet Container	Vendor	Release	Servlet Specification Support
Tomcat	Open Source	6.0 with JDK 6 On Windows installations, you must set the JVM used to the client **jvm.dll**	2.3
WebLogic	Oracle	11gR1 (10.3.2) with JDK 6	2.3
WebSphere	IBM	7.0.0.9	2.3

The Visual Modeler is designed to run in any J2EE-compliant servlet container. Contact Sterling Commerce, an IBM® Company regarding installing your Visual Modeler in another servlet container that meets this specification.

Servlet Container Clustering

The Visual Modeler is designed as a fully J2EE-compliant Web application capable of being deployed in any servlet container that supports the J2EE standard. It can be deployed to all servlet containers that are operating within a cluster or it can be deployed to independent servlet containers that are operating behind a load-balancing solution such as Cisco Local Director.

See "High Availability and Clustering" for more information on implementing a clustered solution. See "Installing a Clustered Implementation" for more information about setting up clustered implementations:

- See "Setting Up a WebLogic Cluster" for information relating to WebLogic clustering.

Network Requirements

The Visual Modeler machine(s) must also be able to establish a JDBC or an ODBC connection to the database server that is used in conjunction with the Visual Modeler. You must ensure that the appropriate database client software is installed on the Visual Modeler machine. See "Installation Overview" for more information.

Browser Requirements

Visual Modeler supports the following browsers:

- MS Internet Explorer 8
- MS Internet Explorer 7
- Mozilla Firefox 3.6
- Mozilla Firefox 3.5
- Google Chrome 5

Security Settings

You must enable your browser to support scripting.

Firefox

1. Select **Options** from the Tools menu.
2. Click the **Content** tab.
3. If the **Enable Java** and **Enable Javascript** check boxes are not already checked, then check them.
4. Click **OK**.

Internet Explorer

1. Select **Internet Options...** from the Tools menu.
2. Click the **Security** tab.
3. Click **Custom Level...**.
4. Under **Scripting**, make sure that Active scripting is enabled.
5. Click **OK**.
6. Click **OK**.

Google Chrome

1. Select **Options** from the Tools menu.
2. Click **Under the Hood**.
3. Click **Content Settings** to open the **Content Settings** window.
4. Click **JavaScript** in the Features section of the window. The **JavaScript Settings** window will open.
5. Select **Allow all sites to run JavaScript** radio button.
6. Click **Cancel**.

Character Sets

Bear in mind that browsers used by Visual Modeler users must support the character sets required to display the data correctly. If your implementation of the Visual Modeler manages data from non-ASCII character sets, then make sure that the browser is set to support Unicode characters.

In particular, make sure that dialog boxes use fonts that support these characters. On Windows systems, this is set using the Display Properties control panel applet.

1. Select **Start -> Settings -> Control Panel**, and start the Display applet. Alternatively, right click the Desktop background and select **Properties**.
2. Click **Appearance**.
3. Select Message Box from the **Item** drop-down list.
4. Select a Unicode font, for example Arial Unicode MS.
5. Click **Apply**.

Database Server Requirements

This section lists the general set-up requirements for Oracle and SQL Server databases. See "Installation Overview" for details.

The Visual Modeler requires one of the following database servers to act as the Knowledgebase database:

- Microsoft SQL Server 2005 SP2 or Microsoft SQL Server 2008
- Oracle 11.1.0.7 or Oracle 11.2.0.1
- DB2 9.7.2

Note: We recommend that you run the database server on a separate machine from the Visual Modeler.

You must ensure that there is a valid userid (username/password pair) set up on the database that acts as the authenticated userid for all Visual Modeler connections to the database. This userid must have the necessary privileges to create, modify, and execute database objects. Make sure that the database default character set is set to UTF-8 Unicode.

Make sure that you have the appropriate client tools installed on the Visual Modeler machine(s). In particular, make sure that you have or can obtain the appropriate JDBC library files from Microsoft, Oracle, or IBM if you plan to deploy against SQL Server, Oracle, or DB2 respectively.

Database Owner Requirements

The database owner must have the following privileges:

- CREATE TABLE
- CREATE VIEW
- CREATE SYNONYM
- CREATE DATABASE LINK
- CREATE TRIGGER
- CREATE SESSION
- CREATE PROCEDURE
- CREATE SYNONYM
- UNLIMITED TABLESPACE

The database owner must have the following roles:

- CONNECT
- RESOURCE

Microsoft SQL Server Requirements

This section describes requirements for running the Visual Modeler with Microsoft SQL Server.

You must have a Microsoft SQL Server Release 2005 SP2 or Microsoft SQL Server 2008 running on Windows 2008 Server. You must connect to this SQL server using the JMicrosoft SQLServer JDBC driver 3.0, version 3.0.1301.101. Use the driver jar called sqljdbc4.jar. You can download the JDBC drivers from: http://www.microsoft.com/downloads/en/details.aspx?FamilyID=a737000d-68d0-4531-b65d-da0f2a735707&displaylang=en.

Configure the SQL Server to not return UPDATE counts. You can do this by executing the following commands in the SQL Server Management Studio:

```
USE master;
GO
EXEC sp_configure 'disallow results from triggers', '1';
RECONFIGURE WITH OVERRIDE;
```

When you set up the SQL Server, you must specify that the database character set is Unicode. You must also set up the SQL Server client software on the Visual Modeler machine to use Unicode. On the servlet container machine:

1. Start the SQL Server Client Network Utility.
2. On the DB-Library Options tab, uncheck **Automatic ANSI to OEM conversion**.
3. Click **Apply**, and then **OK**.

If you use SQL Server, then a search that includes the following characters will return zero results: é, ö, ü, ç.

Oracle Requirements

This section describes the requirements for running the Visual Modeler with Oracle 11.1.0.7 or Oracle 11.2.0.1 using a compatible JDBC driver.

If you do not have a local installation of Oracle products on the installation machine, download the appropriate JAR file from the Oracle Technology Network Web site. The URL is:

```
http://www.oracle.com/technology/index.html
```

Search for “JDBC driver”.

When setting up the database, select the Custom option in the Database Configuration Assistant in order to set the character set to UTF-8. You can verify that the correct settings are set by invoking a SQL*PLUS session to the database server and entering:

```
SELECT * FROM NLS_DATABASE_PARAMETERS WHERE PARAMETER = 'NLS_CHARACTERSET';
```

You should see:

```
NLS_CHARACTERSET UTF8
```

UNIX

To use the Oracle OCI driver to connect from the Visual Modeler to the database server, set the following environment variable:

```
NLS_LANG=AMERICAN_AMERICA.UTF8
```

DB2 Requirements

IBM DB2 Version 9.7.2 or a subsequent compatible version. The database server must support stored procedures, and hence must have the C compiler installed. Make sure that you have installed the DB2 Application Development Client as part of the DB2 server. If you plan to use the gcc compiler, log in as the UNIX user used to run the DB2 database server, and then you must run a command like this (all one line):

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND usr/vac/bin/xlc
-I$HOME/sqllib/include SQLROUTINE_FILENAME.c
-bE:SQLROUTINE_FILENAME.exp -e SQLROUTINE_ENTRY -q64
-o SQLROUTINE_FILENAME -L$HOME/sqllib/lib -ldb2"
```

The exact form of the command will depend on your environment: in particular, on the location of your C compiler. The -q64 flag is used only if DB2 is running in 64-bit mode. Check the DB2 documentation for details. Make sure that you set the codeset to be UTF-8; for example:

```
create database <database_name> codeset UTF-8 territory US
collate using system DFT_EXTENT_SZ 2
CATALOG TABLESPACE managed by Database using (File 'catalogfile'
10000)
USER TABLESPACE managed by Database using (File 'userfile' 10000)
TEMPORARY TABLESPACE managed by Database using (File 'tempfile' 10000)
```

You must set the STMTHEAP database server parameter to 10000. You must set the APPLHEAPSZ parameter to at least 2048. You can update parameters by logging in to the DB2 database server and then entering:

```
>db2 update dbm cfg for database <database_name> using STMTHEAP 10000
>db2 update dbm cfg for database <database_name> using APPLHEAPSZ 2048
```

You must also ensure that the pagesize for the database is set to 16K. A default installation of DB2 sets the pagesize to 4K. The following DB2 commands increase the pagesize for a sample database:

```
create bufferpool datapool size 50 pagesize 16K
```

Restart the DB2 database server to activate the bufferpool:

```
db2stop force;
db2start;
```

Create the tablespaces:

```
create system temporary tablespace temporary pagesize 16K managed by
database using File(file) extentsize 2 bufferpool datapool
create tablespace debsspace pagesize 16K managed by database
using File(file) extentsize 2 bufferpool datapool
```

You can drop the USER tablespace so that your tablespace can be the default for everything that is created.

Check your DB2 Universal Server documentation for further details.

Installation Overview

This topic describes briefly the stages involved in installing the Visual Modeler.

Preparing to Install

This stage covers how to prepare for an efficient installation: identifying known issues, identifying the database information you will need, identifying the servlet container root directory and the destination directory for the `Sterling.war` file, and so on.

Installing the Software Development Kit

This stage covers how to install the Visual Modeler Software Development Kit (SDK). After you install the SDK, you can manage the installation of the Visual Modeler using targets provided by the SDK.

Installing the Visual Modeler Using the SDK

This stage covers how to use the SDK to install the Visual Modeler Web application. After you complete this stage, you deploy the **Sterling.war** to a servlet container as a web application.

Deploying the Visual Modeler

This stage covers how to deploy the `Sterling.war` file as a Web application. How you deploy depends on which servlet container you are using.

Database Server Steps

This stage covers the preliminary configuration steps you perform to prepare for the following tasks:

- Creating the Knowledgebase schema
- Populating the Knowledgebase

The preparation steps require some knowledge of the database server that you plan to use for the Knowledgebase.

Preparing to Install

1. Read the `Readme.txt` file supplied on the Visual Modeler CD-ROM for any final instructions not included in this guide.
2. Determine that your servlet container supports the Java Servlet Specification 2.3.
3. Ensure that your databases and database servers are tuned appropriately for your implementation. Default settings may not always work, for example, you may have to increase settings such as the DEFAULT CURSORS setting. Consult your DBA to ensure proper database and database server tuning is complete before setting up the Visual Modeler application.
4. Determine the database connection information used to connect the Visual Modeler to the Knowledgebase database server:
   - For an Oracle database server, you must set up a TNS alias to access the database from the SDK machine.
   - For a SQL Server database server, you must set up an ODBC source to access the database from the SDK machine. The name of the ODBC source must be the same as the name of the machine on which the SQL Server instance is running. If you are using a SQL Server 2005 database server and plan to use JDBC to access the database, ensure that your SQL Server JDBC driver is at least version 1.1 or higher.
   - For a DB2 database server, you must set up a DB2 alias to access the database from the SDK machine.
5. Identify any known issues.
6. Identify the location of the servlet container root directory, ***`container_home`***.

   In a typical installation on Windows 2008, the location is as follows:

Servlet Container	Home Location
WebSphere	C:\WebSphere\AppServer\
Tomcat	C:\Program Files\Apache Software Foundation\Tomcat 6
WebLogic	C:\bea\weblogic92

In a typical installation on UNIX, the location is as follows:

Servlet Container	Home Location
WebSphere	/usr/WebSphere/AppServer/
Tomcat	/usr/local/tomcat/
WebLogic	/apps/bea/weblogic92/

7. Identify the destination directory location, ***debs_home***, of the Visual Modeler. This is usually a sub-directory of ***container_home***, but its precise location can vary from one servlet container to another.

   In a typical installation on Windows 2008, the location is as follows

Tomcat	C:\Program Files\Apache Software Foundation\Tomcat 6\webapps\
WebSphere	C:\WebSphere\AppServer\hosts\default_host\
WebLogic	C:\bea\weblogic92\user_projects\domains*mydomain*\applications\

8. Remove any existing deployment of the Visual Modeler application from the ***debs_home*** directory before starting the installation procedure. This requires using the servlet container's administrative console to remove the Web application, and then physically deleting the directories and files.
9. If you are implementing IBM® Sterling Configurator, create an environment variable to specify the location of your JDK on the servlet container machine. For Windows systems, at the command line, enter:

   ```
   set JDK_HOME=<path_to_JDK>
   ```

   For example:

   ```
   set JDK_HOME=c:\jdk6
   ```

   For UNIX systems, enter:

   ```
   setenv JDK_HOME <path_to_jdk>
   ```

   For example:

   ```
   setenv JDK_HOME /usr/java/jdk6
   ```

   You must also set a JAVA_HOME environment variable on the machine used to run the SDK. If you use the same machine for both functions, then JAVA_HOME and JDK_HOME must have the same value.

   The PATH environment variable must include the **JDK_HOME/bin/** directory.
10. If you are implementing Sterling Configurator, the following JAR files, which are located in the `<INSTALL_DIR>/jar/smcfs/<current_version>` folder, must be copied to the folder specified in the path configured for the rules:
    - `cmgt-rulesEngine.jar`
    - `cmgt-configuredItem.jar`
    - `cmgt-configurator.jar`

Installing the Software Development Kit

You can use the Software Development Kit (SDK) to install the Visual Modeler and to manage customizations to your implementation. You must use JDK 6 and Version 3.5.4 of the SDK to install the Visual Modeler.

This topic covers only those SDK functions used to install the Visual Modeler. The Software Development Kit provides a comprehensive guide to the SDK.

To install the SDK, identify or create a directory on your machine to use as the development directory, ***sdk_home***.

1. If the JAVA_HOME environment variable is not already set, set it to the location of your Java Development Kit. For example:

   ```
   set JAVA_HOME=<path_to_JDK>
   ```

   For example:

   ```
   set JAVA_HOME=c:\jdk6
   ```

   For UNIX systems, enter:

   ```
   setenv JAVA_HOME <path_to_jdk>
   ```

   For example:

   ```
   setenv JAVA_HOME /usr/java/jdk6
   ```

   For the Bourne shell, enter:

   ```
   export JAVA_HOME=<path_to_jdk>
   ```

   For example:

   ```
   export JAVA_HOME=/usr/java/jdk6
   ```

2. Version 3.5.4 of the SDK is delivered as a JAR file, `sdk-framework.jar`. Unjar the JAR file in the `sdk_home` directory.
3. At the command line, navigate to the `sdk_home` directory, and enter:

   ```
   sdk setup
   ```

4. For Windows systems: if your system directory is not **c:/winnt/system32**, edit the *`sdk_home`*`/my_sdk.properties` file and update the windows.system.dir property.
5. After you install Release 9.1, set the version number on the SDK Index HTML page by running the `generateIndexFile` target.

To continue with the installation, proceed to "Installing the Visual Modeler Using the SDK".

Installing the Visual Modeler Using the SDK

This section describes how to use the SDK to install the Visual Modeler on either a Windows 2000/2008 or a UNIX operating system. Successful installation creates a `Sterling.war` file on your SDK machine.

Note: Please install your servlet container before attempting to install the Visual Modeler.

Follow the steps described in "To Install the Visual Modeler as a Full Release".

Note: If your release number is not Release 9.1, then substitute the string for the release wherever these instructions refer to Release 9.1.

To Install the Visual Modeler as a Full Release

You must perform this task as a user with local machine administration privileges.

1. Locate the release `Sterling.jar` file for this release (called something like `Sterling-9.0-amberOne-9_0-RC2.jar`) and copy it to a temporary location on your system.
2. At the command line, navigate to the *`sdk_home/`* directory.
3. Edit the *`sdk_home/`*`my_sdk.properties` file to specify the value of the container.home and app.name properties. The SDK uses these properties to determine the values of other properties as follows:
   - deploy.home is set to container.home/apps.dir. The deploy.home property is used to specify the servlet container deployment directory.
   - The app.name is the name of the Web application. The app.name is usually the same as the Web application's directory under the deployment directory. In this guide, we assume that the value of this property is "Sterling". If you want to change the name of the Web application (for example, to change the name of the generated WAR file), modify the app.name entry in the `my_sdk.properties` file.
   - debs.home takes the value container.home/apps.dir/app.name.

   The project.name property defined in the `sdk-settings.properties` file is used to specify the name of the project directory in the SDK. In general, this is not the same as the app.name property.

   Notes:

   - If you are running the SDK on UNIX, you may have to add execute permission to the `sdk_home/sdk.sh` file:

     ```
     chmod +x sdk_home/sdk.sh
     ```

   - After you run the merge target, you may have to modify the permissions on the `sdk_home/workspaces/project/OracleCreateSchema.sh` file.
4. If you are running the SDK on Windows 2008, you must perform the following tasks to ensure that the environment variables are picked up correctly:

- Edit the `sdk.bat` file located in the *`sdk_home/`* directory by adding the `-Denv.COMERGENT_SDK_HOME=%COMERGENT_SDK_HOME%` argument. For example:

  ```
  "%JAVA_HOME%/bin/java" -Xmx256m -classpath %CP%
  -Denv.COMERGENT_SDK_HOME=%COMERGENT_SDK_HOME% org.apache.tools.ant.Main
  -emacs -buildfile %COMERGENT_SDK_HOME%\sdk.xml %ARGS%
  ```

- Edit the `sdk.xml` file located in the `sdk_home/` directory by commenting the `property environment` statement. For example:

  ```
  <!-- <property environment="env"/> -->
  ```

5. Run the newproject target, specifying a project name for this installation. For example:

   ```
   sdk newproject matrix
   ```

   This target can take a few minutes to run.

6. You can also set other properties such as the logging level. Values you set here are automatically merged into the `prefs.xml` configuration files under the `sdk_home/builds/project/` directory. See "Email Addresses" for information about the email addresses set in the properties files.

7. Database targets:

   a. If you plan to run the Knowledgebase on Oracle, run the installOracle target, specifying the location of the Oracle JDBC JAR file, to copy the Oracle JDBC drivers JAR file to the project files. The name and location of the JAR file will vary from installation to installation. For example:

      ```
      C:\oracle\product\10.2.0\client_1\jdbc\lib\ojdbc14.jar
      ```

      or
      `/opt/oracle/product/10.2.0/client_1/jdbc/lib/ojdbc14.zip`

      For example:

      ```
      sdk installOracle /tmp/Oracle_jdbc.jar
      ```

      This copies the JAR file to the `WEB-INF/lib/` directory in the release directory. It also renames the file to `oraclejdbc.jar`.

   b. If you plan to run the Knowledgebase on SQL Server 2005, run the installMSSQLJDBC target, specifying the location of the SQL Server JDBC JAR file. For example:

      ```
      sdk installlMSSQLJDBC "C:\Program Files\Microsoft SQL Server
      2005 JDBC Driver\sqljdbc_1.1\enu\sqljdbc.jar"
      ```

      Note: For SQL Server 2005, your JDBC driver must be version 1.1 or higher.

   c. If you are planning to run the Knowledgebase on DB2, then run the installDB2 target, specifying the location of the DB2 JDBC JAR file, to copy the DB2 JDBC drivers JAR file to the project files. For example:

      ```
      sdk installDB2 /tmp/db2java.zip
      ```

      The name and location of the JAR file will vary from installation to installation: typical locations are `C:\Program Files\IBM\SQLLIB\java\db2jcc.jar` or

`$DB2_HOME/java/db2jcc.jar`. This target copies the JAR file to the `WEB-INF/lib/` directory in the release directory. It also renames the file to `db2jdbc.jar`.

The license JAR files (`db2jcc_license_cisuz.jar` and `db2jcc_license_cu.jar`) must be copied from your DB2 installation directory to the `/WEB-INF/lib/` directory.

8. Run the env.setDBType target to set the appropriate database type:

   ```
   sdk env.setDBType Oracle
   ```

   or

   ```
   sdk env.setDBType MSSQLJDBC
   ```

   or

   ```
   sdk env.setDBType DB2
   ```

9. To set the database password to be encrypted:

   a. Set the Encrypted flag to true:

      ```
      sdk setVal DataServices.DataSource.ENTERPRISE.Encrypted true
      ```

   b. Encrypt the database password string:

      ```
      sdk encryptVal <password>
      ```

      The result is an encrypted version of the password.

   c. Edit the encrypted password string into the appropriate properties file as the relevant password property. For example:

      ```
      ORACLE_PASSWORD=<encrypted_password>
      ```

   The encrypted form of the password is entered into the schema creation scripts that are used by the createDB target, so you must run the createDB target before you encrypt the password.

10. If you plan to support locales other than the default U.S. English (en_US) locale, perform the localization installation and customization steps. See "Localization Concepts" for instructions.

11. Run the merge target to create your first build in the `builds/` directory.

    ```
    sdk merge
    ```

    This target can take a few minutes to run. The sdk merge target copies the Web application files from the releases directory and merges in the files and properties currently in your project directory. If the target fails with a message relating to the JDBC driver, check that you have run the database install targets appropriately:

    - If you are creating an Oracle or DB2-based project, then check that you have run the installOracle or installDB2 target, and ensure that the `oraclejdbc.jar` or `db2jdbc.jar` file is now in the *`sdk_home`*`/releases/debs-Aries/overlay/WEB-INF/lib/` directory.
    - If you are creating a SQL Server 2005-based project, then check that you have run the installMSSQLJDBC target, and ensure that the `mssqljdbc.jar` file is now in the *`sdk_home`*`/releases/debs-Aries/overlay/WEB-INF/lib/` directory**.**

12. Run the distWar target to create the WAR file that you will deploy. For example:

    ```
    sdk distWar
    ```

    This target can take a few minutes to run. The generated WAR file is in *sdk_home*`/dist/`. Its name is determined by the app.name and deploy.environment properties and a timestamp. You can rename the WAR file to `Sterling.war`.

13. Alternatively, you can run the dist target. This creates a JAR file that contains the WAR file along with JAR files that provide the SQL scripts and XML data files. You can install the Visual Modeler from this JAR file by following the instructions provided in "Installing the Reference Visual Modeler". Note that you can skip the steps to set the database properties information because your `prefs.xml` file already has this information.

14. In the *sdk_home*`/dist/time_stamp/` directory, rename the generated `prefs_env.xml` file to `prefs.xml` file. This is the basic configuration file that must be copied under the home directory of the user running the servlet container.

15. Run the createDB target to create the Knowledgebase schema.

    a. If you are running the Knowledgebase on either Oracle or SQL Server, run:

       ```
       sdk createDB
       ```

       If you are running the Knowledgebase on SQL Server 2005 and want to use the JDBC connection to connect to it at runtime, then you must still use the ODBC scripts to create the database schema. Consequently, you must set the ODBC properties as well as the MSSQLJDBC properties before running this target.

    b. If you are running the SDK on a Windows machine and using DB2 as the Knowledgebase database server, then before running this target, check that you can connect to the DB2 database by running:

       ```
       db2 connect to <database> user <username> using <password>
       ```

       Use the values that you have specified in the SDK properties file. Before running createDB, enter db2cmd. This opens a DB2 CLP window, and then run the createDB target from this window. Once you have run the createDB target, then exit the DB2 CLP window.

    c. Check the results of the createDB target by looking at log files in the *sdk_home*`/logs/projects/`*project_name* directory.

16. Run either the loadDB target (to load the minimal data set) or the loadMatrixDB target (to load the full Matrix reference data set) into the Knowledgebase.

    ```
    sdk loadDB
    ```

    or

    ```
    sdk loadMatrixDB
    ```

    Check the results of the loadDB or loadMatrixDB targets by looking at the *sdk_home*/workspaces/*project_name*/debs.log file.

    Check for an error similar to the following:

    ```
    File: WEB-INF/xmldata/Minimal/Partner:
    [CMGT_LOOKUP_CACHE_ENTRY_DOESNT_EXIST] error: "No Lookup Cache entry for
    locale en_ Lookup Category PartnerStatus Lookup Code 10."
    com.comergent.api.dataservices.NoLookEntryExistForSourceException:
    [CMGT_LOOKUP_CACHE_ENTRY_DOESNT_EXIST] error: " No Lookup Cache entry for
    locale en_ Lookup Category PartnerStatus Lookup Code(or Lookup Description)
    10."
    This error is caused by an invalid locale specification in your system
    environment. On UNIX, check whether the LANG environment variable has a
    country setting. It should look like this:

    LANG=en_US.UTF-8
    ```

Email Addresses

As part of implementing the Visual Modeler, set up the email addresses used by the system. They reside in one of the following locations:

- Configuration Files
- Minimal Data

Configuration Files

Email addresses set in the configuration files are used by applications when sending email from the system. You set the values for these addresses in the `*.properties` files used by your SDK. When you run the SDK merge target, these values are merged into the configuration files. The following email addresses must be set:

- SMTP_SENDER: used as the From address when email is sent from the Visual Modeler.
- INVOICE_EMAIL_ADDRESS: the email address of an enterprise user to whom emails are sent relating to invoices. The user must have the AccountReceivable role.
- RFQ_EMAIL_ADDRESS: the email address of an enterprise user to whom emails are sent relating to RFQs: the user must have the CustomerServiceRepresentative role.
- ENTERPRISE_EMAIL_ADDRESS: set to the same value as SMTP_SENDER.
- SMTP_RECIPIENT: no longer used.

Minimal Data

When you load the minimal data, you create some partners and some users. The email addresses associated with these are currently set to changeme@changeme.com. You should change these values to more suitable values before loading the data.

The Partner Profile email addresses for the Enterprise, AnonymousUserPartner, and RegisteredUserPartner should all be set to a system administrator email account.

The email addresses for the users (admin, ERPAdmin, and AnonymousUser) should be set to the email address of a system administrator at your implementation. They can be changed through the Visual Modeler user interface.

Deploying the Visual Modeler Application

After you successfully install the `Sterling.war` file, you must deploy it as a Web application. This process varies from one servlet container to another. Check your servlet container documentation for further details. This section provides specific deployment steps for each of the supported servlet containers.

You must identify the operating system user that runs the servlet container. You must copy the `prefs.xml` configuration file to a sub-directory of the home directory of this user. The sub-directory is called `user_home/cmgt/debs/conf/`.

The *`user_home`* home directory location can vary from one operating system to another. The following table provides some typical locations:

Operating System	Standard Home Directory
Windows	C:\Documents and Settings*username*
Solaris	/export/home/*username*
Linux	/home/*username*

Note: You can put this file in an alternate location which is specified as the comergent.preferences.store system property. If you do so, then you must specify its location so that it can be read when the servlet container starts the Visual Modeler Web application. You can do this in the following ways:

- Set its location as a system variable. For example, add the following to the command that starts the servlet container:

