NCIA – National Cancer Imaging Archive

Installation Guide

Release 2.4

TABLE OF CONTENTS

1. INTRODUCTION

The purpose of this document is to document the installation process for Release 2.4 of the National Cancer Imaging Archive.

1.1 NCIA BACKGROUND

NCIA is a searchable repository of in vivo cancer images that provides the cancer research community, industry, and academia with access to image archives to be used in the development and validation of analytical software tools that support:

- Lesion detection and classification
- Accelerated diagnostic imaging decision
- Quantitative imaging assessment of drug response

NCIA provides access to imaging resources that will improve the use of imaging in today's cancer research and practice by:

- Increasing the efficiency and reproducibility of imaging cancer detection and diagnosis