  ```
  -Dcomergent.preferences.store=/home/scowner/tomcat6014/prefs.xml
  ```

- Set its location in the `WEB-INF/web.xml` using the following element:

  ```
  <init-param>
     <param-name>comergent.preferences.store</param-name>
     <!--BEGIN:com.comergent.tools.ant.taskdefs.SetFileContents
           (do not modify this tag) -->
     <param-value>/home/scowner/tomcat6014/prefs.xml</param-value>
     <!--END:com.comergent.tools.ant.taskdefs.SetFileContents
           (do not modify this tag) -->
     <description>Location of Comergent's preferences store
     </description>
  </init-param>
  ```

We provide deployment steps for the following servlet containers:

- Deploying the Visual Modeler on Apache Tomcat
- Deploying the Visual Modeler on IBM WebSphere
- Deploying the Visual Modeler on WebLogic

XML Parser Settings

To ensure that the correct Java classes are used for the XML processing performed by the Visual Modeler, you must ensure that the Java Virtual Machine settings specify the correct classes. In general, you can either set the classes as additional parameters in the command line that starts the servlet container or you can specify them as parameters for the Web application.

The following sections describe the steps necessary to set the command line parameters for each of the supported servlet containers. To set the parameters for the Web application, add the following to the `web.xml` file located in the *`debs_home`*`/Sterling/WEB-INF/` directory:

```
<context-param>
   <param-name>Comergent.xml.SAXParserFactory</param-name>
   <param-value>
      org.apache.xerces.jaxp.SAXParserFactoryImpl
   </param-value>
   <description>SAX Parser factory configuration</description>
</context-param>
<context-param>
   <param-name>Comergent.xml.DocumentBuilderFactory</param-name>
   <param-value>
      org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
   </param-value>
   <description>DOM Parser factory configuration</description>
</context-param>
```

Note: These settings are overridden by values set at the command line.

Deploying the Visual Modeler on Apache Tomcat

If you have installed the `Sterling.war` file into the default Web applications directory, *`container_home`*`/webapps/`, then Tomcat can automatically detect it, and it is deployed automatically when you start Tomcat. Make sure that there is no pre-existing `Sterling/` directory already in *`container_home`*`/webapps/`.

To deploy the Visual Modeler on Apache Tomcat:

1. Ensure that your JDK `bin` directory is defined in the system PATH environment variable. For example:

   ```
   C:\Program Files\Java\jdk1.6.0_08\bin
   ```

2. On Windows installations of Tomcat, you must use the client version of the Java VM DLL. Open `Start -> All programs -> Apache Tomcat 6.0 -> Configure Tomcat`, and on the Java tab, set the Java Virtual Machine to the location of the client JVM DLL (for example: `C:\Program Files\Java\jdk1.6.0_07\jre\bin\client\jvm.dll`).

3. Modify the Tomcat startup parameters to set the following Java parameters:

   - -Xms128m
   - -Xmx*<75% of physical memory>*
   - -XX:MaxPermSize=128M

   Set the Java parameter -Xmx to 75% of physical memory. For example, on a machine with 1 gigabyte of RAM, set -Xmx768m. On a machine with 2 GB of RAM, set -Xmx1793m.

   Do not use the -Xss option, which sets the Java thread stack size.

   a. If you are running Tomcat on a UNIX or Linux system, then set the Java parameters as follows:

      ```
      set JAVA_OPTS=-Xms128m -Xmx<75% of physical memory>
      -XX:MaxPermSize=128M
      ```

      Note: While you can also use the **set** command to set the Java parameters on Windows, it is best to define them in the JAVA_OPTS or CATALINA_OPTS environment variables.

   b. If you are running Tomcat as a service, set the Java parameters as part of the Tomcat service configuration. Open `Start -> All programs -> Apache Tomcat 6.0 -> Configure Tomcat`, then click the **Java** tab. The fields map as follows:

   - Initial memory pool: -Xms
   - Maximum memory pool: -Xmx

4. On Windows systems with Tomcat installed as a service, you must set the login information through the service. To ensure that the Visual Modeler works correctly and locates the correct `prefs.xml` file, set the login to the user that owns `prefs.xml`.

   For example, if the `prefs.xml` file is located in:

   ```
   C:\Documents and Settings\adminuser.DOMAIN\cmgt\debs\conf
   ```

   Then the Tomcat login information should be similar to the following:

   - Login: DOMAIN\adminuser
   - Password: adminuser’s password

Notes on Using Apache Tomcat

Apache Tomcat does not automatically re-compile JSP pages that are included in other JSP pages; if you make a change to an included JSP page, remove the corresponding compiled servlet classes from the `container_home/work/` directory to force the JSP page to be re-compiled.

If you use the shutdown command to stop Tomcat gracefully, then persistent session information is saved to a file:

- `container_home/work/Standalone/localhost/Sterling/SESSIONS.ser` on Tomcat 6

When you restart the servlet container, the servlet container will attempt to reload this session data and throw exceptions. You should remove this file before re-starting the servlet container.

You can force Tomcat to not save session information by setting the saveOnRestart attribute of the Context element to “false”. To do this within the SDK, modify the `tomcat-context.xml` file to the following:

```
<Context path="/@app.name@" docBase="@project.base@" >
<Manager className="org.apache.catalina.session.PersistentManager"
   debug="0"
   saveOnRestart="false"
   maxActiveSessions="-1"
   minIdleSwap="-1"
   maxIdleSwap="-1"
   maxIdleBackup="-1">
   <Store className="org.apache.catalina.session.FileStore"/>
   </Manager>
</Context>
```

If you run the fastdeploy target, then this XML file is used to declare the Web application in the `container_home/webapps/` directory.

Certain class files in JAR files are not loaded from `container_home/webapps/Sterling/WEB-INF/lib/`. If you place the JAR files in `container_home/lib/` or `container_home/common/lib/`, then they will be loaded, but note that this may affect the running of other Web applications in the same servlet container.

Continue with the steps described in "Database Server Steps".

Deploying the Visual Modeler on IBM WebSphere

Follow these steps to deploy the Visual Modeler into your installation of IBM® WebSphere Application Server. These steps are written to deploy into WebSphere. In this section, we refer to ***WAS_HOME***: it is the IBM terminology for the *container_home* directory. On a UNIX system, its default location is `/usr/WebSphere/AppServer/`. In the WebSphere Administrator's Console, the node on which the servlet container is installed is displayed as the machine name: we refer to this as <*server*>.

Note: Start and stop the WebSphere application servers using the administration console (Integrated Solutions Console) or choose **Start** and **Stop** from the Windows Start menu's **IBM WebSphere -> Application Server -> Profiles -> <server>** menu. Stopping the server manually can corrupt the configuration data.
Do not use the "_" character in WebSphere server names: WebSphere regards URLs with this character as invalid.

1. Log in to administration console at:

   `https://<server>:9043/ibm/console/logon.jsp`

   using the WebSphere administrator account. You can also start the administration console from the Windows Start menu.
2. Browse to **Security->Secure administration, applications, and infrastructure**, and uncheck the following check boxes:
   - **Enable application security** to disable WebSphere security
   - **Java 2 Security** to disable Java policy security
3. Click **Apply**.
4. Browse to **Applications -> Install New Application**.
5. In the **Preparing for the application installation** panel, check the **Local file system** radio button, and click **Browse** to browse to the location of the WAR file.
6. Enter the context name (for example, "Sterling") for your Web application in the **Context root** text field, then click **Next**.
7. In the **Install New Application** panel, **Step 1 Select installation options**, set the value of Application Name to your preferred one, for example "Sterling", then click **Next**.
8. In **Step 4: Summary**, click **Finish**.
9. When WebSphere completes installing the Visual Modeler Web application, click the **Save directly to the master configuration** link.

   The Install New Application page displays when deployment completes.
10. Click the **Enterprise Applications** link. On the Enterprise Applications page, select the application, then click **Start**.
11. Verify that the Visual Modeler is successfully deployed by pointing your browser to:

    `http://<server>:9080/Sterling/en/US/enterpriseMgr/matrix`

Solaris Optional Step

On Solaris installations of WebSphere, the following may help with JSP page compilation problems.

1. Add a JVM setting as follows:
   a. Using the Integrated Solutions Console, navigate to:

      Servers -> Application Servers -> <*server*> ->Java and Process Management ->Process Definition -> Java Virtual Machine.

   b. In the Maximum Heap Size, enter 128.

   c. Click **Apply,** then click **Save**.
2. Restart the WebSphere Application Server.

Continue with the steps described in "Database Server Steps".

Pre-Compiling JSP Pages

As an optional step, you can pre-compile the JSP pages in the Visual Modeler as follows:

1. At the command line, navigate to `WAS_HOME/bin/`.
2. Enter:

   ```
   ./JspBatchCompiler.sh -enterpriseApp Sterling -webModule "Sterling Product Suite" -verbose true [-nameServerPort <port>]
   ```

   You only need to specify the nameServerPort if your servlet container is listening on a non-standard port.

Deploying the Visual Modeler on WebLogic

This topic describes how to deploy the Visual Modeler application on WebLogic.

Deployment Considerations For WebLogic

WebLogic requires additional JVM settings to allow credit card authorization to CyberSource. To ensure that credit card authorization to CyberSource works properly, start the Weblogic instance with the following JVM setting:

```
-Dweblogic.security.SSL.allowSmallRSAExponent=true
```

If you run WebLogic with proxy enabled, disable the SSL hostname verification as follows:

```
-Dweblogic.security.SSL.ignoreHostnameVerification=true
```

For more information or any version specific instructions please refer to WebLogic documentation.

Deploying the Visual Modeler on WebLogic

Deployment of the Visual Modeler into WebLogic must run as an “expanded” Web application (as opposed to as a WAR file), these instructions ensure that the Web application WAR file is expanded as part of the deployment process.

1. In your WebLogic installation, identify the *`domain_home`* directory that you plan to use for your deployment of the Visual Modeler. The default location is *`container_home`*`/user_projects/domains/mydomain/`.
2. In the *`container_home`*`/user_projects/domains/mydomain/` directory, create a directory in which to deploy your applications called **applications**. In the **applications** directory, create the directory in which to deploy your Visual Modeler application, **Sterling**.
3. Expand the `Sterling.war` file into the *`container_home`*`/user_projects/domains/mydomain/applications/Sterling` directory using a tool such as WinZip. This process can take a few minutes. Verify that the Sterling directory structure is in place as you expand the `Sterling.war` file.
4. Log in to the WebLogic server as the user used to install WebLogic and start the Oracle WebLogic Administration Console.
5. Install the Visual Modeler as a WebLogic application:
    a. In the Domain Configurations section, click **Deployments**, then click the **Lock & Edit** button in the left-hand panel.
    b. In Deployments, click the **Install** button. Using the Install Applications Assistant, locate the *`container_home`*`/user_projects/domains/mydomain/applications` directory, select **Sterling**, and click **Next**.
    c. Use the Install Applications Assistant to configure your application, or accept the defaults and click **Finish**.

d. In the left-hand panel, click the **Activate Changes** button. The application appears in the **Deployments** list with a State of Prepared.

6. Click **Start** to start the application and choose **Servicing All Requests**. The application's State will update to **Start/Running**.
7. Your application is now installed. Click the **Release Configuration** button.
8. Verify that you can log into the Visual Modeler by pointing your browser to the standard URL:

   `http://<server>:<port>/Sterling/en/US/enterpriseMgr/matrix`

   For WebLogic servers, the default port number is 7001.
9. If you run WebLogic with proxy enabled, disable the SSL hostname verification as follows:

   `-Dweblogic.security.SSL.ignoreHostnameVerification=true`

Pre-Compiling JSP Pages

As an optional step, we suggest that you pre-compile the JSP pages before going live to improve performance. You can follow the instructions provided by the *WebLogic JSP Reference* (consult the document currently at this URL: http://edocs.bea.com/wls/docs81/jsp/reference.html) to pre-compile JSP pages.

WebLogic cleans up the directories of compiled JSP pages when the server is stopped and restarted. It is possible to use the `weblogic.xml` file to ensure that compiled JSP pages are preserved by specifying that the keepgenerated parameter is set to true, and specifying a working directory as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
   <!DOCTYPE weblogic-web-app
   PUBLIC "-//BEA Systems, Inc.//DTD Web Application 7.0//EN"
   "http://www.bea.com/servers/wls700/dtd/weblogic700-web-jar.dtd" >
<weblogic-web-app>
   <jsp-descriptor>
      <jsp-param>
         <param-name>
            keepgenerated
         </param-name>
         <param-value>
            true
         </param-value>
      </jsp-param>
      <jsp-param>
         <param-name>
            workingDir
         </param-name>
         <param-value>
            Comergent_jsp
         </param-value>
      </jsp-param>
   </jsp-descriptor>
</weblogic-web-app>
```

XML Parsing

If an error message displays, review "Troubleshooting and Backing Up the Visual Modeler". If you suspect problems with the XML parser settings, then you can set up an XML Registry for the WebLogic server as

follows. Make the following changes to the *container_home*/config/mydomain/config.xml configuration file.

1. Add the following:

```
<XMLRegistry Name="Comergent XML Registry"
   DocumentBuilderFactory=
         "org.apache.xerces.jaxp.DocumentBuilderFactoryImpl"
   SAXParserFactory=
         "org.apache.xerces.jaxp.SAXParserFactoryImpl"
   TransformerFactory=
         "org.apache.xalan.processor.TransformerFactoryImpl" />
```

2. Add the following attribute to the Server element: XMLRegistry="Comergent XML Registry". For example:

```
<Server ListenPort="7001" Name="server" NativeIOEnabled="true"
   StdoutDebugEnabled="true" StdoutSeverityLevel="64"
   TransactionLogFilePrefix="config/ICC/logs/"
   XMLRegistry="Comergent XML Registry">
```

Continue with the steps described in "Database Server Steps".

Installing the Reference Visual Modeler

This section describes the steps to install Visual Modeler without using the SDK. This provides you with a relatively quick verification of the basic deployment of our reference Web application. Before you start the process of customizing the Visual Modeler, install the Visual Modeler into the SDK and work in that environment to create your customized deployment.

Note: To provide support for locales other than U.S. English, you must install a language localization pack using the SDK.

These instructions use three logically distinct machines: the developer machine, the servlet container machine, and the database server machine. Depending on your situation, these machines may actually all be the same physical machine or different ones: we identify what steps are performed on which logical machine.

Make sure that you have the Java Development Kit (JDK) 6 installed on all three machines. The servlet container machine and the database server machine should have the database client tools installed to connect to the database server. You will need to know the database connection information required to connect from the servlet container machine to the database server.

1. Identify the location on your development machine in which you will unpack the installation files: we refer to this as *`cmgt_home`*.
2. Copy the release JAR file into *`cmgt_home`*.
3. Unjar the release JAR file by navigating to *`cmgt_home`* at the command line, and executing:

   `jar -xvf <release JAR file name>`

   This unpacks the release JAR file and creates several sub-directories under *`cmgt_home`*.
4. You must now set the values of system properties that the Web application will need, such as the location of the database. At the command line, execute:

   `java -jar install/cmgt-preferences-tools.jar`

   If you are running on Linux and see an error message, then you may have to install the `xorg-x11-deprecated-libs.rpm` RPM package appropriate for your Linux version.

   The initial settings dialog box displays.
5. Click the `Open dir` or `.war` file button to navigate to the release WAR file, then select the file named `Sterling-Aries-def-RC-1.war` (or something very similar to this). The release WAR file name appears in the **File Name** field. Click **Open**. The Preferences Store Open File dialog box should open up in the correct directory and you should be able to leave the value of the Preferences store with its current value.
6. Click **Next**. The main Preferences Viewer window displays.
7. Using this window, set the values for the following properties as follows:
   a. Navigate to the DataServices.DataSource.ENTERPRISE.ConnectString property. You should see the lower property panel display the name of the property, its current value, and where the value is stored:
   b. In the Tree View, right-click the Connect String property and select **Modify Value**.

c. In the Modify Value dialog box, enter the correct value for the connection string property: this is the URL used by the data services layer to connect to the Knowledgebase database server. The form of this URL depends on the database type as follows:

- For Oracle: "`jdbc:oracle:thin:@<machine>:1521:<sid>`"
- For SQL Server: "`jdbc:sqlserver://<sqldb_name>`; DatabaseName=<*dbname*>"
- For DB2: "`jdbc:db2://<machine>:50000/<instance>`"

d. Repeat these steps for the following properties:

- DataServices.DataSource.ENTERPRISE.DataService:
  - For Oracle: "JdbcService"
  - For SQL Server: "JdbcService"
  - For DB2: "JdbcService"
- DataServices.DataSource.ENTERPRISE.SrvcSubType:
  - For Oracle: "ORACLE"
  - For SQL Server: "MS"
  - For DB2: "DB2"
- DataServices.DataSource.ENTERPRISE.UserId
- DataServices.DataSource.ENTERPRISE.Password
- DataServices.General.JdbcDriver1:
  - For Oracle: "oracle.jdbc.driver.OracleDriver"
  - For SQL Server: "com.microsoft.sqlserver.jdbc.SQLServerDriver"
  - For DB2: "com.ibm.db2.jcc.DB2Driver"
- DataServices.General.DsKeyGenerators: set to the database-specific value:
  - For Oracle: "OracleKeyGenerators.xml"
  - For SQL Server: "MSSQLJDBCKeyGenerators.xml"
  - For DB2: "DB2KeyGenerators.xml"
- You can set any other properties that you wish using the Preferences Editor, but these should be enough to get your reference deployment up and running.

8. When you have finished setting property values, click **File -> Save**.
9. Click **File -> Exit**.

   The new property values will be saved to the file: `cmgt/debs/conf/prefs.xml` in your home directory, referred to as ***user_home*** (for example: `C:\Documents and Settings\username\` or `/export/home/username/`).

10. If the database server is inaccessible from your current network location, then transfer the appropriate sql-data file (depending on your DB_TYPE) to a temporary location, *`cmgt_data_home`*, on a machine that can access the database server machine:

- *cmgt_home/sql/DB2sql-data-7.1-def-RC-1.jar*

or

- *cmgt_home*/sql/MSSQLJDBCsql-data-<release JAR file name>

or

- *cmgt_home*/sql/Oraclesql-data-<release JAR file name>

11. On this machine, at the command line navigate to *cmgt_data_home*.
12. Unpack the JAR file by executing:

    jar -xvf DB_TYPEsql-data-<*release JAR file name*>
13. Set up your database users, privileges, and, if you are using an Oracle database server, the Oracle database link.
14. If you are using an Oracle database server, edit the following files:

- *cmgt_data_home*/WEB-INF/sql/Oracle/setup/oracle_indexes.sql file: change the value of @ORACLE_INDEX_TABLESPACE@ name, if necessary, to TABLESPACE <*tablespace_name*>. The TABLESPACE name specifies the location of your Oracle database server's index files.
- *cmgt_data_home*/WEB-INF/sql/Oracle/setup/Oracle_privileges.sql file: replace @ORACLE_USERNAME@ with the name of the transactional database user name.

15. If you are using a SQL Server 2005 database server, edit the following files:

- *cmgt_data_home*/WEB-INF\sql\MSSql\setup/mssql_create_replication.sql file: replace the following variables with appropriate values. See "SQL Server 2005 Setup" for details.
  - @MSSQLJDBC_SERVERNAME@: This is the actual name of the server machine, not the network name
  - @MSSQLJDBC_SA_PASSWORD@
  - @MSSQLJDBC_DISTRIBUTOR_DATABASE@
  - @MSSQLJDBC_DISTRIBUTOR_DATA_FOLDER@
  - @MSSQLJDBC_DATABASE@
  - @MSSQLJDBC_START_DATE@
- *cmgt_data_home*/WEB-INF\sql\MSSql\setup/mssql_drop_replication.sql file: replace the following variables with appropriate values. See "SQL Server 2005 Setup" for details.
  - @MSSQLJDBC_SA_PASSWORD@
  - @MSSQLJDBC_DISTRIBUTOR_DATABASE@
  - @MSSQLJDBC_DISTRIBUTOR_DATA_FOLDER@
  - @MSSQLJDBC_DATABASE@

16. Edit the following batch scripts to add your connection information. The scripts reside in *cmgt_data_home*.

- For DB2 database servers:
  - **DB2CreateSchema.bat or DB2CreateSchema.sh**
- For Oracle database servers:

- **OracleCreateSchema.bat** or **OracleCreateSchema.sh**

- For SQL Server 2005 database servers:
  - **MSSQLJDBCCreateSchema.bat**

17. Run the following scripts:

- For DB2

  DB2CreateSchema.bat or DB2CreateSchema.sh

- For Oracle:

  OracleCreateSchema.bat or **OracleCreateSchema.sh**

- For SQL Server 2005:

  MSSQLJDBCCreateSchema.bat

18. Copy the following files to a temporary location, *cmgt_app_home*, on the servlet container machine:
    - *cmgt_home*/Sterling-Aries-def-RC-1.war
    - *user_home*/cmgt/debs/conf/prefs.xml
    - *cmgt_home*/data/Sterling-xmldata-<release JAR file name>
    - *cmgt_home*/data/cmgt-xmlloader-tool.jar
    - *cmgt_home*/install/cmgt-cryptography-tool.jar
    - *cmgt_home*/install/cmgt-jspResourcer.jar
    - *cmgt_home*/install/xmlClient-tool.jar
19. On the servlet container machine, copy the *cmgt_app_home*/prefs.xml file to the following directory (which you may have to create): *user_home*/cmgt/debs/conf/prefs.xml. Note that this should be the *user_home* for the user that is used to run the servlet container.
20. If you are running against Oracle, SQL Server, or DB2, then install the appropriate JDBC JAR file into the servlet container so that it can be used by the Visual Modeler web application when deployed. In the Tomcat Application Server, install these JAR files by placing them in the *container_home*/common/endorsed/ directory. In WebLogic application servers, this step is probably unnecessary, since WebLogic servers are deployed with an extensive collection of JDBC clients.
21. Make sure that you are logged in as the user who is running the application server, and at the command line, navigate to *cmgt_app_home*.
22. Unpack the Sterling-xmldata-<release JAR file name> file by executing:

    jar -xvf Sterling-xmldata-<*release JAR file name*>

23. Load the data.
    a. On UNIX, run:

       `loadDBFromXML.sh full <jdbc_jar_file.jar>`

       where <jdbc_jar_file.jar> is the full path to your JDBC driver. If you get a permissions error, then modify the permissions on the script to give yourself execution privileges

       Note: On Linux, if you see errors reporting that a network connection cannot be established to the database server, then check that you do not have Secure Linux enabled. If need be, navigate to `/etc/sysconfig/selinux` and make sure that the following is set:

       `SELINUX=disabled`

    b. For Windows configurations using ODBC to connect to the Knowledgebase database server, run:

       `loadDBFromXML full`

    c. For Windows configurations with JDBC, run:

       `loadDBFromXML full <jdbc_jar_file.jar>`

       where <*jdbc_jar_file.jar*> is the full path to your JDBC driver.

       Note: You can load just the minimal data by specifying “minimal” rather than “full” when you run this command.
24. Check the results by examining the log files in the directory in which you ran the data load command.
25. Rename the *`cmgt_app_home`*`/Sterling-Aries-def-RC-1.war` file to `Sterling.war` and deploy it to the servlet container using the steps described in "Deploying the Visual Modeler Application".
26. Restart the application server.
27. You should now be able to point your browser to the standard Visual Modeler URL and log in as the enterprise administrator user admin/admin.

Database Server Steps

Depending on which database server you use with the Visual Modeler, perform the required steps in this section. See "Managing Database Connections" for information about connection pooling.

Support for Oracle Server

If you plan to use Oracle Server for your database server, then you must run the installOracle target as part of the installation of the Visual Modeler. This ensures that the Oracle JDBC drivers are in the deployed Web application.

If you plan to use the OCI driver to connect from the Visual Modeler machine to the Oracle Server, then you must make sure that the OCI driver is set up correctly. On a UNIX system:

1. Make sure that the OCI `liboci*jdbc.so` file is installed on the Visual Modeler machine. The “*” in the name of the file is the version number of the OCI library.
2. Make sure that the servlet container scripts ensure that the LD_LIBRARY_PATH includes the location of the OCI `liboci*jdbc.so` file and that ORACLE_HOME is set to point to the location of the Oracle client tools.
3. Make sure that the Oracle JAR file matches the version of the OCI library file:

OCI Library	JDBC JAR File
oci804jdbc.so	Oracle.jar
ocijdbc8.so	Classes111.zip

Support for SQL Server

If you are running the Visual Modeler on a Windows system and plan to use SQL Server 2003 for your database server, then you must perform the following steps.

1. Copy the appropriate DLL file from *`debs_home`*`/Sterling/WEB-INF/lib/winnt/` to the `C:\WINNT\system32\` directory. Note that the installMsSql target will do this on a machine running the SDK, but if your deployment machine is different from the SDK machine, then you must do this step manually.
2. If you plan to support data in locales other than en_US, then you must consider how basic searches are performed. If you do not want searches to differentiate between upper and lower cases instances of the same character, then you must provide values for the elements LowerCase and UpperCase of the

Microsoft element of the `DataServices.xml` file, contained in the `WEB_INF\lib\cmgt-dataservices.jar` file.

Set LowerCase to "LOWER(" and UpperCase to "UPPER("; for example:

```
<UpperCase controlType="text" runtimeDisplayed="true"
ChangeOnlyAtBootTime="true" visible="true" boxsize="45"
displayQuestion="UpperCase SQL Function" defaultChoice="" help="Enter the
SQL function that converts strings to uppercase for the selected
database.">UPPER(</UpperCase>
```

Note: The use of the UPPER and LOWER functions in searches means that indexes on the tables are not used and this can result in reduced performance.

3. If more than one deployment of the Visual Modeler is accessing the same Knowledgebase on SQL Server, then you must set a two-digit server ID for each deployment.
   a. If the machines are not clustered, then set the ServerId element in the `prefs.xml` file so that each has a unique integer value: 01, 02, and so on.
   b. If the machines are clustered, then you must modify the servlet container command or script that starts the servlet container on each machine so that a Java system property is set: Comergent.DataServices.General.ServerId. This should be set on each machine so that each has a unique value: 01, 02, and so on.

      For example, in a Tomcat installation, you can modify the batch file to include:

      ```
      set JAVA_OPTS=-DComergent.DataServices.General.ServerId=02
      ```

Support for DB2 Server

If you plan to use DB2 Server for your database server, then you must run the installDB2 and env.setDBType targets as part of the installation of Visual Modeler. This ensures that the DB2 JDBC drivers are in the deployed Web application.

Filtering Static Content

In general, you should use a servlet container in conjunction with a Web server. The Web server can be used to serve other content for your Web site. In addition, the Web server can be used to serve static content from the Visual Modeler. In this way, you can enhance the performance of your Web site.

Setting up Apache to Serve Static Content

If you have Apache running as a Web server in front of your servlet container, then you can make use of Apache's capabilities to serve static content. In this section, we show how to use the expires_module module to mark images so that a client's browser caches images rather than re-requesting them each time a page displays the image. In particular, this approach can be used to prevent image-flicker if a user has their browser settings such that images are re-loaded from the server on every visit to the page.

These instructions assume that the Apache Web server and the Tomcat servlet container are running on different machines.

Follow these steps:

1. Edit the Apache `httpd.conf` configuration file to add or uncomment:

   ```
   LoadModule expires_module modules/mod_expires.so
   ```

2. Add the following expires rules:

   ```
   ExpiresActive On
   ExpiresByType image/gif "access plus 1 day"
   ExpiresByType image/jpg "access plus 1 day"
   ExpiresByType text/css "access plus 1 day"
   ExpiresByType text/js "access plus 1 day"
   ```

   In these lines, you are specifying that the expires module is active, and that by default all the static content served by the Apache Web server should be cached by browsers for one day after accessing it. You can change these settings to meet the needs of your deployment of the Visual Modeler.

3. Restart the Apache Web server

   Note: Under certain circumstances, this may give rise to unwanted behavior. For example, if partner administrators frequently upload partner logos in the form of GIF files, then some storefront users who have the older version of the GIF file already cached will not see the new version of the GIF file until a day passes.

Creating a NSAPI Filter

We provide a small file of C code that can be used to ensure that certain files are served by the Web server rather than by the iPlanet Application Server. It uses the NSAPI.

1. Locate the `ctrans.c` file under *`debs_home/`*.
2. Compile it to a dynamic library.

   For Solaris, the compile command is:

   ```
   gcc -DXP_Unix -DMCC_HTTPD -DNET_SSL -DSOLARIS -D_REENTRANT -Wall -c
   module.c * ld -G module.o -o module.so
   ```

   For Windows, the compile command is:

   ```
   cl -LD -MD -DMCC_HTTPD -DXP_WIN32 -DNET_SSL module.c -Insapi\include /link
   nsapi\examples\libhttpd.lib
   ```

3. Add the module to the NameTrans directive in the Web server's `obj.conf` configuration file as follows:

   a. Find the block where all the Inits are declared. Add line:

      ```
      Init fn="load-modules" shlib="/container_home/local/nsapi/comergent.so"
      funcs="handle_comergent_static"
      ```

      Replace the string `/container_home/` with the appropriate path.

   b. Find the block:

      ```
      <Object name="default">
      ```

   c. Add the line:

      ```
      NameTrans fn="handle_comergent_static"
      ```

      By default, use the following values for the parameters:

      ```
      . prefix /NASApp/Comergent/
      . newPrefix /container_home/ias6/ias/APPS/modules/Comergent/
      . list *.css, *.gif, *.js, *.jpg
      ```

      If you need to override any of the above values, then append them to the "NameTrans" line. For example:

      ```
      NameTrans fn="handle_comergent_static"
      newPrefix="container_home/ias6/ias/TEST/modules/Comergent/"
      ```

Gathering the Database Information

Identify the connection information for the database you are using. This includes:

1. Identify the connection information required to connect from the Visual Modeler machine to the database server.
   - For a DB2 Server, this is the catalog name of the database server. You must create a catalog entry for the target DB2 database on the Visual Modeler machine. You can use the following DB2 commands to create a catalog entry:

     ```
     db2 CATALOG TCPIP NODE <node name> REMOTE <ip address of server> SERVER <port number>
     ```

     ```
     db2 CATALOG DATABASE <catalog name> AT NODE <node name> AUTHENTICATION SERVER
     ```

   - For an Oracle Server, this is the machine name or IP address of the database server, the port at which the database server is listening, and the SID (name of the database instance) of the database server. You must create a TNS alias for the database server on the Visual Modeler machine.
   - For a SQL Server, this is the machine name or IP address of the database server.
2. Establish a database userid on the database server which is used by the Visual Modeler.

   Use this userid to perform all the Visual Modeler-related calls to the database. This userid must have sufficient privileges to create and modify database tables.

   You may use one of your existing database userids. However, we suggest that you set up a database userid that is dedicated to Visual Modeler-related tasks.
3. Check that you can connect to the database server using the userid and connection information:
   - For DB2, you must be able to connect using the catalog entry for the database server.
   - For Oracle, you must be able to connect from the Visual Modeler machine to the database server using SQL*Plus and the TNS alias.
   - For SQL Server, you must be able to connect using OSQL and the JDBC data source name and userid.

Populating the Knowledgebase

Having created the Knowledgebase, you need to load data into it, in order to run the Visual Modeler. Visual Modeler loads data into the Knowledgebase using a set of XML definitions of each data object. Data loading is invoked from the SDK using the loadDB and loadMatrixDB targets: these load the minimal and reference data sets respectively. These targets invoke the XMLLoader scripts as described in "XMLLoader Script".

XML Data Format

The data to be loaded using the XMLLoader scripts must be created in the form of XML elements: one for each data object. The form of the XML elements closely matches the structure of the data object: The name of the top-level element is the name of the data object and each child element corresponds to a data field or child data object of the data object.

The top-level element has these attributes:

- A state attribute: set this value to “INSERTED” when you are creating new data. You can use the value “MODIFIED” if you are modifying an existing data object using the XML data loader.
- A type attribute: set this value to “BusinessObject”.

You can use a list data object to act as a container for a list of data objects to be loaded. You must provide a value for each element that is declared as mandatory in the data object definition.

The XMLLoader script essentially sets a classpath and then invokes an XMLLoader class, passing it parameters for the location of the `Comergent.xml` file, the operation (usually “persist”), the partner name (usually “matrix”), and a list of one or more files of data to be loaded.

The files must be in one of two forms:

- Either a set of XML elements: the root element must be named *DataObject*Data. For example:

```
<PromotionData>
   <!--Record 1 ----------------------------------------------->
   <Promotion state="INSERTED" type="BusinessObject">
         <PromotionKey state="INSERTED">126</PromotionKey>
         <PromoCode state="INSERTED">ID 360</PromoCode>
         <PromotionName state="INSERTED">Packages</PromotionName>
         ...
   </Promotion>
   <!--Record 2 ----------------------------------------------->
   <Promotion state="INSERTED" type="BusinessObject">
         <PromotionKey state="INSERTED">127</PromotionKey>
         <PromoCode state="INSERTED">ID 3837</PromoCode>
         ...
   </Promotion>
   ...
</PromotionData>
```

- Or each file can point to a list of files:

```
WEB-INF/xmldata/ProductCategoryList
WEB-INF/xmldata/ProductList
...
```

By convention, the files that provide lists of other files have the suffix “lst”.

XMLLoader Script

The XML data loading script invokes a Java class that uses the Visual Modeler DataManager to load each object.

The script is called `XMLLoader.bat` (or `XMLLoader.sh` on UNIX systems) and it is located in *`debs_home`*`/Sterling/WEB-INF/scripts/`. The Visual Modeler provides two sets of data that can be loaded: see "Data Sets" for further information.

Encryption

You can encrypt sensitive data fields so that data stored in the Knowledgebase does not store the data in plain text. Use this mechanism for fields such as user passwords and credit card numbers, but any field can be encrypted provided that its corresponding database column can store strings.

Note: You must determine which fields are to be encrypted before loading any data into the Knowledgebase.

When data is encrypted, a special file called `dcmsKey.ser` is created. You must ensure that this file is stored safely. If it is deleted or moved, then the encrypted data cannot be recovered. Note that you cannot

export and re-import data that has encrypted fields. The encrypted fields will be garbled if you attempt to do this.

Defining the Knowledgebase as the Data Source

To run the data loading script, you must configure the Visual Modeler to access the Knowledgebase. You do this using the configuration files `prefs.xml`, and `DataServices.xml`. You must also ensure that the DsKeyGenerators element of the `DataServices.xml` file points to the correct key generator file: `DB2KeyGenerators.xml`, `MsSqlKeyGenerators.xml` or `OracleKeyGenerators.xml`. Note that if you use the SDK to install the Visual Modeler, then the correct values will be set up automatically in these files.

Note: Please read the database server-specific instructions below. Set the logging level to INFO before running the data loading script.

The following sections provide a brief description of the syntax of the `DataSources.xml` file for the supported database servers.

DB2DataSources Syntax

```
<Primary DataService="JdbcService" SubType="DB2"
ConnectionString="jdbc:db2://DB2_MACHINE:DB2JD_PORT/DATABASE"
UserId="DB2_OS_USER" Password="DB2_OS_PASSWORD"/>
```

• DB2_MACHINE is the machine name or IP address of the machine on which DB2 is running.

• DB2JD_PORT is the port at which the JDBC listener is listening.

• DB2_OS_USER and DB2_OS_PASSWORD are the userid used to create the schema.

MsSqlDataSources Syntax

```
<Primary DataService="MsSqlService" SubType="MS"
   ConnectionString="MSSQL_MACHINE"
   UserId="MSSQL_USERNAME" Password="MSSQL_PASSWORD" />
```

- MSSQL_MACHINE is the machine name or IP address of the machine on which SQL Server is running.
- MSSQL_USERNAME and MSSQL_PASSWORD are the username and password used to create the Visual Modeler schema. Note that the default database for this user must be the database in which the schema was created.

OracleDataSources Syntax

You can use either the OCI JDBC driver or the Oracle thin client JDBC driver to connect from the Visual Modeler machine to the Oracle Server. Use:

- For Oracle 8*i*:

```
<Primary DataService="JdbcService" SubType="ORACLE"
   ConnectionString="jdbc:oracle:oci8:@ALIAS"
   UserId="ORACLE_USERNAME" Password="ORACLE_PASSWORD"/>
```

- For Oracle 9*i*:

  ```
  <Primary DataService="JdbcService" SubType="ORACLE"
     ConnectionString="jdbc:oracle:oci:@ALIAS"
     UserId="ORACLE_USERNAME" Password="ORACLE_PASSWORD"/>
  ```

  or

  ```
  <Primary DataService="JdbcService" SubType="ORACLE"
  ConnectionString="jdbc:oracle:thin:@ORACLE_MACHINE:ORACLE_PORT:SID"
    UserId="ORACLE_USERNAME" Password="ORACLE_USERNAME" />
  ```

- ALIAS is the TNSNAMES alias set up on the Visual Modeler machine.
- ORACLE_USERNAME and ORACLE_PASSWORD are the userid used to create the Visual Modeler schema.
- ORACLE_MACHINE is the machine name or IP address of the machine on which the Oracle Server is running.
- ORACLE_PORT is the port number at which the Oracle Server listener is listening for connections.
- SID is the Oracle SID of the database.

Internationalization and Support for Locales

This section describes how to create locales and update data objects using the XMLLoader.

Creating Locales

You must edit the `LocaleDataList` file (located in *`debs_home`*`/Sterling/WEB-INF/xmldata/`)to specify the locales that you want the Knowledgebase to support. Each installation of the Visual Modeler can support one or more locales.

Make sure that the server locale (defined by the defaultSystemLocale element of the `Internationalization.xml` file) is included in the list of locales defined in `LocaleDataList`.

Updating Data using XMLLoader

When you load locale-specific data into the Knowledgebase, you can make use of the XMLLoader's ability to modify data objects. In each data object element and child elements, set the state attribute to "Modified". This will update business objects rather than inserting a new business object.

This feature is particularly useful when you are adding locale-specific information to an existing implementation of the Visual Modeler. For example, the following data object can be used to update a product category business object:

```
<ProductCategory state="MODIFIED">
   <ProductCategoryKey state="MODIFIED">1002</ProductCategoryKey>
   <ParentCategoryKey state="MODIFIED">-1</ParentCategoryKey>
   <SequenceId state="MODIFIED">5</SequenceId>
   <ResourceKey state="MODIFIED">3</ResourceKey>
   <StartDate state="MODIFIED">2000-10-06 17:20:28.0</StartDate>
   <EndDate state="MODIFIED">2100-10-06 17:20:28.0</EndDate>
   <OwnedBy state="MODIFIED">1</OwnedBy>
   <AccessKey state="MODIFIED">2006</AccessKey>
```

```
   <ProductCategoryLocale state="MODIFIED">
      <Locale state="MODIFIED">de_DE</Locale>
      <Name state="MODIFIED">Software</Name>
      <Description state="MODIFIED">
         Alle Anwendungspakete, die auf unserer Site vorhanden sind, werden auf
allen unsere Qualitätscomputersysteme geprüft und bestätigt.
      </Description>
   </ProductCategoryLocale>
</ProductCategory>
```

Database Server-Specific Steps

When running the data loading scripts, the steps vary a little from one database server to another. This section covers the supported database servers.

DB2 Server Steps

1. When running the XMLLoader.bat script, the DB2 server must be running the DB2 JDBC listener. Start this listener by entering at the command line:

   ```
   >db2jd <port number>
   ```

2. Edit the DataServices.xml file to specify the DB2DataSources.xml file and DB2KeyGenerators.xml file. You must also make sure that both JdbcDriver1 and JdbcDriver2 elements are not commented out.
3. In DB2DataSources.xml, make sure that the connection information sets the port number used in Step 1. Make sure that the same UserId and Password as were used to create the schema.
4. For performance reasons, after you have run the data loading scripts, you must perform the following steps to ensure that the indexes are used correctly:
   a. Generate a list of the schema tables:

      ```
      >db2 list tables
      ```

   b. Create and run a script that invokes the REORG command on each table:

      ```
      REORG table <schema_name>.<table_name>
      ```

   c. Create and run a script that invokes the RUNSTATS command on each table:

      ```
      RUNSTATS on table <schema_name>.<table_name> with distribution and
      indexes all
      ```

SQL Server Steps

1. If you are using MS SQL Server as your database server, then make sure that you have copied the `MsSqlJNI.dll` file to the `Winnt\system32\` directory on the Visual Modeler machine.
2. Edit the `DataServices.xml` file so that all the JdbcDriver elements are commented out. Use the `<!--` and `-->` tags to comment out each element.
3. In `MsSqlDataSources.xml`, make sure that the connection information sets the same UserId and Password as were used to create the schema.

Oracle Steps

1. Edit the `DataServices.xml` file to specify the `OracleDataSources.xml` file and `OracleKeyGenerators.xml` file.
2. Make sure that the JdbcDriver1element takes the value of the name of the Oracle JDBC driver.

Data Sets

The Visual Modeler provides two sets of XML data objects:

- Reference implementation: this set populates the Knowledgebase with the complete reference implementation (Matrix Solutions) data set. You use this set if you want to deploy our reference implementation in order to familiarize yourself with the Visual Modeler.
- Minimal implementation: this set populates the Knowledgebase with the minimal data required to get the Visual Modeler up and running. You use this set when you want to deploy your production system using your data. See "Email Addresses" for information about email addresses set in the minimal data set.

  Note: You must always install the minimal data set: you can optionally layer the reference data on top. Use the SDK loadDB target to load the minimal data only; use the loadMatrixDB target to load both.

To Edit and Run the XML Data Loading Script

1. Configure the `Comergent.xml`, `DataServices.xml`, and appropriate `DataSources.xml` configuration files to point to the database server to be used for the Knowledgebase.
   a. Make sure that the DataServices element of the **Comergent.xml** file points to the correct location of the `DataServices.xml` file.
   b. Make sure that the DsDataSources element of the `DataServices.xml` file points to the correct location of your `DataSources.xml` file.
   c. Make sure that the appropriate connection information has been entered in the `DataSources.xml` file.
2. If your environment does not have a JAVA_HOME environment variable set, then edit the `XMLLoader.bat` to set the JAVA_HOME environment variable to point to your installation of the JDK. For example:

   ```
   >set JAVA_HOME=C:\JDK1.2.2
   ```

   If you are using either DB2 or Oracle as the database server, then make sure that the classpath is set to include the location of the appropriate JDBC driver class. Typically, you must ensure that the XMLLoader script includes a line of the form:

   ```
   CP=%CP%;%DH%/lib/oracle816_jdbc2.jar
   ```

   or

   ```
   CP=%CP%;%DH%/lib/db2runtime71.jar
   ```

   If you have deployed the Visual Modeler in WebSphere, then you must edit the classpath setting portion of the script to allow for the fact that the class and library files are in the *debs_home*`/Sterling/servlets/` directory.
3. Save the edited file to *debs_home*`/Sterling/`.
4. Run the XML data loading script from *debs_home*`/Sterling/`.

- The syntax to load the reference data is:

  ```
  >XMLLoader persist
  ```

- The syntax to load the minimal data is:

  ```
  >XMLLoader persist minimal
  ```

Removing Locales

Out of the box, the schema creation scripts and the minimal and reference data sets create data for several locales. Before going live with your implementation, you should remove from the Knowledgebase any locales not supported by your implementation.

To remove a locale, you must remove references to it from the following tables:

- CMGT_ANALYZER_TEXT: for example

  ```
  DELETE FROM CMGT_ANALYZER_TEXT WHERE LOCALE = 'de_DE';
  ```

- CMGT_CURRENCIES: for example

  ```
  DELETE FROM CMGT_CURRENCIES WHERE LOCALE = 'de_DE';
  ```

- CMGT_LOCALE: for example

  ```
  DELETE FROM CMGT_LOCALE WHERE LOCALE_NAME = 'de_DE';
  ```

- CMGT_LOCALE_CURRENCY: for example

  ```
  DELETE FROM CMGT_LOCALE_CURRENCY WHERE LOCALE_NAME = 'de_DE';
  ```

- CMGT_LOCALE_NAMES: for example

  ```
  DELETE FROM CMGT_LOCALE_NAMES WHERE LOCALE_NAME = 'de_DE';
  DELETE FROM CMGT_LOCALE_NAMES WHERE EFFECTIVE_LOCALE = 'de_DE';
  ```

- CMGT_LOOKUPS: for example

  ```
  DELETE FROM CMGT_LOOKUPS WHERE LOCALE = 'de_DE';
  ```

- CMGT_<*OBJECT*>_LOCALE: there are multiple tables that store locale-specific strings for data objects such as products, features, and so on. You must remove the references to the deleted locale from each such table. For example

  ```
  DELETE FROM CMGT__<OBJECT>_LOCALE WHERE LOCALE_NAME = 'de_DE';
  ```

Logging in to the Visual Modeler

Point your browser to the standard login page. The standard URL to access this page is:

`http://<server>:<port>/Sterling/enterpriseMgr/matrix`

Irrespective of whether you have generated the reference data set or minimal data set, you can log in as the enterprise administrator whose username and password are “admin” and “admin”.

You can now administer the Visual Modeler through the standard browser interface.

Note: Before going live with your implementation of the Visual Modeler, you *must* change the passwords of the admin and ERPAdmin users. Failure to so presents a security breach.
Do not use the ERPAdmin user for administration tasks. It is intended only for integration with an ERP system. You should not use this user to log in to the system through the Web.

When you have successfully completed your installation, proceed to the next topic. Otherwise, refer to "Troubleshooting and Backing Up the Visual Modeler" to troubleshoot your installation.

Troubleshooting and Backing Up the Visual Modeler

This section discusses troubleshooting and backing up the Visual Modeler.

Troubleshooting

Testing with the Administration URL

You can make sure that the various parts of the installation are functioning by pointing your browser to the URL used to access the administration pages:

`http://<server>:<port>/Sterling/en/US/enterpriseMgr/matrix`

Email Server

You must make sure that the SMTP Mail Server used to send email from the Visual Modeler is up and running. Make sure that you can ping the SMTP Mail Server from the Visual Modeler machine using the machine name specified in the SMTPHost element of the `Comergent.xml` file. Storefront administrators can configure the SMTP host by setting the SMTP Host Machine system property to the appropriate value. To set the SMTP Host Machine system property, navigate from the System Administration panel of the home page to System Services, then the Commerce Manager category, then the SMTP category, and enter the appropriate value in the SMTP Host Machine field.

Certain UTF-8 characters may not display well in the subject lines of email sent from the Visual Modeler to users. This is due to email clients that are not configured to display UTF-8 characters correctly. If the problem persists, review the characters being used in the subject lines of the email and provide information to users about suitable email clients.

General Troubleshooting Tips

This section includes general diagnosis approaches that can help to quickly pinpoint the source of your problem.

Tomcat Server

The following considerations apply to running Apache Tomcat:

- SESSIONS.ser: when Tomcat is shut down, the server saves the current session information to a file called *`container_home`*`/work/Standalone/localhost/Sterling/SESSIONS.ser`**.** You should delete copies of this file before restarting Tomcat.
- By default, Tomcat does not recompile JSP pages if it determines that its compiled version has a timestamp newer than the corresponding JSP page. If you see errors relating to MethodNotFound exceptions, then the likely cause is an old compiled page. You can solve this problem by deleting the *`container_home`*`/work/Standalone/localhost/Sterling/` directory to force re-compilation of all the JSP pages.

Common Problems

This section covers problems that commonly occur during startup and runtime. You can use the messageTypeValidate element to validate all the message types as the system starts. The element is set to TRUE by default. You should set this element to FALSE once the system has passed its acceptance tests.

Errors at Startup Time

When the Visual Modeler is started by the servlet container, it logs its progress through initialization. Look for these errors in the console window or event log.

Error	Cause and Solution
`InputStream: 24, 384: "</Comergent>" expected.`	A syntax error in one of the configuration files has caused the DataManager to fail to initialize. You must correct the syntax error. The InputStream line provides the exact location of the error.
`java.io.FileNotFoundException: C:\jakarta-tomcat\webapps\Sterling\WEB-INF\properties\Comergent.xml (The system cannot find the file specified)`	The main `Comergent.xml` file is not in the correct location as specified by the propertiesFile element of the `web.xml` file.
`Env/main:W1:DATASERVICES Primary Connection Failed: Io exception: Connection refused(DESCRIPTION =(TMP=)(VSNNUM=135290880)(ERR=12505)(ERROR_STACK=(ERROR=(CODE=12505)(EMFI=4))))`	The server has failed to connect to the database server. Perform the following checks: Ping the database server machine; there may be a network failure. Enter the IP address of the machine to see whether the host name can be resolved. If you can ping the database server machine, then check that the database connection information that you have entered in the `DataSources.xml` file is correct. If possible, use an alternate database connection method (such as SQLPLUS or SQL Server's Enterprise Manager). If this fails, then either the database is down or you have incorrect connection information.
`Primary Connection Failed: No suitable driver`	The server has failed to find a valid Driver class in its classpath. Check that any JDBC driver specified in the `DataServices.xml` file lies in one of the classpath directories or archive files. In particular, check that an appropriate JDBC driver has been specified to connect to the database server: For Oracle Servers, you may use oracle.jdbc.odbc.OracleDriver.
`Cannot find file=web-inf/HostedPartner.xml`	The initialization servlet has failed to find one of the properties files referred to in the main `Comergent.xml` file. Check the names and paths to the properties files. In a UNIX installation, check for case-sensitive file names.

Error	Cause and Solution
`Failed to create a NameKeyTable. com.comergent.dcms.util.ICCException: [CMGT_E_SCHEMA_KEY_GEN_NOT_FOUND] error: " Schema Error - DataObject: Promotion ExternalObject: PromotionControl specifies KeyGenerator: ControlKey which does not exist."`	A problem lies in your definition of the XML schema. On startup, the Visual Modeler reads the schema files and attempts to load the schema as an internal data structure. Exceptions are most commonly thrown when the definition of an element is omitted from the schema. In this example, the Visual Modeler has read the `DsRecipes.xml` file, and attempted to load the Promotion DataObject. This DataObject includes in its definition file, `Promotion.xml`, a reference to a KeyGenerator element called "ControlKey". Inspection of the DsKeyGenerators file shows that no KeyGenerator element called ControlKey is declared.
`java.net.BindException: Address in use`	A process is already bound to one of the ports that the Visual Modeler is attempting to use. You must either stop the existing process that is using the port or use a different port.
`com.comergent.dcm.util.ICCException: [CMGT_E_UNKNOWN_ELEMENT] error: "Element: Comment of DataObject: Comment is not in the DCMS schema"`	An error has occurred while the DataManager initializes the schema. Check the definition of business and data objects. Check that a header DsElement has been declared for the data object and that DsElements are declared for each data element.
`Comergent Init Servlet: DataManager NOT initialized com.comergent.dcm.util.ICCException: [CMGT_E_SCHEMA_KEY_GEN_NOT_FOUND] error: "Schema Error - DataObject: OrderAddress ExternalObject: OrderAddress specifies KeyGenerator: OrderAddressKey which does not exist."`	You have declared a KeyGenerator in the data object definition, but it is not defined in the `KeyGenerators.xml` file. Make sure that you have modified the correct `KeyGenerators.xml` file: this is usually `OracleKeyGenerators.xml` or `MsSqlKeyGenerators.xml`. Also make sure that your `DataServices.xml` file points to the correct `KeyGenerators.xml` file.

`2002.10.22 13:33:03:459 Env/Thread-2:ER:DATASERVICES JDBCService.restore Error: ORA-00600: internal error code, arguments: [ttcgcshnd-1], [0], [], [], [], [], [], []`
`java.sql.SQLException: ORA-00600: internal error code, arguments: [ttcgcshnd-1], [0], [], [], [], [], [], []`
`at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:168)`
`at oracle.jdbc.ttc7.TTIoer.processError(TTIoer.java:208)`
`...`
	There is a mismatch between the Oracle database version (usually 8i or 9i) and the JDBC driver that is being used to connect from the Visual Modeler. Check the classpath that the servlet container is using and remove any references to JDBC driver JAR files that come before the `oraclejdbc.jar` file in *`debs_home`*`/Sterling/WEB-INF/`.

Error	Cause and Solution
`Env/main:ER:MSGT Illegal Unresolved Reference to a MessageType: contentFrame`	A MessageTypeRef element references a message type definition that does not exist.
`Env/main:ER:MSGT com.comergent.api.dcm.messageType.MessageTypeInstantiationException: Failed instantiating or locating com.comergent.apps.partnerMkt.blc.MissingBLC in MessageType GenericLoginDisplay`	A message type failed validation: typically, this means that one of its elements is missing such as a missing BLC, controller class, or JSP page.

`2003.08.05 15:35:28:359 Env/Thread-6:ER:CORE java.lang.NoClassDefFoundError`
`java.lang.NoClassDefFoundError`
`at com.comergent.dcm.authentication.UserPasswordCredentials.verify(UserPasswordCredentials.java:38)`
`at com.comergent.dcm.authentication.LoginController.execute(LoginController.java:59)`
	On startup, the Visual Modeler has tried to re-instantiate a session stored when the servlet container was last stopped. Before starting the servlet container, make sure that you have deleted any stored sessions.
	For example, in Tomcat 4.1, check the `container_home/work/Standalone/localhost/Sterling/` directory, and delete any files called `SESSIONS.ser`.

Errors at Runtime

Some errors are listed here that have occurred infrequently in running instances of the Visual Modeler.

Error	Cause and Solution
Assertion failed: 1 == pConnectionObject->fCallCheck, file q:\SPHINX\NETLIBS\nt\ssock\src\ntssockc.c, line 1039	This error has been observed when the Visual Modeler is run with SQL Server. Make sure that you have applied the latest Windows and SQL Server Service Packs to the machine on which SQL Server is running, and make sure that the client SQL Server software installed on the Visual Modeler machine matches your version of SQL Server.

Error	Cause and Solution
On Solaris, the servlet container cannot find a certain servlet or URL.	First make sure that you did not make a typo. If you are certain that there was no mistake, then do the following: 1. Run the following command on `web.xml`: java com.comergent.dcm.util.CheckWebXML web.xml > newWeb.xml 2. Edit the file `newWeb.xml`. Look for the following string `<!-- (8192) XXX BOUNDARY BREAK -->` The start of the comment <!-- is the start of a 8192 boundary break. If it falls within a value for an XML node, then that node will get truncated. A work around is to pad the `web.xml` file such that the boundary break will fall inside a comment. For more information, see the comments at the start of file CheckWebXML.java.
You see parser errors such as: java.lang.NoSuchMethodError at org.apache.xpath.DOM2Helper.getNamespaceOfNode (DOM2Helper.java:348) at org.apache.xml.utils.TreeWalker.startNode (TreeWalker.java:281) at org.apache.xml.utils.TreeWalker.traverse (TreeWalker.java:119) at org.apache.xalan.transformer.TransformerIdentityImpl.transform (TransformerIdentityImpl.java:320)	Check that you have followed the instructions to copy the XML parser-related JAR files to the servlet container's `lib/` directory, and that you have removed any default `parser.jar` files.
Running iPlanet, you see the following in your browser: GX Error (GX2GX) socket result code missing!!!	There is a mismatch between the `web.xml` and `ias-web.xml` files. All servlets mentioned in `web.xml` must have a corresponding entry in the `ias-web.xml` file. Use the kguidgen utility to generate a GUID for the servlet.

Backing Up the Visual Modeler

It is good practice to plan for the possibility of a catastrophic failure that renders the Visual Modeler machine unusable. In this eventuality, you need to be able to restore the Visual Modeler as rapidly as possible.

We suggest taking the following steps:

1. Replicate the servlet container: keep the installable for the exact release of the servlet container, together with any patches applied. Back up any changes to archive files, or startup scripts that might affect the order of class-loading for example. A copy of the JDK used to run the servlet container would also be useful.
2. Back up the Visual Modeler itself: that is, create a WAR file of the running Visual Modeler application directory. This will capture any changes to system properties, business rules, as well as the XML model files, resource files (product images and so on) and other files (such as uploaded GIF files).
3. Note that the Visual Modeler enables a fair amount of customization that can place files outside of the Visual Modeler Web application directory. If you take advantage of this capability, then you have to

backup these directories too. In particular, a clustered installation of the Visual Modeler requires the creation of a shared location that will be external to the Web application directory on any particular machine.

4. Take a snapshot of the database at regular intervals: verify that the Visual Modeler Knowledgebase can be restored from this backup.
5. Make a copy of the `dcmsKey.ser` file and put this in a very safe place. Encrypted data will be unrecoverable if this file is lost. For customers with Release 9.0 or higher, this instruction needs to be modified depending on your choice of encryption scheme and key management policy.

Logging

Use the logging settings of the Visual Modeler to monitor activity of the Visual Modeler and to help diagnose problems.

Logging Preferences and Configuration

The log4j API handles logging and uses the Preferences API to retrieve logging configuration properties. The basic configuration file for the log4j API is `log4j.properties`. A copy of this file with default logging properties is included in the `WEB-INF/lib/cmgt-logging.jar` JAR file packaged with the Visual Modeler and is also placed in the `WEB-INF/classes/` directory. To override the default properties permanently, you must modify the `log4j.properties` file. Any values that you specify in this file will overwrite the corresponding values in the `log4j.properties` file in the `cmgt-logging.jar` file. You can override the default properties on a transient basis for testing purposes by logging in to the System Administration site of the Visual Modeler as a site administrator and modifying the System Logging properties. See "Making Transient Logging Configuration Changes" for more information.

The following sections describe some typical changes you may want to make:

- Logging to the Console
- Changing Logging Level for a Package
- Formatting Logging
- Logging File Size

Logging to the Console

If you want logging output to the standard output stream, rather than to a logging file, specify the use of the STDOUT appender:

```
log4j.rootCategory=info, STDOUT
```

Depending on the configuration of the servlet container, the logging output will be directed to the standard destination of the System.out output stream. Note that when you specify a different appender, then you must include the appender's properties in the custom `log4j.properties` file too. For example:

```
log4j.appender.STDOUT=org.apache.log4j.ConsoleAppender
log4j.appender.STDOUT.layout=org.apache.log4j.PatternLayout
log4j.appender.STDOUT.layout.ConversionPattern=[%t] (%c{2}) - %m%n
```

Note that you can also force logging to be output to the System.out output stream by specifying -Dcomergent.console.logging=true as part of the command line that starts the servlet container. This overrides any logging properties specified in the `log4j.properties` configuration file.

Changing Logging Level for a Package

If you want to see more detailed logging information from just one Java package as it is executed, then you can specify this by overriding the root logging level. By default, all logging is done at the INFO level, because the rootCategory is defined as follows:

```
log4j.rootCategory=info, CMGT
```

For example, to specify DEBUG level logging for the visualModeler API package, enter:

```
log4j.logger.com.comergent.api.apps.visualModeler=DEBUG
```

The specification of logging is hierarchical following the package organization of the Visual Modeler. Thus if you specify:

```
log4j.logger.com.comergent.api.apps=WARNING
```

Then, all logging at the API level will be done at the WARNING level except for the visualModeler package.

Formatting Logging

You can format the logging output to suit your needs. For example, the following will provide a more compact logging format than the standard default layout:

```
log4j.appender.CMGT.layout.ConversionPattern=[%r] [%t] (%c{1}) - %m%n
```

Logging File Size

If log files get too large, then consider modifying the logging preferences to rotate the log files. For example, you can specify that log files are rotated once they reach 10MBytes in size as follows:

After the line:

```
log4j.appender.CMGT=com.comergent.logging.ComergentRollingFileAppender
```

add:

```
log4j.appender.CMGT.MaxFileSize=100KB
```

Alternatively, to specify that log files should be rotated daily, change:

```
log4j.appender.CMGT=com.comergent.logging.ComergentRollingFileAppender
```

to:

```
log4j.appender.CMGT=com.comergent.logging.ComergentDailyRollingFileAppender
```

Making Transient Logging Configuration Changes

If you want to change the logging configuration settings for troubleshooting or other testing purposes, then make the changes as a site administrator using the **System Logging (log4j dynamic)** page of the site administration's System Properties page. Changes that you make remain in effect until you restart the Visual Modeler. If you are working in a clustered environment, then the logging configuration changes will be propagated to all the nodes in the cluster, and will also remain in effect until you restart the Visual Modeler.

To make permanent logging level changes, you must modify the `log4j.properties` file, as described in "Logging Preferences and Configuration".

To make transient logging configuration changes:

1. Navigate to the System Administration URL and log in as a site administrator. The System Administration URL is similar to:

   http://*server*:*port*/Sterling/en/US/enterpriseMgr/admin

   The System Administration home page displays.

2. Click **System Services**, then click **System Logging (log4j dynamic)**.

   The **Configure log4j** page displays, similar to the following figure.

3. Copy the name of the logger you want to change and paste it into the **Logger name:** field, then choose a logging level from the **Threshold** drop-down list.

   Logging levels range from trace (logging all activity) to fatal (logging only fatal errors). You can also use the drop-down list to turn logging off.

4. Click **Update** to update the logging level. The new logging level displays in the Level column for the logger.

Logging File Locations

The location of logging files varies from one servlet container to another: here are some standard locations

Servlet Container	Log File Location
Tomcat	`container_home/logs/`
WebLogic	`container_home/user_projects/domains/mydomain/`
WebSphere	`container_home/logs/`

Managing Database Connections

This section describes how to manage the connections between the Visual Modeler and the Knowledgebase.

Configuration Files

The following files manage the configuration of the data services layer:

- `DataServices.xml`: this file is contained in the `WEB_INF\lib\cmgt-dataservices.jar` file that ships with the Visual Modeler. This file specifies values for all the data services properties unless they are overridden by the `prefs.xml` configuration file.
- `prefs.xml`: this file contains the properties and their values created during the installation process. In addition, if you make changes to the system properties through the Visual Modeler administration UI, then the changes are persisted to this file.

Connection Pooling

This section answers some common questions about connection pooling.

What is the purpose of connection pooling?

Establishing a database connection is typically processor-intensive. The use of connection pools allows us to maintain a set of open connections for use by database requests. These connections can then be allocated to short duration SQL requests and then immediately returned to the pool for re-use.

How does connection pooling work?

The Visual Modeler maintains these logical pools of connections:

- Pool of message-based connections. This pool is simply a HashMap that mates a message version to an appropriate Message-based DataService. One DataService instance is shared by all requests requiring that message version.
- Pool of database connections. This pool is used for all database requests. When a data bean *persist()* or *restore()* method is invoked, we retrieve a connection from the pool; process the operation; then return the connection to the pool for reuse.

  In past releases the Visual Modeler supported sharing a database connection across multiple concurrent requests. Not all databases are capable of supporting this functionality. In addition, performance testing has shown that an expensive SQL request can drastically impact the performance of all requests sharing the same connection. Based on these issues, the Visual Modeler has eliminated support for sharing of Query connections.

Why are there separate query and update connection cools?

The use of separate pools for Query and Update Connections allows the Visual Modeler to optimize connections for read-only access.

How do I validate connections prior to reuse?

The following properties control connection timeout and validation:

- The ConnectTimeout element provides a timeout setting for connections in the pools. The value is the number of minutes for a connection to timeout. For example, if you set this value to “1”, then if a connection has been unused for more than one minute, it is validated before being used. A setting of 0 means that connections do not timeout.
- The ReconnectOnTimeout element controls what is done when a connection timeout.
  - A setting of “true” indicates that when a connection times out it will automatically reconnect the next time it is retrieved from the pool.
  - A setting of “false” indicates that a connection timeout will result in the connection being validated prior to reuse. If the validation fails, then a reconnect will occur.

How can I limit the number of connections used?

Each DataSource specified in the `DataServices.xml` configuration file supports a MaxConnections property. This specifies an absolute upper limit on the number of connections that will be used. A setting of “-1” indicates there is no limit. The `DataServices.xml` file is contained in the `WEB_INF\lib\cmgt-dataservices.jar` file.

How can I free up connections when demand drops?

Each DataSource specified in the `DataServices.xml` configuration file also supports a MaxPoolSize property. This provides a soft limit on the number of connections that will be pooled. A setting of “-1” indicates there is no limit. The pool size is not an absolute limit, but as connections are released the pool will gradually move back down to its maximum size.

The Visual Modeler does allow the number of connections to grow beyond the maximum pool size, but when the number of free connections exceeds a preset limit we will begin releasing connections until the number of connections eventually drops back to the maximum pool size. We do this gradually to avoid excessive connection requests when pool is at the boundary.

What happens when connection limits are reached?

If a connection is requested from the pool, but no free connections are available, then we would normally create a new connection. If the connection limit is reached, then we will instead wait for a connection to be returned to the pool.

Why are the connection limits on the data source?

Providing connection limits for each data source provides greater flexibility in allocating connection resources. For example, this allows you to limit the number of connections to a back-end ERP system, while providing higher limit when accessing the primary database server.

Common Problems

This section describes some common problems.

My database requests fail with a “connection reset by peer” message

This error is normally a result of either the database server timing out the database connection, or of the network connection being timed out by a firewall. This problem can be resolved by setting the ConnectionTimeout element to ensure validation of connections that exceed the timeout.

My database connections are not being released when traffic drops

Setting the MaxPoolSize property on the data source will allow the number of database connections to drop back to predefined limits as connections are freed.

I share a database with other applications. I cannot allow the Visual Modeler to use more than n connections

Setting the MaxConnections property on the data source will allow you to limit the maximum number of database connections used by the Visual Modeler.

Pagination Settings

The Visual Modeler supports the use of paginated data sets so that long lists can be displayed one page at a time. This functionality is implemented by saving a set of files to the Visual Modeler machine's file system. These represent the pages of data objects to be paged through. The location of the paginated file sets is determined by the rsCachePath element in the `DataServices.xml` file, contained in the `WEB_INF\lib\cmgt-dataservices.jar` file. The rsCachePathIsAbsolute element is used to specify whether the value of the rsCachePath element should be treated as a relative or absolute path. By default, its value is "false" and so the path is treated as being relative to *`debs_home`*`/Sterling/`. The adjustFileName() method call is used to resolve this location to an absolute location in the servlet container's file system.

If your implementation of the Visual Modeler uses a cluster of servlet containers, then the location of the pagination directory must be accessible to all members of the cluster. See "High Availability and Clustering".

Setting the Session Timeout

Servlet containers and Web applications attach a session to each user interaction with the server. By this means, they can maintain information from one request to another as a user interacts with the application. To help ensure that a user's browser is not used by an unauthorized user, the servlet container will mark a session as being invalid once a certain time has elapsed from the time when the session was last accessed. This is referred to as the session timeout period. Sessions automatically become inactive if the time from the last access exceeds the session timeout setting.

You can set the session timeout period in the Visual Modeler `web.xml` configuration file using the session-timeout element. For example, to timeout sessions after 30 minutes, set the element to:

```
<session-timeout>30</session-timeout>
```

When setting the session timeout period, bear in mind the following:

- The longer the time out, the greater the risk that the servlet container will run out of memory. Each session takes up space in memory, and when objects are added to the session, then the memory usage increases. Often, users may not actively log out: their session will stay resident in memory until the servlet container times it out. If your Web site is likely to see heavy user traffic, then bear in mind this memory consumption when determining JVM memory settings.
- The longer the timeout, the greater security risk presented: either by an unauthorized person using an unattended Web browser or by an unauthorized person spoofing a session simply by guessing its session ID.
- The session timeout period must be sufficiently long to enable users to complete their tasks. If the tasks include activities such as using a third-party Web application or obtaining information from a third-party source, then allow for this amount of time so that a user is not inadvertently timed out of the Visual Modeler.

For these reasons, we suggest setting a session timeout value of 30 (30 minutes). However, you must assess the needs of your implementation and select a value accordingly.

Modifying the URL for the Web application DTD

When you start the servlet container, the Visual Modeler is loaded as a Web application. The `web.xml` file configuration file is read to determine the basic configuration of the application. The **web.xml** file is validated against a DTD specified by its web-app element.

By default, the validating DTD is at the URL:

```
http://java.sun.com/dtd/web-app_2_3.dtd
```

However, access to this URL can be limited either by your network status or by Sun Microsystems. As an implementation step, we recommend that you modify the validating URL to point to a copy of the DTD whose location is assured by your implementation.

Our suggested solution is to use a relative URL to reference the DTD within the Visual Modeler context. For example:

```
/WEB-INF/lib/web-app_2_2.dtd
```

Note: The form of this relative URL is servlet container-specific.

Alternatively, you can add the DTD to a Web server and point to this location. For example, if you are using a Web server to act as a front-end to the Visual Modeler, then put the DTD on this Web server.

Managing Memory

In general, you should allocate as much memory as possible to the JVM running your application server. Typically, this is done by modifying the configuration parameters that are used to start the Java process, say:

```
-Xms256M -Xmx512M -XX:MaxPermSize=128M
```

However, if your system is likely to experience heavy load at times, then you can use a Visual Modeler configuration parameter to ensure that the system can recover from a burst of memory-intensive activity.

Set the memoryThreshold element of the C3_Commerce_Manager element of **Comergent.xml** to an integer value between 0 and the maximum allocated memory size (in Kilobytes). When memory usage exceeds this value, then new requests will be refused with the HTTP status of 503. The default value, -1, disables the parameter.

For example, suppose that you have set the maximum memory to -Xmx512M (524,288K). Suppose that you set memoryThreshold to 498074. Then when memory usage exceeds 498,074K, new requests are refused until memory usage has dropped back down to below this value.

Configuring Ehcache

The Visual Modeler uses Ehcache to manage data caching, both for clustering and for the global application cache. Out of the box, Ehcache manages pre-configured caches for applications such as pricing, product categories and features. You configure new caches and modify cache properties such as the amount of time to cache data and the maximum number of elements in memory by editing the `WEB-INF\properties\EhCache.xml` file.

High Availability and Clustering

The Visual Modeler can be deployed in a distributed environment in which more than one individual instances of the Visual Modeler run as a cluster. This provides support for ensuring high availability of the Visual Modeler and to support fail-over of individual machines.

Sharing Directories

In some deployments of the Visual Modeler, for example a clustered deployment, you must specify the location of directories to be used for uploaded and generated files. The locations of these directories is specified using the `web.xml` file to set context parameters.

You have these sets of attributes for the directories to specify:

- share-noshare: share directories can be accessed by two or more machines, noshare directories should be accessed only by the machine whose `web.xml` file specifies the location.
- public-private: public directories must be accessible by the Web server serving the static content, private directories should not be.
- loadable-noloadable: loadable directories can be used to upload files, noloadable directories should not be used for uploaded files.

The same directory can be used for more than one of these combinations of choices.

At minimum, you must specify the location of the share.public.loadable and share.public.noloadable directories. If you have two or more machines in a cluster, then these directories must be accessible from all of the cluster machines.

The `web.xml` file lets you specify how a front-end Web server can access files in the public directories. Use the WebPathToPublicLoadableWritableDirectory element to map a Web server virtual path to the directory identified by the share.public.loadable element. Use the WebPathToPublicNoLoadableWritableDirectory element to map a Web server virtual path to the directory identified by the share.public.noloadable element. These elements should reflect the Web server settings used to specify virtual paths.

To set these directories up, you typically perform these steps:

1. Select one of the machine as the “primary machine”. Allocate a directory on this machine to provide the shared location.
2. Share this location so that all member of the cluster have access to it:
   - Windows: share this directory to the other machines
   - UNIX: use NFS to share the directory
3. On all machines, mount the file system so that all cluster members have the same mount point to this directory. For example:

   ```
   /DEBS_shared
   ```
4. Under **DEBS_shared/**, create a sub-directory for each of the categories shown in the configuration file (loadable, writable, and so on), for example:

   ```
   /DEBS_shared/lw
   ```

   and set that value in the configuration file, for example:

   ```
   <loadable ...>/DEBS_shared/lw</loadable>
   ```

Directory and File Organization

When the `Sterling.war` Web application is deployed to the servlet container, it is deployed into a directory, *`debs_home`*, that you specify during the deployment or which the servlet container sets. This section describes the organization of the sub-directories under *`debs_home`*.

Beneath this directory, a sub-directory is created for the Visual Modeler application when the Web application is deployed. This directory is the Web application directory for the Visual Modeler. We refer to it as *`debs_home`*`/Sterling/`. It contains:

- A locale directory for each supported locale. Each locale may be expressed as <*la*>_<*CO*>, where *la* is one of the standard language codes and *CO* is one of the standard country codes, for example: en_US or fr_CA. For a locale, the corresponding directory is *`debs_home`*`/Sterling/la/CO/`. This directory contains:
  - **css/**: holds the cascading style sheets used by the Visual Modeler.
  - **images/**: holds common images used by the Visual Modeler.
  - **js/**: holds Javascript libraries used by the Visual Modeler.
  - **htdocs/**: holds the HTML templates, images, and online help for the Visual Modeler.
- **dXML/**: holds the DTDs for the dXML message types.
- **htdocs/**: a directory for content that can be served up directly by the servlet container or Web server. Content stored here should not be locale-specific.
- **j2ee/**: a directory to hold local copies of the J2EE DTDs. See "Modifying the URL for the Web application DTD" for more information.
- **WEB-INF/**: holds all the configuration files used by the server. It contains the following subdirectories:
  - **bizobjs/**: holds the business object DTDs. These DTDs are used to validate XML messages. The DTDs can be generated automatically by the generateDTD target provided by the SDK.
  - **classes/**: holds the Visual Modeler Java classes.
  - **commerceone/**: used as part of Commerce One integration.
  - **converters/**: holds the configuration files used in message conversion.
  - **data/**: holds data provided as part of the reference implementation.
  - **extralib/**: holds class libraries that are needed for implementation work, but that should not be used at runtime.
  - **integrator/**: holds the configuration files for Sterling Integrator.
  - **lib/**: holds the Java class libraries used by the Visual Modeler Web application.
  - **lib/winnt/**: this holds any Windows-specific DLL files that are required.
  - **messages/**: holds the DTDs for the Sterling message family.
  - **properties/**: holds the **Comergent.xml** file and the other configuration files used to set the configuration of the server.

- **rosettanet/**: contains the DTD and XML files that define the RosettaNet messages supported by the Visual Modeler.
- **schema/**: holds the XML files that specify the schema for your implementation.
- **stylesheets/**: holds the XSL files used to translate messages from one message family to another.
- **templates/**: holds the text templates used to generate messages such as email notifications.
- **web/**: holds most of the JSP pages, HTML pages and support files for the applications. It has the following structure:
- ***/locale/*** directories for each of the Visual Modeler applications. Each locale supported by your implementation of the Visual Modeler must have its own set of JSP pages in its corresponding locale directory.
- Each locale may be expressed as <*la*>_<*CO*>, where la is one of the standard language codes and CO is one of the standard country codes, for example: en_US or fr_CA. For a locale, the corresponding directory is `debs_home/Sterling/WEB-INF/web/la/CO/`.
- ***x509/:*** holds the certificates used to authenticate SSL sessions.

Setting Up Apache as a Front-end to Tomcat

This section describes how to set up an instance of Apache HTTP Server 2.2 Web Server so that it can act as a front-end to a deployment of the Visual Modeler on Tomcat. It uses the JK 1.2 connector supplied by the Apache Jakarta Project.

This section assumes that Apache and Tomcat are installed on two different machines, referred to as the Web server machine and servlet container machine respectively.

Prerequisites

1. Install Apache Web Server on the Web server machine.
2. Deploy the Visual Modeler into the instance of Tomcat running on the servlet container machine.
3. You should confirm that both Apache and Tomcat can be started individually with no error. In particular, make sure that the deployment of the Visual Modeler in Tomcat works correctly using the Tomcat port.

Overview

JK 1.2 is a connector that connects an Apache instance with a Tomcat instance. This allows Apache to serve as a front-end Web server for Tomcat. There are several advantages to this kind of setup:

- You can configure Apache to manage page expiration (reducing the number of HTTP requests).
- You can configure Apache to compress responses (reducing the number actual bytes transmitted).

Once the connector is set up and configured to work properly, a typical request flow is as follows (using default ports):

1. The browser connects to Apache's port 80 and submits its request.
2. Apache determines if the incoming URL needs to be managed by the JK 1.2 connector, mod_jk.
3. If so, then Apache initiates an AJP 1.3 connection to Tomcat's port 8009. The request is now sent to Tomcat.
4. Tomcat processes the request and returns the response through the same AJP 1.3 connection.
5. Apache in turn relays the same response to the browser.

Configuring Apache to Use mod_jk

1. Download a copy of `mod_jk-apache-2.0.58.so` for Apache. The location is similar to http://tomcat.apache.org/download-connectors.cgi. For the rest of these instructions, we assume that you rename this file to `mod_jk.so`.
2. Put the `mod_jk.so` file in the Apache Web server *`apache_home`*`/modules/` directory.

3. Edit the *apache_home*/conf/httpd.conf file as follows:
   a. Add the following line in the LoadModule section:

      ```
      LoadModule jk_module modules/mod_jk.so
      ```

      Take care to provide the exact name of the mod_jk module file.
   b. Add an IfModule element to force Apache to set up the Tomcat servlet container connection and access to the Visual Modeler web application:

      ```
      <IfModule mod_jk.c>
      JkWorkersFile apache_home/conf/workers.properties
      JkLogFile apache_home/logs/mod_jk.log
      JkLogLevel info
      JkMount /Sterling/* ajp13
      </IfModule>
      ```
   c. Add the following line to the end of httpd.conf:

      ```
      Include /tomcat-home/conf/auto/mod_jk.conf
      ```
4. Ensure that the sample Tomcat tomcat_home/conf/workers.properties file is similar to the following:

   ```
   # Set properties for Tomcat
   worker.list=worker1
   worker.worker1.port=8009
   worker.worker1.host=<servlet_container_machine_name>
   worker.worker1.type=ajp13
   ```

   Replace <*servlet_container_machine_name*> with the name of the servlet container machine.
5. Ensure that the corresponding apache_home/conf/workers.properties file is similar to the following:

   ```
   # Define 1 real worker using ajp13
   worker.list=ajp13
   # Set properties for worker1 (ajp13)
   worker.ajp13.type=ajp13
   worker.ajp13.host=<servlet_container_machine_name>
   worker.ajp13.port=8009
   worker.ajp13.lbfactor=50
   worker.ajp13.cachesize=10
   worker.ajp13.cache_timeout=600
   worker.ajp13.socket_keepalive=1
   worker.ajp13.recycle_timeout=300
   ```

   Replace <*servlet_container_machine_name*> with the name of the servlet container machine.

Configure Tomcat to Use mod_jk

By default, Tomcat is pre-configured to listen on port 8009 for ajp13 connections. Ensure that the following entry is in the Tomcat *tomcat_home*/conf/server.xml file:

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
   <Connector port="8009"
   enableLookups="false" redirectPort="8443" protocol="AJP/1.3" />
```

Edit Tomcat's *tomcat_home*/conf/server.xml file:

- Add the following line to the Listeners section:

  ```
  <Listener className="org.apache.jk.config.ApacheConfig"
  modJk="<apache-home>/modules/mod_jk.so" />
  ```

Starting Apache and Tomcat

1. Start up Apache, then start up Tomcat.
2. Try to verify whether you can access the Visual Modeler through Apache.

   ```
   http://<web server>/Sterling/en/US/enterpriseMgr/matrix
   ```

Setting up Apache to Support SSL

If you set up Apache as a front-end to Tomcat, then you can use the SSL capabilities of Apache to manage secure access to the Visual Modeler. The following steps provide an outline as to how to do this. Note that we do not provide a compiled binary of the Apache SSL module. You must either obtain this from a third-party such as: http://hunter.campbus.com/, or build it yourself using the OpenSSL source obtained from: http://www.openssl.org/source/. Once you have created mod_ssl.so and copied it to *apache_home*/modules/, then follow these steps:

1. Uncomment in the following line in *apache_home*/conf/httpd.conf:

   ```
   LoadModule ssl_module modules/mod_ssl.so
   and
   <IfModule mod_ssl.c>
      Include conf/ssl.conf
   </IfModule>
   ```

2. Create the *apache_home*/conf/ssl.conf file. This file is where you specify your SSL configuration using the SSL directives. It should look something like this:

   ```
   Listen 443
   <VirtualHost _default_:443>
   ServerName http://www.example.com
   SSLEngine on
   SSLCertificateFile /usr/local/apache2/conf/server.cert
   SSLCertificateKeyFile /usr/local/apache2/conf/server.key
   </VirtualHost>
   ```

3. Obtain or generate the certificates and keys for your site. You can use the openssl utility to generate a self-signed key and certificate using commands like this. First create the key by using:

   ```
   openssl req -new -nodes -out server.csr
   -keyout server.key -config openssl.cnf
   ```

   Then use the key to generate the certificate:

   ```
   openssl x509 -in server.csr -out server.crt -req
   -signkey server.key -days 365 -set_serial 1 -config openssl.cnf
   ```

   The -config parameter points to your `openssl.cnf` configuration file that can be used to maintain OpenSSL configuration information.

4. Copy the key and certificate to the location specified by the SSLCertificateFile and SSLCertificateKeyFile properties of the `ssl.conf` file.

5. Restart Apache.

Keep Alive Settings

In some circumstances, problems have been reported with Apache and SSL such as slow and dropped connections. If you encounter these, then consider these steps:

1. Make sure that the setting for KeepAlive is On in *`apache_home`*`/conf/httpd.conf`:

   ```
   KeepAlive On
   ```

   It appears that this setting is set to Off as default in some distributions of Apache.

2. Older versions of IE, in particular IE 5.x, have a bug in the SSL/TSL shutdown and keepalive feature. A work-around for these bugs is to configure Apache's SSL to behave in a non-standard way for these connections. In *`apache_home`*`/conf/ssl.conf`, add the following lines if they are not there already:

   ```
   SetEnvIf User-Agent ".*MSIE.*" \
   nokeepalive ssl-unclean-shutdown \
   downgrade-1.0 force-response-1.0
   ```

Compressing Output From the Visual Modeler

If network performance is a high concern, then one step that you can take is to configure the Visual Modeler so that it returns compressed output to the users' browsers, and the browsers decompress the output to render the page. This section describes how to use Apache to do this. Note that an alternate approach is to use Servlet Specification 2.3 filter to perform the compression.

These steps assume that you have set up Apache as a front-end to the servlet container in which the Visual Modeler is deployed.

1. Edit the Apache `httpd.conf` configuration file to add or uncomment:

   ```
   LoadModule deflate_module modules/mod_deflate.so
   LoadModule headers_module modules/mod_headers.so
   ```

2. Copy the following text into *apache_home*/conf/httpd.conf. Putting it at the bottom of the file is fine.

```
<Location /Sterling>

# Insert filter
SetOutputFilter DEFLATE

# Netscape 4.x has some problems...
BrowserMatch ^Mozilla/4 gzip-only-text/html

# Netscape 4.06-4.08 have some more problems
BrowserMatch ^Mozilla/4\.0[678] no-gzip

# MSIE masquerades as Netscape, but it is fine
# BrowserMatch \bMSIE !no-gzip !gzip-only-text/html

# NOTE: Due to a bug in mod_setenvif up to Apache 2.0.48
# the above regex won't work. You can use the following
# workaround to get the desired effect:
BrowserMatch \bMSI[E] !no-gzip !gzip-only-text/html

# Don't compress images
SetEnvIfNoCase Request_URI \.(?:gif|jpe?g|png)$ no-gzip dont-vary

# Make sure proxies don't deliver the wrong content
Header append Vary User-Agent env=!dont-vary

</Location>
```

If necesarry, change the context string “/Sterling” to the name of the context used for the Visual Modeler.

Creating the Knowledgebase Schema

You run the schema creation script to create the Knowledgebase in your designated database server. You can run the schema creation script as a batch file as described in this section or from the SDK. See "To Run the Schema Creation Script with the SDK".

Note: Running the schema creation scripts directly is no longer supported. You must run the createDB target as described in "Installing the Visual Modeler Using the SDK".

Before using the schema creation scripts and XML Loader scripts, you must decide what locales your implementation will support. You should remove any locales from the scripts that you do *not* intend to support. To begin with, you can try just creating the “en_US” locale and adding more as the implementation progresses.

Creating the Schema

Note: To run the database schema creation target, you must have the correct database client software installed: SQLPLUS for an Oracle server installation, or OSQL for Microsoft SQL Server,.

The schema creation script is a batch file that connects to the database server and then runs a sequence of SQL scripts to create the database objects.

- If you are running against an IBM DB2 Universal Database database server, then run DB2CreateSchema.bat from the debs_home/Comergent/ directory.

  In each SQL file, edit the Connect command to use the catalog entry set up to connect to the DB2 Server:

  ```
  connect to <catalog name>
  ```

  or

  ```
  connect to <catalog name> user <database username> using <database password>
  ```

- If you are running against an Oracle Server database server, then run one of the following scripts from the *debs_home*/Sterling/ directory:
  - `OracleCreateSchema.bat` (for Windows)
  - `OracleCreateSchema.sh` (for UNIX)

  By default, the Oracle schema creation script creates indexes for tables in a separate tablespace called INDX. You can choose to create the indexes in the same tablespace as the main schema or in a tablespace of your choice. To do either of these two choices, you must edit the `oracle_indexes.sql` SQL script.
- If you are running against a Microsoft SQL Server database server, then run `MssqlCreateSchema.bat` from the *debs_home*/Sterling/ directory. Make sure the database is the default database for the User ID you use.

Locales and Loading Data Using the XML Loader

The current schema creation script creates seven locales as part of the table creation script. You should review this part of the table creation script and modify it to remove locales if need be before running the script.

For each locale, you must take care to ensure that the correct value in the DB_SORT_LOCALE_NAME column of the CMGT_LOCALE table is set.

The following database sorting settings table summarizes the most frequently used values for this column:

Database	**Locale**	**Value**
Oracle	en_US	BINARY
	de_DE	XGERMAN
	fr_FR	XFRENCH
	fr_CH	BINARY
	de_CH	BINARY
	ja_JP	BINARY
	zh_TW	BINARY
SQL Server	en_US	Latin1_General_BIN
	de_DE	FRENCH_CS_AS
	fr_FR	FRENCH_CS_AS
	fr_CH	SQL_Latin1_General_CP1_CI_AS
	de_CH	SQL_Latin1_General_CP1_CI_AS
	ja_JP	SQL_Latin1_General_CP1_CI_AS
	zh_TW	Chinese_Taiwan_Stroke_CS_AS

For other locales, please contact your Support representative for further information. DB2 does not require these settings: the database sort order is set at the database level and cannot be changed.

To Run the Schema Creation Script with the SDK

As an alternative to running the schema creation scripts as batch files, you can also run the schema creation script from within the SDK as follows:

1. Edit the appropriate SDK project `*.properties` file to enter the connection information used by the createDB target. Typically, during an implementation cycle, this is the `project_dev.properties` file to be found in the *`sdk_home`*`/projects/project/templates/` directory.
   a. To create the knowledgebase on an Oracle database server, enter:

      ```
      DB_TYPE=Oracle
      ORACLE_URL=jdbc:oracle:thin:@<Machine>:<Port>:<SID>
      ORACLE_USERNAME=<Username>
      ORACLE_PASSWORD=<Password>
      ORACLE_DATABASE=<TNS alias>
      ```

   b. To run the knowledgebase on SQL server, enter:

      ```
      ODBC_URL=<ODBC DSN>
      ODBC_USERNAME=<Username>
      ODBC_PASSWORD=<Password>
      ```

   c. To run the knowledgebase on DB2 server, enter:

      ```
      jdbc:db2://DB2_MACHINE:DB2JD_PORT/DATABASE
      ```

2. If you have deployed the Visual Modeler in WebSphere, then you must edit the classpath setting portion of the script to allow for the fact that the class and library files are in the *`debs_home`*`/Sterling/servlets/` directory.
3. Check that the appropriate database client software has been installed on your machine and that it is in your path. For example, if you are using an Oracle server for the Knowledgebase, then make sure that SQLPLUS is in your path. Make sure also that the `tnsnames.ora` file includes an alias as specified as the ORACLE_DATABASE parameter. You should be able to successfully run:

   ```
   tnsping <TNS alias>
   ```

4. Run the createDB target from the SDK.
5. Check the generated log files to verify that the script ran without error.

Localization Concepts

This section introduces localization concepts and the Visual Modeler support for localization. You can provide your customers with an e-commerce experience in their preferred language and location, or locale.This topic describes how to add support for locales other than United States English to an implementation of the Visual Modeler.

Individual locales are provided as localization pack JAR files that you install into your release using the SDK.

Note: There are known issues with the Visual Modeler using SQL Server to support locales other than en_US.

Built-in Localization Support

The Visual Modeler has built-in support for:

- multiple currencies
- multiple languages
- number and date formats
- character sets

You can manage other aspects of localization for specific markets, such as:

- local laws and regulations
- currency processing
- shipping and export information
- tax

Locale Specification

You manage support for internationalization using locales. Each locale identifies a language and country. By identifying the locale to use when displaying information to a user, you ensure that the user sees information that is specific to their locale: they see your site's Web pages in their preferred language, with numbers and dates in their expected format.

A locale comprises a language and a country: for example, "English and United States" or "Italian and Switzerland". The same language may be used in more than one country: French in France, Switzerland, and Canada for example. In one country there may be speakers of more than one language: French, German, Italian, and Romansch in Switzerland for example.

The ISO standards 639 and 3166 specify a list of standard abbreviations for languages and countries that you must use. Some common language abbreviations are: Arabic (ar), Chinese (zh), English (en), French (fr), German (de), Hindi (hi), Japanese (ja), and Spanish (es).

Some common country abbreviations are: Canada (CA), China (CN), France (FR), Germany (DE), India (IN), Indonesia (ID), Japan (JA), United Kingdom (GB), and United States (US).

By combining a language and a country, you can uniquely specify a locale. For example: en_US (English-United States), it_CH (Italian-Switzerland), and zh_TW (Chinese-Taiwan). Locales are stored in the Visual Modeler using this representation.

Using Locales

Each installation of the Visual Modeler defines a *system default locale* in its `Internationalization.xml` file using the defaultSystemLocale element.

User Locales

When a user works in the Visual Modeler, their *current locale* determines the look-and-feel of Web pages and the locale-specific data (such as product descriptions) to use to display business object data to the user.

Each user has a *preferred locale* specified in their user profile. When a user first enters the Visual Modeler, their current locale is set to their preferred locale. If they change their current locale as they work, they see the Web pages in the new locale.

Users change locale by selecting a new locale from a drop-down list of available locales in their user profile. The display names for each locale in the drop-down list depend on the user's current locale. For example, if the user's current locale is en_US, the display name for fr_FR can be "French-France", and if the user's current locale is fr_FR, then the display name for en_US can be "Anglais-Etats Unis".

Default Locales for Languages

The Visual Modeler enables you to specify a default locale for each language so that if JSP pages are not available for the specific locale, the language's default locale's JSP pages are used instead. You specify the language's default locale using the defaultCountry elements of the Languages element of the `Internationalization.xml` configuration file.

The defaultSystemLocale element determines which JSP pages to serve if they are not provided in the language's default locale directory.

Sorting in Locales

The Visual Modeler enables users to sort displayed data in a number of ways while they perform their tasks. For example, they can sort the display of partners by name or inquiry lists by inquiry list ID. When a column is sorted based on a String value, you can specify whether the sorting is performed using the binary value of the String or whether a locale-specific sort is used. This switch is set at the system level, so the same method is used for all such sorts.

Note: SQL Server support for user-locale sorting is limited. You can set only a single collating sequence which is then used for all users.

The sorting behavior is controlled by the UseLocalizedSort element of the `DataServices.xml` configuration file. The types of sorting behavior are:

- Binary: sorting is based on the binary value of the String value
- Locale specific

By default, the value of the UseLocalizedSort element is “false”: binary sorting is used. To use locale-specific sorting, set UseLocalizedSort to “true”. Note that binary sort is always fastest.

You can change the value of the UseLocalizedSort element by editing the `DataServices.xml` configuration file.

Site administrators can also change the UseLocalizedSort value as follows:

1. Navigate to the Visual Modeler administration site as a site administrator.
2. Click **System Services** to display the System Properties page, then click `DataServices.`
3. Scroll to Use Localized Sorting, then select the **false** or **true** radio button as required for your site.

Note: You must stop and restart the Visual Modeler to enable the change.

The sorting behavior must be supported by the knowledgebase on which the Visual Modeler runs. See your database administrator for information about the type of sorting behavior supported for your implementation.

Localization Pack Installation Overview

The localization pack installation instructions describe how to install a Visual Modeler localization pack using SDK. You can install into a new implementation or add support for a locale to an existing implementation. You install the localization pack into your release, not into your project.

These instructions assume that you are adding support for one or more locales to your new or existing implementation, but the default locale remains en_US.

What’s contained in a localization pack:

- The resource bundle properties files (*.properties) for the locale.
- The locale’s version of the look-up information from the `xmlloader` file, such as Keytype property names and values (descriptions).
- All the Javascript (.js) files.
- All the properties files with the locale appended to the file names, for example, `AttribMgrAGGENResources_fr.properties`.
- The .js files translated into the locale. The .js files retain their original names but are placed in locale-specific directories, such as fr\FR\js.

Note: The Matrix reference data, online help and online help images are not localized.

Localization Pack Installation Steps: New Implementation

These steps apply to a new implementation of the Visual Modeler.

These steps assume that you completed the Release 9.0 Visual Modeler set-up steps, including:

- Database configuration, privileges grants, and so on
- Installation of the Release 9.0 JAR file into your SDK
- Creation of a project using the `sdk newproject` target
- Installation of the appropriate database driver using the `sdk installDB` target, where *DB* is Oracle, MSSQLJDBC, or DB2.

1. Install the localization pack JAR file. For example:

   ```
   sdk install SterlingSellingSuite-Locale-DEBS-9.0-fr-FR.jar
   ```

   This installs the localization pack in the `releases\debs-9.0` directory.

2. Modify the `web.xml` file:
   a. Enter the following in a command window:

      ```
      sdk customize web.xml
      ```

   b. Open *`sdk_home`*`\projects\`*`project-name`*`\WEB-INF\web.xml` in a text editor.
   c. Copy the section that begins with the following:

      ```
       <!-- Start of English US mapping -->
      ```

      And ends with the following:

      ```
      <!-- End of English US mapping -->
      ```

   d. Paste the entire section after the English US mapping section.
   e. Modify the new section's comments to refer to the locale name you are adding. For example:

      ```
      <!-- Start of French France mapping -->
      ```

   f. Modify the new section's file path references to refer to the locale name file path, such as /fr/FR. For example:

      ```
      <servlet-mapping>
            <servlet-name>DispatchServlet</servlet-name>
            <url-pattern>/fr/FR/catalog/*</url-pattern>
      </servlet-mapping>
      ```

3. Configure the `Internationalization.xml` file.
   a. Enter the following in a command window:

      ```
      sdk customize Internationalization.xml
      ```

   b. Open `projects\`*`project-name`*`\templates\WEB-INF\properties\Internationalization.xml` in a text editor.

c. Add the locale designation to the Presentation element's supportedLocales field. For example, for the French France locale:

```
<supportedLocales controlType="text" runtimeDisplayed="true"
ChangeOnlyAtBootTime="true" visible="true" boxsize="60"
displayQuestion="Presentation Locales"
displayOptions="en_US,en_US (English-United States),
zh_TW,zh_TW (Chinese-Taiwan),fr_FR,fr_FR
(French-France),fr_BE,fr_BE (French-Belgium),
de_DE,de_DE (German-Germany)" defaultChoice="en_US"
help="Supported presentation locales.">en_US,fr_FR</supportedLocales>
```

d. Add the following to the Languages element. Replace ***la*** with the language code and ***CO*** with the country code for your locale:

```
<la visible="false">
<defaultCountry controlType="text" runtimeDisplayed="true"
ChangeOnlyAtBootTime="true" visible="false" boxsize="60"
displayQuestion="URL for the Help Files" defaultChoice="US"
help="This is the default country for a specific language">CO
</defaultCountry>
</la>
For example, for the French France locale:
<fr visible="false">
<defaultCountry controlType="text" runtimeDisplayed="true"
ChangeOnlyAtBootTime="true" visible="false" boxsize="60"
displayQuestion="URL for the Help Files" defaultChoice="FR"
help="This is the default country for a specific language">FR
</defaultCountry>
</fr>
```

e. Add the locale table entry.

The locale table entry includes the DB_SORT_LOCALE_NAME field, which specifies how to sort data that your users are viewing. Possible values are BINARY (sort by the binary value of the string data value) and LATIN_GENERAL_BIN (perform locale-specific sorting). See "Sorting in Locales" for information about determining sorting behavior for your implementation. Check with your Database Administrator to ensure that the sorting behavior that you select is supported for your database implementation.

For Oracle-based implementations:

a. Enter the following in a command window:

```
sdk customize oracle_tables.sql
```

b. Open `projects\`*`project-name`*`\WEB-INF\sql\Oracle\setup\oracle_tables.sql` in a text editor.

c. Search for the text "INSERT INTO CMGT_LOCALE"

d. Add an INSERT statement for the locale you are installing after the INSERT INTO CMGT_LOCALE statement. The format is:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY,LOCALE_NAME,LOCALE_DESCRIPTION,ACTIVE_FLAG,DB_SORT_LOCALE_NA
ME )
VALUES ( key,'la_CO','Country Language','Y','BINARY')
/
```

- *Key* is the locale key. The locale key must be a unique numeric value.
- *la_CO* is the language code and country code. Use the appropriate language_COUNTRY encoding for the locale you are installing, for example, fr_FR for French France, jp_JP for Japanese Japan, and so on.

  For example, the following INSERT statement is for supporting the fr_FR (French France) locale:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY,LOCALE_NAME,LOCALE_DESCRIPTION,ACTIVE_FLAG,DB_SORT_LOCALE_NA
ME )
VALUES ( 2,'fr_FR','France French','Y','BINARY')
/
```

e. Save and close the file.

For SQL Server 2005-based implementations:

a. Enter the following in a command window:

```
sdk customize mssql_schema.sql
```

b. Open `projects\`*`project-name`*`\WEB-INF\sql\MSSql\setup\mssql_schema.sql` in a text editor.

c. Search for the text "INSERT INTO CMGT_LOCALE"

d. Add an INSERT statement for the locale you are installing after the INSERT INTO CMGT_LOCALE statement. The format is:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY,LOCALE_NAME,LOCALE_DESCRIPTION,ACTIVE_FLAG,
DB_SORT_LOCALE_NAME ) VALUES ( key,'la_CO','Country
Language','Y','LATIN_GENERAL_BIN')
GO
```

- *Key* is the locale key. The locale key must be a unique numeric value.
- *la_CO* is the language code and country code. Use the appropriate language_COUNTRY encoding for the locale you are installing, for example, fr_FR for French France, jp_JP for Japanese Japan, and so on.

  For example, the following INSERT statement is for supporting the fr_FR (French France) locale:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY,LOCALE_NAME,LOCALE_DESCRIPTION,ACTIVE_FLAG,
DB_SORT_LOCALE_NAME )
VALUES ( 2,'fr_FR','France French','Y','LATIN_GENERAL_BIN')
GO
```

e. Save and close the file.

4. Add the country translations:

   a. Enter the following in a command window:

      ```
      sdk customize LocaleNameDataList
      ```

   b. Open `projects\`*`project-name`*`\WEB-INF\xmldata\LocaleNameDataList` in a text editor.

   c. The `LocaleNameDataList` file should contain only English and the language mappings for the locale(s) that you support. Remove all other Locale Mappings from the `LocaleNameDataList` file.

   d. At the end of the LocaleNameDataList clause, add lines to supply translations for the United States and the country for which you are installing a locale. For example, to provide French translations for the United States and France:

      ```
      <LocaleNameData state="INSERTED">
          <DisplayName state="INSERTED">France</DisplayName>
          <EffectiveLocale state="INSERTED">en_US</EffectiveLocale>
          <LocaleName state="INSERTED">fr_FR</LocaleName>
        </LocaleNameData>
        <LocaleNameData state="INSERTED">
          <DisplayName state="INSERTED">La France</DisplayName>
          <EffectiveLocale state="INSERTED">fr_FR</EffectiveLocale>
          <LocaleName state="INSERTED">fr_FR</LocaleName>
        </LocaleNameData>
        <LocaleNameData state="INSERTED">
          <DisplayName state="INSERTED">Les Etats-Unis</DisplayName>
          <EffectiveLocale state="INSERTED">fr_FR</EffectiveLocale>
          <LocaleName state="INSERTED">en_US</LocaleName>
        </LocaleNameData>
      ```

   If your implementation supports other locales, follow the same pattern so that each supported locale has translations for each country name.

5. Add loading of the `LightWeightLookupList` to the minimal data load.

   a. Enter the following in a command window:

      ```
      sdk customize LightWeightLookupList.lst
      ```

   b. Open `projects\`*`project-name`*`\WEB-INF\scripts\LightWeightLookupList.lst` in a text editor.

   c. Add a line specifying the language_COUNTRY code:

      ```
      WEB-INF/xmldata/I18N/la_CO/LightWeightLookupList
      ```

      For example, to add French France:

      ```
      WEB-INF/xmldata/I18N/fr_FR/LightWeightLookupList
      ```

   d. Save and close `LightWeightLookupList.lst`.

6. Configure the `SearchConfigurationProperties.xml` file.
   a. Enter the following in a command window:

      ```
      sdk customize SearchConfigurationProperties.xml
      ```

   b. Open `projects\`*`project-name`*`\WEB-INF\properties\SearchConfigurationProperties.xml` in a text editor.
   c. Add the following section to the the <Locales> element. Replace ***la_CO*** with the appropriate language_COUNTRY code for the locale you are installing, for example, fr_FR for French France.

      ```
      <Locale id="la_CO"
      queryParserClass="com.comergent.api.appservices.search.queryParser.standard.CmgtQueryParser">
      <Analyzers>
      <Analyzer
      analyzerClass="com.comergent.api.appservices.search.analysis.CatalogSearchAnalyzer"
      description="CatalogAnalyzer" id="search"/>
      <Analyzer
      analyzerClass="com.comergent.api.appservices.search.analysis.CatalogSearchAnalyzer"
      description="CatalogAnalyzer" id="build"/>
      </Analyzers>
      <DictionaryFile file="CatalogDictionary.mappings"/>
      </Locale>
      ```

   d. Save and close the file.
7. Rebuild the project:

   ```
   sdk merge -clean
   ```

8. Load the database with the new locale information:

   ```
   sdk createDB
   sdk loadDB or sdk loadMatrixDB
   sdk createSegDB
   sdk loadSegDB or sdk loadSegMatrixDB
   ```

9. Build the deployable (.war file) image:

   ```
   sdk distWar
   ```

10. Deploy the .war file to your servlet container.
11. Navigate to the `sdk_home\dist\timestamp-WAR` directory, where *timestamp* has the form YYYYMMDD and is the date on which you issued the sdk distWar command. Rename the `prefs_dev.xml` file to `prefs.xml`, then copy it to the home directory of the user who is running the servlet container: `user_home/cmgt/debs/conf/` directory.
12. Restart your servlet container.

13. Verify that the installation succeeded:
    a. Navigate to your implementation home page. The URL is similar to:

       `http://<server>:<port>/Sterling/en/US/enterpriseMgr/matrix`

    b. Log in as an administrator user, then click **My Account**.

       The User Detail page displays.

    c. The Preferred Locale drop-down list in the User Locale panel should include the locales that you just installed.

    d. Choose a locale from the drop-down list, click **Save**, log out, and log back in.

       The administrator user's home page should display with localized text. You can now create users who use the new locales.

Localization Pack Installation Steps: Existing Implementation

These steps apply to an existing implementation of the Visual Modeler.

Adding support for a locale to an existing implementation requires that you enter SQL commands directly to modify and populate the Knowledgebase. To complete these steps, you must have access to a SQL client such as Microsoft SQL Server Management Studio Express or SQLPlus for Oracle.

Note: The file `LightWeightLookupList`, needs updating in the below scenarios:

1. While installing the L10n pack for a new locale, the file `\WEB-INF\xmldata\I18N\<locale code>\LightWeightLookupList`, state should be set to `INSERTED`.
2. While updating any entries for an exisiting locale, the state should be set to `MODIFIED`.

These steps assume that your implementation was installed using the SDK and that you have an existing release structure within which to work.

Before you begin, stop your servlet container instance.

1. Install the localization pack JAR file. For example:

   ```
   sdk install SterlingSellingSuite-Locale-DEBS-9.0-de-DE.jar
   ```

   This installs the localization pack in the `releases\debs-9.0` directory.

2. Modify the `web.xml` file:

   a. If you have not already customized your project's `web.xml` file for other locales, enter the following in a command window:

   ```
   sdk customize web.xml
   ```

   b. Open `projects\`*`project-name`*`\WEB-INF\web.xml` in a text editor.

   c. Copy the section that begins with the following:

   ```
    <!-- Start of English US mapping -->
   ```

   And ends with the following:

   ```
   <!-- End of English US mapping -->
   ```

   d. Paste the entire section after the English US mapping section.

   e. Modify the new section's comments to refer to the locale name you are adding. For example:

   ```
   <!-- Start of German Germany mapping -->
   ```

   f. Modify the new section’s file path references to refer to the locale name file path, such as /**de/DE**. For example:

   ```
   <servlet-mapping>
         <servlet-name>DispatchServlet</servlet-name>
         <url-pattern>/de/DE/catalog/*</url-pattern>
   </servlet-mapping>
   ```

3. Configure the `Internationalization.xml` file.

   a. If you have not already customized your project's `Internationalization.xml` file for other locales, enter the following in a command window:

      ```
      sdk customize Internationalization.xml
      ```

   b. Open `projects\`*`project-name`*`\templates\WEB-INF\properties\Internationalization.xml` in a text editor.

   c. Add the locale designation to the Presentation element's supportedLocales field. For example, for the German Germany locale:

      ```
      <supportedLocales controlType="text" runtimeDisplayed="true"
      ChangeOnlyAtBootTime="true" visible="true" boxsize="60"
      displayQuestion="Presentation Locales"
      displayOptions="en_US,en_US (English-United States),
      zh_TW,zh_TW (Chinese-Taiwan),fr_FR,fr_FR
      (French-France),fr_BE,fr_BE (French-Belgium),
      de_DE,de_DE (German-Germany)" defaultChoice="en_US"
      help="Supported presentation locales.">en_US,de_DE</supportedLocales>
      ```

   d. Add the following to the Languages element. Replace ***la*** with the language code and ***CO*** with the country code for your locale:

      ```
      <la visible="false">
      <defaultCountry controlType="text" runtimeDisplayed="true"
      ChangeOnlyAtBootTime="true" visible="false" boxsize="60"
      displayQuestion="URL for the Help Files" defaultChoice="US"
      help="This is the default country for a specific language">CO
      </defaultCountry>
      </la>
      ```

      For example, for the German Germany locale:

      ```
      <de visible="false">
      <defaultCountry controlType="text" runtimeDisplayed="true"
      ChangeOnlyAtBootTime="true" visible="false" boxsize="60"
      displayQuestion="URL for the Help Files" defaultChoice="DE"
      help="This is the default country for a specific language">DE
      </defaultCountry>
      </de>
      ```

4. Modify the `I18NLookup.lst` file:

   a. Enter the following in a command window:

      ```
      sdk customize I18NLookup.lst
      ```

   b. Open `projects\`*`project-name`*`\WEB-INF\xmldata\I18N\I18NLookup.lst` in a text editor.

c. Add a line specifying the `LightWeightLookupList` file to load for the locale you are adding:

   `WEB-INF/xmldata/I18N/la_CO/LightWeightLookupList`

   For example, to load the LightWeightLookupList file for German Germany:

   `WEB-INF/xmldata/I18N/de_DE/LightWeightLookupList`

5. Configure the `SearchConfigurationProperties.xml` file:

   a. If you have not already customized your `SearchConfigurationProperties.xml` file for other locales, enter the following in a command window:

```
sdk customize SearchConfigurationProperties.xml
```

   b. Open `projects\`*`project-name`*`\WEB-INF\properties\SearchConfigurationProperties.xml` in a text editor.

   c. Add the following section to the the <Locales> element. Replace ***la_CO*** with the appropriate language_COUNTRY code for the locale you are installing, for example, de_DE for German Germany.

```
<Locale id="de_DE"
queryParserClass="com.comergent.api.appservices.search.queryParser.standard.CmgtQueryParser">
<Analyzers>
<Analyzer
analyzerClass="com.comergent.api.appservices.search.analysis.CatalogSearchAnalyzer"
description="CatalogAnalyzer" id="search"/>
<Analyzer
analyzerClass="com.comergent.api.appservices.search.analysis.CatalogSearchAnalyzer"
description="CatalogAnalyzer" id="build"/>
</Analyzers>
<DictionaryFile file="CatalogDictionary.mappings"/>
</Locale>
```

   d. Save and close the file.

6. Set the location of the XML loader script and prefs.xml file:

   a. Enter the following in a command window:

```
sdk customize loadI18NFromXML.bat (For Windows systems)
or:
sdk customize loadI18NFromXML.sh (For Unix systems)
```

   b. Open `\projects\`*`project_name`*`\WEB-INF\scripts\loadI18NFromXML.bat` or `.sh` in a text editor.

   c. Search for the line containing:

```
set LOADER_JAR=%DEBS_RELEASE_DIR%/cmgt-xmlloader-tool.jar (For Windows systems)
LOADER_JAR=$DEBS_RELEASE_DIR/cmgt-xmlloader-tool.jar (For Unix systems)
```

d. Replace %DEBS_RELEASE_DIR% (Windows) or $DEBS_RELEASE (Unix) with the full pathname of your project's `image/data` directory location. Ensure that the directories are separated by forward slashes. For example: `C:/SDK351/releases/debs-9.0/image/data` (Windows)
or `/debs/sdk351/releases/debs-9.0/image/data` (Unix).

e. Specify the location of the `prefs.xml` preferences store file. Search for the line containing:

```
-DDataServices.General.ServerId=1
```

f. Add the following parameter to the JAVA_OPTS parameter list:

```
-Dcomergent.preferences.store="prefs.xml_full_pathname"
```

Where *prefs.xml_full_pathname* is the location of the `prefs.xml` file in your project. For example, if your project name is matrix and your sdk_home is `/debs/sdk351`, the command is as follows:

```
$JAVA $JAVA_OPTS -DDataServices.General.ServerId=1
-Dcomergent.preferences.store="debs/sdk351/projects/matrix/prefs.xml"
-classpath $CP ${MAIN} dummy persist PARTNER_NAME=matrix $PHASE1_LIST
MODE=QUIET
```

g. Save and close the file.

7. Rebuild the project:

```
sdk merge -clean
```

8. Load the database with the new locale information:

a. Use your SQL client tool to connect to the existing knowlegebase. For connection information, consult the values in the `projects\`*`project_name`*`\project_name-dev.properties` file.

b. Update the CMGT_LOCALE table entry. In your SQL client tool, run the following SQL command:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY,LOCALE_NAME,LOCALE_DESCRIPTION,ACTIVE_FLAG,DB_SORT_LOCALE_NA
ME )
VALUES ( key,'la_CO','Country Language','Y','BINARY')
```

- *Key* is the locale key. The locale key must be a unique numeric value.
- *la_CO* is the language code and country code. Use the appropriate language_COUNTRY encoding for the locale you are installing, for example, de_DE for German Germany, fr_FR for French France, jp_JP for Japanese Japan, and so on.

For example, the following INSERT statement is for supporting the de_DE (German Germany) locale:

```
INSERT INTO CMGT_LOCALE
(LOCALE_KEY,LOCALE_NAME,LOCALE_DESCRIPTION,ACTIVE_FLAG,DB_SORT_LOCALE_NA
ME )
VALUES ( 3,'de_DE','Germany German','Y','BINARY')
```

To check that your INSERT into the CMGT_LOCALE table is correct, run the following SQL statement:

```
select * from CMGT_LOCALE
```

9. Add the country name translations:
   a. Update the CMGT_LOCALE_NAME Table. In your SQL client tool, run the following SQL commands:

      ```
      insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
      DISPLAY_NAME, ACTIVE_FLAG) values ('la_CO', 'la_CO',
      'Locale_Country_Name', 'Y')
      ```

      ```
      insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
      DISPLAY_NAME, ACTIVE_FLAG) values ('la_CO', 'defaultLA_defaultCO',
      'default_Country_Name', 'Y')
      ```

      ```
      insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
      DISPLAY_NAME, ACTIVE_FLAG) values ('defaultLA_defaultCO', 'la_CO',
      'Country', 'Y')
      ```

      Where:

      - *la_CO* is the language_COUNTRY combination for the locale you are adding, such as de_DE
      - *Locale_Country_Name* is the name of the country in the locale's language, such as Deutschland
      - *defaultLA_defaultCO* is the language_COUNTRY combination for the default locale, such as en_US
      - *default_Country_Name* is the name of the default country in the language of the locale that you are adding, such as Vereinigte Staaten
      - *Country* is the name of the country in the default locale's language, such as Germany

      For example, to add country name translations for German:

      ```
      insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
      DISPLAY_NAME, ACTIVE_FLAG) values ('de_DE', 'de_DE', 'Deutschland', 'Y')
      ```

      ```
      insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
      DISPLAY_NAME, ACTIVE_FLAG) values ('de_DE', 'en_US', 'Vereinigte
      Staaten', 'Y')
      ```

      ```
      insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
      DISPLAY_NAME, ACTIVE_FLAG) values ('en_US', 'de_DE', 'Germany', 'Y')
      ```

   b. If there are several locales installed on your system, add a country name translation for the new locale to each of the existing locales, and add a country name translation for each existing locale to the new locale. For example, if you already support the fr_FR locale and are adding support for the de_DE locale, run the following SQL:

      ```
      insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
      DISPLAY_NAME, ACTIVE_FLAG) values ('fr_FR', 'de_DE', 'L''Allemagne',
      'Y')
      ```

      ```
      insert into CMGT_LOCALE_NAMES (EFFECTIVE_LOCALE, LOCALE_NAME,
      DISPLAY_NAME, ACTIVE_FLAG) values ('de_DE', 'fr_FR', 'Frankreich', 'Y')
      ```

      To check the results of the INSERT commands, run the following SQL command:

      ```
      select * from CMGT_LOCALE_NAMES
      ```

10. In a command window, navigate to your *sdk_home*\workspaces*project-name* directory and run the following command:

    `WEB-INF\scripts\loadI18NFromXML.bat I18N jdbc_driver (for Windows Systems)`

    or

    `WEB-INF/scripts/loadI18NFromXML.sh I18N jdbc_driver (for Unix Systems)`

    where `jdbc_driver` is the full pathname of your JDBC jar file.

    For example:

    `WEB-INF\scripts\loadI18NFromXML.bat I18N oraclejdbc.jar`

11. In a command window, navigate to your *sdk_home* directory and build the deployable (.war file) image:

    `sdk distWar`

12. Deploy the .war file to your servlet container.
13. Navigate to the `projects\`*project_name*`\dist\`*timestamp*`-WAR` directory, where *timestamp* has the form YYYYMMDD and is the date on which you issued the sdk distWar command. Rename the `prefs_dev.xml` file to `prefs.xml`, then copy it to the home directory of the user who is running the servlet container: *user_home*`/cmgt/debs/conf/` directory.
14. Restart your servlet container.
15. Verify that the installation succeeded:
    a. Navigate to your implementation home page. The URL is similar to:

       `http://<server>:<port>/Sterling/en/US/enterpriseMgr/matrix`

    b. Log in as an administrator user, then click My Account.

       The User Detail page displays.

    c. The **Preferred Locale** drop-down list in the **User Locale** panel should include the locales that you just installed.

    d. Choose a locale from the drop-down list, click **Save**, log out, and log back in.

       The administrator user's home page displays with localized text. You can now create users who use the new locales.

Starting the Visual Modeler Server

In general, you can start the Visual Modeler by starting the servlet container in which the Visual Modeler is installed. The order in which the servlets load is specified in the Visual Modeler Web application `web.xml` file and you can read this file in any text editor.

As the Visual Modeler starts, the servlet console window displays preliminary logging information. Once the Visual Modeler has initialized its logging environment, then it uses the logging methods to record events.

Troubleshooting

This section covers some basic steps that you must perform to ensure that the system starts correctly. This list is not comprehensive; rather it covers some check points that are a common source of problems. In general, you should troubleshoot your installation using the SDK to ensure that any modifications you make are contained in your project directory.

To Perform Pre-startup Checks

1. Review the `prefs.xml` configuration file. Check that it is in the correct location as this is the most frequent cause of problems on startup. Remember:
   a. By default, the location of this file is assumed to be `user_home/cmgt/debs/conf/` where `user_home` is the home directory for the operating system user running the servlet container.
   b. This location can be overridden by:
   - Either: specifying the location of the file as a system property:

     ```
     -Dcomergent.preferences.store=/path/prefs.xml
     ```
   - Or: specifying its location using the comergent.preference.store parameter in the Visual Modeler `web.xml` configuration file:

     ```
     <init-param>
     <param-name>comergent.preferences.store</param-name>
     <param-value>/path/prefs.xml</param-value>
     </init-param>
     ```
2. Review the `Comergent.xml` configuration file. Check that:
   - It contains the value of system properties that you expect to see (or that are overridden by the `prefs.xml` configuration file).
3. Using the SDK, run the generateDTD target.
   - If you get a series of lines of the form: "Writing DTD for ACL...done!", then the DTDs have been successfully generated. Look in the `debs_home/Sterling/WEB-INF/bizobjs/` directory to verify that a complete set of DTDs are there.
   - If you get an error message, then review the steps outlined above.
4. Using the SDK, run the generateBean target. This should generate all the beans specified by the data objects. If you see any error messages, then you should fix their cause before proceeding.
5. Using the SDK, run the merge target. If this runs successfully, then run the dist target to generate the Web application WAR file.

Error Messages on Startup

When the Visual Modeler starts, you can see initialization information in either the console window or the servlet container log file. Refer to the topic "Troubleshooting" for a summary of the most likely error messages, together with their causes and how to resolve them.

To troubleshoot problems with message types, you can set the messageTypeValidate element in the `Comergent.xml` file to “TRUE”.

Runtime Troubleshooting

This section covers some problems identified during testing.

Problem	Solution

On Solaris, the servlet container cannot find a certain servlet or URL.	First make sure that you did not make a typo. If you are certain that there was no mistake, then do the following:
	1. Run the following command on `web.xml`:
	`java com.comergent.dcm.util.CheckWebXML web.xml > newWeb.xml`
	2. Edit the file `newWeb.xml`. Look for the following string
	`<!-- (8192) XXX BOUNDARY BREAK -->`
	The start of the comment <!-- is the start of a 8192 boundary break. If it falls within a value for an XML node, then that node will get truncated.
	A work around is to pad the `web.xml` file such that the boundary break will fall inside a comment. For more information, see the comments at the start of file `CheckWebXML.java`.

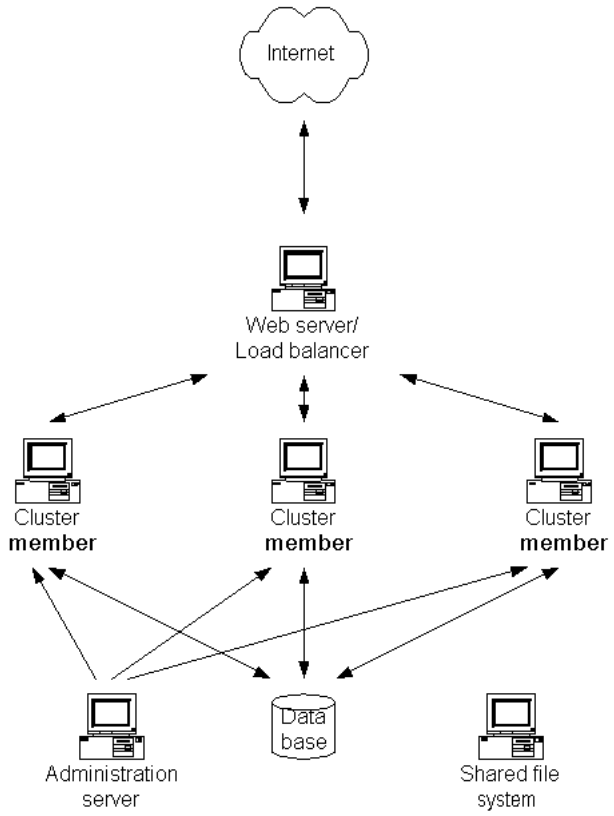
You see parser errors such as: `java.lang.NoSuchMethodError at org.apache.xpath.DOM2Helper.getNamespaceOfNode (DOM2Helper.java:348) at org.apache.xml.utils.TreeWalker.startNode (TreeWalker.java:281) at org.apache.xml.utils.TreeWalker.traverse (TreeWalker.java:119) at org.apache.xalan.transformer.TransformerIdentityImpl.transform (TransformerIdentityImpl.java:320)`	Check that you have followed the instructions to copy the XML parser-related JAR files to the servlet container’s `lib/` directory, and that you have removed any default `parser.jar` files.
Running iPlanet, you see the following in your browser: `GX Error (GX2GX) socket result code missing!!!`	There is a mismatch between the `web.xml` and `ias-web.xml` files. All servlets mentioned in `web.xml` must have a corresponding entry in the `ias-web.xml` file. Use the kguidgen utility to generate a GUID for the servlet.

Installing a Clustered Implementation

A cluster provides an environment that supports higher performance and reliability than a single machine can. Typically, a cluster comprises two or more member machines that from the outside world appear to work as one machine: when users submit a request to the cluster URL, they are not aware of which machine in the cluster processes the request and returns the response.

The cluster URL is usually directed to a Web server that sits “in front” of the cluster: this Web server provides the entry point for users, and it is responsible for distributing the requests to cluster members as requests come in. The Web server acts as a load balancer and distributes requests using an algorithm to determine which cluster member machine should receive each inbound request.

The following figure illustrates the general cluster configuration.



Administration Servers

In some cluster configurations, each cluster member is effectively independent of the others: you install the Visual Modeler into each cluster member and configure it independently of the other members of the cluster. Other cluster configurations make use of an administration server: this is a machine that manages the cluster. Cluster members are typically registered with the administration server and the administration server maintains a single image of the Visual Modeler. When a machine joins the cluster, the administration

server pushes a copy of the Web application to the new cluster member. In this case, each cluster member has the same configuration information because it has been pushed to them from the administration server.

The Visual Modeler uses Ehcache to provide the notification mechanism required to synchronize cluster members.

Shared Files

To ensure that cluster members behave consistently with each other, they must access configuration files, templates, and image files that are common to all members of the cluster. You do this by establishing a shared file server and point to a common location on this file server.

- On UNIX systems, use an NFS file system to share common files. For example:

```
<context-param>
   <param-name>WritableDirectory.share.public.loadable
   </param-name>
   <param-value>/usr/Comergent/shared</param-value>
</context-param>
```

- On Windows systems, use one of two methods to set up a shared file server.
  - Using one method, on each cluster member you map the same drive letter to the shared file server, then use the drive letter to provide a common reference to the location of the shared files. For example:

    ```
    <context-param>
    <param-name>WritableDirectory.share.public.loadable
    </param-name>
    <param-value>T:/Comergent/shared</param-value>
    </context-param>
    ```

    Here, the T: drive on each machine has been mapped to the C: drive on the file server machine.
  - Or, using the other method, use the UNC convention to refer to the shared directory location. For example:

    ```
    <context-param>
    <param-name>WritableDirectory.share.public.loadable
    </param-name>
    <param-value>\\fileserver\Comergent\shared</param-value>
    </context-param>
    ```

Load Balancer

If you run your cluster using a load-balancing solution (either a hardware- or software-based solution), then make sure that the load-balancing is done in a session-sticky fashion. That is, all requests relating to a session should be handled by the same member machine in the cluster.

General Installation Instructions for Clustered Deployment

1. Depending on the cluster architecture, install the Visual Modeler on each instance or into the Administrator server that deploys the Web application to the managed servers.

2. If you are using SQL Server as the Knowledgebase database server, then make sure that you set the ServerId system property and element of the `DataServices.xml` file to a unique two-digit value on each machine that makes up the cluster. This ensures that generated keys are managed correctly. See "Support for SQL Server" for more information.
3. If you are using 2-way encryption anywhere in the implementation, then follow these steps:
   a. Make sure that you start one of the machines before the others.
   b. Perform a persist operation that requires the use of 2-way encryption.
   c. Identify the location of the `dcmsKey.ser` file on this machine and copy this file to the corresponding location on the other machines of the cluster.
4. Follow the steps described in the topic "Sharing Directories".
5. As a site administrator, set the value of the useSessionCaching system property to “true”. This property is in the Profile Manager section of the system properties.
6. Enable your Visual Modeler implementation as a distributed implementation as follows:
   a. As a site administrator, set the value of the GlobalCache: Implementation Class system property to com.comergent.globalcache.DistributedCache. This property is in the GlobalCache section of the system properties.

      This tells the Visual Modeler to use the Ehcache configuration file `WEB-INF\properties\DistributedCache-Config.xml`.
   b. Enable the DistributedEventService by uncommenting the RefreshServiceHelper listener code in the `WEB-INF/web.xml` configuration file:

```
<!-- Start of Listeners -->
  <listener>
    <listener-class>
     com.comergent.reference.appservices.cache.CacheManagersHelper
    </listener-class>
  </listener>
<!-- comment this out to allow preferences refresh event to propagate
     to other nodes -->
<!--
   <listener>
     <listener-class>
      com.comergent.reference.appservices.cache.RefreshServiceHelper
      </listener-class>
   </listener>
-->
   <listener>
     <listener-class>
       com.comergent.dcm.core.SessionMonitor
     </listener-class>
    </listener>
<!-- End of Listeners -->
```

c. As a site administrator, set the value of the cronRefreshTime property. The cronRefreshTime property specifies the polling interval, in seconds, at which a node should check for modified or added cron jobs. Set the value of this property in the Job Scheduler refresh time in seconds field of the Job Scheduler section of the system properties. The default value, -1, prevents the node from periodically checking for changes to cron jobs.

7. By default, distributed nodes are discovered automatically using the Ehcache configuration for both the GlobalCache and EventService. However, you can also modify the cacheManagerPeerProviderFactory property settings for multicastGroupAddress and multicastGroupPort in the `\WEB-INF\properties\DistributedGlobalCache-Config.xml` and `WEB-INF\classes\DistributedEventService-config.xml` files to specify the unique IP addresses and ports for a cluster to adjust the scoping of the discovery mechanism.

```
<cacheManagerPeerProviderFactory

class="net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory"
      properties="peerDiscovery=automatic,
                  multicastGroupAddress=230.0.0.1,
                  multicastGroupPort=4567, timeToLive=1" />
```

You can also modify the timeToLive property setting to restrict how far packets should go. The setting values are:

- 0 - the same host
- 1 - the same subnet
- 32 - the same site
- 64 - the same region
- 128 - the same continent
- 255 - unrestricted

The default timeToLive value is 1, the same subnet.

The GlobalCache and EventService configuration must be the same on each cluster node, and must be unique for each cluster. For example, if you have two separate clusters, each cluster's configuration must be consistent across that cluster's nodes. The clusters themselves must each have unique configurations so that they do not conflict.

8. Copy the `prefs.xml` configuration file to a shared location which is visible to all member machines of the cluster. The location of the file must be specified in the startup script for each cluster member as follows:

   ```
   -Dcomergent.preferences.store=<Path to prefs.xml>
   ```

9. Configure the cluster to check for new and updated files as soon as possible. This ensures that all servers are in sync and will serve the same information to customers accessing your site. This is especially important in ensuring that the latest generated product index file is available at all times.

   Place your configuration property XML files in a shared location accessible by all member machines of the cluster. Then, activate the AutoReload element of the `SearchConfigurationProperties.xml` configuration file as follows:

   ```
   <AutoReload activated="true" reloadFilePeriod="30"/>
   ```

   This activates the AutoReload function and instructs the cluster to check for updates every 30 seconds.

10. Follow any remaining steps required by your servlet container or load balancer to implement their specific solution. See:

    a. Setting Up a WebLogic Cluster

    b. Setting Up a WebSphere Cluster

    Contact your Sterling Commerce representative for information about setting up other clustering architectures.

Setting Up a WebLogic Cluster

You can use the clustering capabilities to set up a cluster of WebLogic servlet containers to run your implementation of the Visual Modeler. In general, you should follow the instructions provided by Oracle to set up the cluster. This section provides some additional information used to install the Visual Modeler in the cluster.

Web Server

We suggest that you set up the cluster by placing a Web server or separate WebLogic Server as a front-end to the servlet container cluster. You should choose one of these options:

1. Set up a Web server with the appropriate WebLogic Web server plug-in. Supported Web servers include Apache and Microsoft Internet Information Server

   Note: If you use Apache, ensure that your Apache release matches the mod_wl_20.so version. At this time of writing (October 2003), Apache 2.0.42 works with the current mod_wl_20.so provided by WebLogic.

2. Set up a WebLogic Server with the HttpClusterServlet Web application. The HttpClusterServlet maintains the list of all servers in the cluster, as well as the load balancing logic to use when accessing the cluster.

When the user's browser makes a request, the Web server or HttpClusterServlet proxies the request to the WebLogic Server cluster. See the WebLogic documentation for further details.

Administration and Managed Servers

Typically, a WebLogic cluster comprises an Administration Server and one or more Managed Servers. The Web applications are deployed into the Administration Server and then as Managed Servers start or join the cluster, the Administration Server deploys the Web applications to each Managed Server. Consequently, you must deploy the Visual Modeler Web application `Sterling.war` file into the Administration Server first.

Note that when a Managed Server restarts, the Administration Server redeploys the Web applications to the Managed Server: this can take a considerable time, and so you should restart servers at times that ensure that they can be offline for the time they need to restart.

Preparation to Deploy the Visual Modeler Web Application

Because the same WAR file is used to deploy to all cluster members, you must make sure that this WAR file is correctly configured before you deploy the WAR file to the Administration Server. In particular:

1. Make sure that you have used the SDK to build the deployment WAR file.

2. While using the SDK, make sure that the following configuration properties are correctly set:
    a. `web.xml`: Make sure that the WritableDirectory parameters are correctly set to point to the shared directory location. See "Common Directories" for more information. Make sure that you have declared the SharedPublicServlet class as described in "SharedPublicServlet Class".
    b. `weblogic.xml`: Make sure that you have added a `weblogic.xml` file to the `sdk_home/projects/project/WEB-INF/` directory. To support session-sharing across the cluster members, consider adding the element described in "Session Sharing".
    c. Make sure that you have correctly specified the database connection information in the appropriate properties file so that they are correctly set in the `prefs.xml` configuration file.
3. Build the `Sterling.war` file using the SDK distWar target.
4. Copy the `prefs.xml` configuration file to a shared location which is visible to all member machines of the cluster. The location of the file must be specified in the startup script for each cluster member as follows:

    ```
    -Dcomergent.preferences.store=<Path to prefs.xml>
    ```

Deploying the Visual Modeler Web Application

Follow these steps to deploy the Visual Modeler Web application into the cluster. These instructions assume that you have set up the cluster using the WebLogic administration console on the Administration Server: we refer to the name of the cluster as *cluster_name*. We also assume that the managed servers are up and running. Make sure that you have used the SDK to create the `Sterling.war` file and that you have moved a copy of the file to a location on the Administration Server.

1. Log into the administration console of the WebLogic Administration Server.
2. Click **Servers** and verify that the managed servers are listed.
3. Click **Clusters** and verify that the name of the target cluster is *cluster_name*.
4. Click **Lock & Edit**.
5. Click **Deployments**.
6. Click **Install**.
7. In the next window, navigate to the location of the `Sterling.war` file and select the radio button next to the **Sterling.war** file name.
8. Click **Next**.
9. Select the **Install this deployment as an application** radio button.
10. Click **Next**.
11. Check the check box next to the cluster named *cluster_name*. By default, the **All servers in the cluster** radio button is selected. You should usually leave this setting unchanged.
12. Click **Next**.
13. In the **Name** field of the General panel on the Optional Settings page, enter the name of your deployment, for example, Sterling. Accept the defaults for the other values on the Optional Settings page.
14. Click **Next** to review your choices, then click **Finish** to complete the deployment.

15. Click **Activate Changes** to activate the deployment.

    Deployment can take ten to twenty minutes. At the end of the deployment process, a page displays a Success message.

SQL Server

Because more than one deployment of the Visual Modeler is accessing the same Knowledgebase on SQL Server, you must set a two-digit server ID for each deployment. You must modify the servlet container command or script that starts the servlet container on each machine so that a Java system property is set: Comergent.DataServices.General.ServerId. This should be set on each machine so that each has a unique value: 01, 02, and so on.

For example, in a Tomcat installation, you can modify the starting batch file to include:

```
set JAVA_OPTS=-DComergent.DataServices.General.ServerId=12
```

Cron Jobs

The Visual Modeler distinguishes between system cron jobs and application cron jobs. Typically, system cron jobs are run without an associated user and run on every system in a clustered environment whereas application cron jobs must be run associated to a user and usually should be run only by one machine in a cluster.

To set this up, you must do the following:

1. Make sure that in the deployment WAR file, the value of the cronApps system configuration property is set to “system”.
2. For the one application server that should run application cron jobs, make sure that a system property is set as follows:

   ```
   -DComergent.Cron.cronApps=both
   ```

   For example, in a Tomcat installation, you can modify the starting batch file to include:

   ```
   set JAVA_OPTS=-DComergent.Cron.cronApps=both
   ```

   Note: How you do this will vary from one servlet container to another: see, for example, "Command Line Settings" for WebSphere instructions. Note that the valid values for this property are: “application”, “both”, “none”, and “system”.
3. Set the value of the Cron Job URL system property to the value of the URL used to access the cluster: for example:

   ```
   http://loadbalancer/Sterling/msg/matrix
   ```

Common Directories

All the Managed Servers in the cluster must be able to access the same directory locations in the file system: this is where configuration files, shared data files, and other related files such as pagination data is stored for the cluster. You must ensure that all members of the cluster access this location using the same directory paths.

The location of the shared directories is specified in the Visual Modeler `web.xml` file using context parameter elements of this form:

```
<context-param>
```

```
    <param-name>WritableDirectory.share.public.loadable</param-name>
    <param-value>/tmp</param-value>
</context-param>
<context-param>
    <param-name>WritableDirectory.share.public.noloadable
    </param-name>
    <param-value>/tmp</param-value>
</context-param>
<context-param>
    <param-name>WritableDirectory.share.private.loadable</param-name>
    <param-value>/tmp</param-value>
</context-param>
<context-param>
    <param-name>WritableDirectory.share.private.noloadable
    </param-name>
    <param-value>/tmp</param-value>
</context-param>
```

See "Shared Files" for the form that the values of these parameters can take. Note that by default, these elements are commented out: in this case, each instance of the Visual Modeler Web application acts independently of the other instances in the cluster. All file accesses are performed locally on the machine running the Web application.

The following table summarizes shared file locations:

Location	Purpose
share.public.loadable	Do not use.
share.public.noloadable	Image files and other files that should be accessible to Web servers to serve up static content. Examples include GIF files associated with promotions and storefront partners.
share.private.loadable	Class files to be shared across the cluster: this directory is used primarily for Sterling Configurator and Visual Modeler.
share.private.noloadable	Configuration files, pagination files, and other files that must be shared across the cluster, but which should not be accessible from users' browsers.

SharedPublicServlet Class

You must uncomment in the element that declares the SharedPublicServlet class: this class is used to serve up static content such as partner logos and promotion images that are uploaded to the Visual Modeler.

```
<servlet>
    <servlet-name>SharedPublicServlet</servlet-name>
    <servlet-class>
        com.comergent.dcm.core.SharedPublicServlet
    </servlet-class>
</servlet>
```

You must also uncomment in the following elements that map URLs to the SharedPublicServlet:

```
<servlet-mapping>
    <servlet-name>SharedPublicServlet</servlet-name>
    <url-pattern>/htdocs/partnerlogos/*</url-pattern>
```

```
</servlet-mapping>

<servlet-mapping>
   <servlet-name>SharedPublicServlet</servlet-name>
   <url-pattern>/htdocs/promotions/images/*</url-pattern>
</servlet-mapping>
```

For each supported locale, uncomment in the corresponding element:

```
<servlet-mapping>
   <servlet-name>SharedPublicServlet</servlet-name>
   <url-pattern>/la/CO/htdocs/*</url-pattern>
</servlet-mapping>
```

For example, uncomment in the following element for the en_US locale:

```
<servlet-mapping>
   <servlet-name>SharedPublicServlet</servlet-name>
   <url-pattern>/en/US/htdocs/*</url-pattern>
</servlet-mapping>
```

Session Sharing

You must also provide information about how sessions are to be shared across the cluster using the `weblogic.xml` deployment file. You may have already created this file to pass in information about the WebLogic environment or you may have to create it only for this purpose. It should be located in your `Sterling.war` Web application file at the same level as the `web.xml` file.

You must add the following fragment to the `weblogic.xml` file:

```
<session-descriptor>
   <session-param>
      <param-name>PersistentStoreType</param-name>
      <param-value>file</param-value>
   </session-param>
   <session-param>
      <param-name>PersistentStoreDir</param-name>
      <param-value>
         <Directory location common to all members of cluster>
      </param-value>
   </session-param>
</session-descriptor>
```

Note that a more common setting is:

```
<session-param>
   <param-name>PersistentStoreType</param-name>
   <param-value>memory</param-value>
</session-param>
```

This setting does not support session-failover.

Reloading Files

If shared configuration files can be updated, then each managed server may need to reload the shared copy to pick up changes made by other servers in the cluster. For example, the `SearchConfigurationProperties.xml` file has a setting:

```
<SearchSystemConfigurations>
   <AutoReload activated="true" reloadFilePeriod="30"/>
</SearchSystemConfigurations>
```

Set the activated attribute to “true” and set the reloadFilePeriod attribute to an interval (in seconds) to specify that if an interval of more than 30 seconds elapses between accesses, then the file should be reloaded.

Running a Clustered WebLogic Installation

In a clustered deployment of WebLogic, you must also perform these steps to ensure that the DTDs used by the Visual Modeler are correctly located. On each machine in the cluster:

1. Create or identify a designated directory that may be used to store the DTDs. For example, you can create a sub-directory called `container_home/local/working/` in each WebLogic installation.
2. Unjar the Visual Modeler WAR file, and copy the DTD files from their locations under `WEB-INF/` to the designated directory.
3. Modify `startManagedWebLogic.cmd` or `startManagedWebLogic.sh` to set a new runtime flag: `-DComergent.workingDir`. You can use the $WL_HOME variable if the designated directory lies under the `container_home/` location. For example:

   ```
   -DComergent.workingDir=$WL_HOME/local/working
   ```

Setting Up a WebSphere Cluster

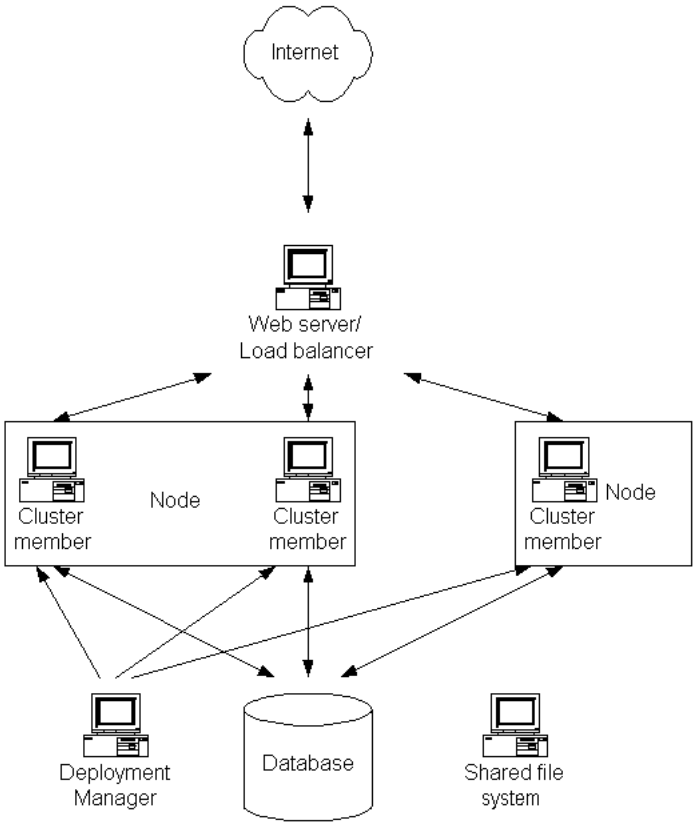
This section describes how to set up a WebSphere cluster and to deploy the Visual Modeler to the cluster.

Releases of the Visual Modeler before Release 6.4.1 have known issues with WebSphere and clustering: you should make sure that you have applied hot fixes to your release that ensure that the SDK builds the WAR file for WebSphere correctly, and so that configuration properties can be passed to the Visual Modeler by setting command line parameters.

A typical WebSphere cluster comprises:

- A deployment manager server
- two or more applications servers running on the same or different machines: these are the cluster member servers
- a load balancing server: this is the Web server to which users point their browsers and which distributes requests to the cluster member servers.

The following figure illustrates the WebSphere cluster configuration.



1. Set up the deployment manager on a machine in your network. It need not run on the same machine on any of the application servers, but the deployment manager and application servers should all be running on the same segment of a local area network. Start the deployment manager (either through the UI or at the command line in `websphere_home/AppServer/bin/`, enter: startServer dmgr).

   You use the deployment manager to manage the WebSphere cell: this is where the cluster is run.

2. Install your application servers on the cluster member machines. Each cluster member machine is referred to as a node. You can create one or more application servers on each node.
3. Start the node agent on each node: typically you do this at the command line, by navigating to `websphere_home/AppServer/bin/` and entering: startNode.
4. The node agent is what the deployment manager uses to communicate with each node in its cell.
5. Log in to the Deployment Manager administration UI.
6. Add nodes to the cell by clicking **System Administration > Nodes**, and clicking **Add Node** in the Nodes panel.
7. For each node, add the application servers running on the node to the cell by clicking **Servers > Application Servers**, and clicking **New** in the Application Servers panel. You must select each node in turn and specify a unique name for each server on the node.

   When you have completed creating all the application servers, then you must create the cluster.

8. In the Deployment Manager UI, click **Servers -> Clusters**, and then click **New** in the Server Cluster panel.
9. Enter a name for the cluster and then without add servers to the cluster complete the creation of the cluster.
10. Once the cluster is created, select the cluster form the list of clusters. On the Server Cluster page, click Cluster members and use the Cluster members page to add application servers to the cluster.

    When you have added all the application servers to the cluster, check that you can start and stop the cluster by clicking **Servers > Clusters**, check the check box for the new cluster, and click **Start**. Verify that the cluster starts without any errors, and then click **Stop**.

Building and Deploying the Visual Modeler Web Application

This topic describes how to build and deploy the Visual Modeler Web application.

Building the Visual Modeler Deployment WAR File

You must build the deployment WAR file using the SDK, and must make sure that the cluster settings have been made before you build the WAR file for deployment. Typically, this means that you must customize the `web.xml` and `Comergent.xml` configuration files to set the cluster settings and the location of the shared directories

Note: You must do this in the SDK because once the WAR file is deployed, WebSphere processes the configuration parameters by assigning them IDs. If the cluster settings are commented out, then they will be missed.

Releases prior to Release 6.4.1 should be built using the patched version of the SDK 2.0.4, and you must make sure that you applied any patches relating to your release: notably to support WebSphere and to allow for configuration properties to be set at the command line.

Deploying the Visual Modeler Web Application

Now that you have created the cluster and verified that the cluster can be stopped and started successfully, you are ready to deploy the Visual Modeler Web application to the cluster. Make sure that you have built the deployment WAR file as described in "Building the Visual Modeler Deployment WAR File".

1. Copy the deployment WAR file to the deployment manager machine.
2. Log in to the Deployment Manager UI.
3. Click **Applications > Install New Application**.
4. On the Preparing for the application installation page, select the **Server path** radio button, and click **Browse...** to where you copied the deployment WAR file. Select the radio button next to the WAR file. Click **OK**.
5. Specify a context root, typically of the form: `/Sterling`.
6. Click **Next**.
7. On the next page, click **Next**.
8. On the next page, click **Continue**.
9. On the Install New Application page, you can change the name of the application, or just click **Next**.
10. On the next page, select a virtual host for the Visual Modeler Web application and click **Next**.
11. On the next page, you must specify that the application is to be deployed to the cluster. You do this by selecting the cluster in the list box of Clusters and Servers, checking the Visual Modeler check box, and then click **Apply**.
12. Click **Next**.
13. On the next page, check the deployment details for the new application, and then click **Finish**.

14. After the Web application finished deploying, then click **Save** to save the details of the new deployment to the master configuration.

    When the WAR file is deployed to the cluster members, it is expanded into a directory `websphere_home/AppServer/installedApps/` sub-directory.

Configuration

You must do the following configuration steps before you can start the cluster and access the new deployment.

Updating the Web Server Plugin

The load-balancing Web server makes use of a configuration XML file to know which requests it receives should be forwarded to the cluster and which machines are in the cluster. This file is called `plugin-cfg.xml` and you must regenerate it after deploying the Web application to the cluster.

1. Log in to the Deployment Manager UI.
2. Click **Environment > Update Web Server Plugin**.
3. Click **OK**.

   The new version of the file is created on the Deployment Manager machine as `websphere_home/DeploymentManager/config/cells/plugin-cfg.xml`.
4. Copy this file to the appropriate location on the load-balancer Web server machine. The precise destination location depends on your Web server solution. For example, if you use the IBM HTTPServer that is part of the IBM WebSphere suite, then you must copy it to the `websphere_home/AppServer/config/cells/plugin-cfg.xml` location on the Web server machine.
5. Restart the Web server to pick up the changed configuration file.

Command Line Settings

All of the application servers in the cluster will deploy exactly the same form of the Visual Modeler Web application in them. Typically, there are some configuration settings that must be set at the application server level: for example, the for SQL Server ServerId setting and cron job settings that determine whether application cron jobs run on the cluster member. To set these:

1. Log in to the Deployment Manager UI.
2. Click **Application Servers**.
3. For each application server in the cluster, repeat these steps:
   a. On the Application Servers page, click the link to the application server.
   b. Under Additional Properties, click **Process Definition**.
   c. Under Additional Properties, click **Java Virtual Machine**.

d. In the Generic JVM arguments text field, enter the application server properties as appropriate. **Note**:You only have to do this for properties whose value must be different from that set in the deployed Web application.

   For example, to specify that application and system cron jobs should be run on this application server, enter:

   ```
   -DComergent.Cron.cronApps=both
   ```

   To specify a ServerId value, enter:

   ```
   -DDataServices.General.ServerId=12
   ```

e. Click **OK**.

4. Once you have completed this step for all members of the cluster, then Save these changes to the master configuration.
5. Stop and then restart the cluster.

Testing the WebSphere Cluster

You can test that the cluster is working correctly simply by pointing your browser to the load-balancing Web server, and verify that you can access the Visual Modeler Web application using the context name that you specified when you deployed the Web application as described in "Deploying the Visual Modeler Web Application". For example:

```
http://loadbalancer/Sterling/en/US/enterpriseMgr/matrix
```

By logging in simultaneously from different browsers, and by examining the application server logs on each application server, you should be able to verify that the browser requests are being distributed among the application servers by the load-balancer. You should also be able to verify that if an application server goes down, that the load-balancer stops routing requests to that application server.

Setting up a Database for Caching

This implementation of the distributed Global Cache uses the Knowledgebase database server to store session information. Note that, only implementations that use the Oracle database server are supported.

1. Log in to the Visual Modeler system administration site as a site administrator.

   Your system administration site URL is similar to:

   `http://server:port/Sterling/en/US/enterpriseMgr/admin`
2. Click **System Services**.
3. Click **Commerce Manager**.
4. In the GlobalCache: Class Name property field, enter:

   `com.comergent.dcm.cache.impl.db.DBCache`
5. Click **Save All and return to List**.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual

Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing

Legal and Intellectual Property Law

IBM Japan Ltd.

1623-14, Shimotsuruma, Yamato-shi

Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS

FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

J46A/G4

555 Bailey Avenue

San Jose, CA__95141-1003

U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available. This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are

ficticious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© IBM 2011. Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. 2011.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium and the Ultrium Logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Connect Control Center®, Connect:Direct®, Connect:Enterprise, Gentran®, Gentran:Basic®, Gentran:Control®, Gentran:Director®, Gentran:Plus®, Gentran:Realtime®, Gentran:Server®, Gentran:Viewpoint®, Sterling Commerce™, Sterling Information Broker®, and Sterling Integrator® are trademarks or registered trademarks of Sterling Commerce, Inc., an IBM Company.

Other company, product, and service names may be trademarks or service marks of others.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

W

X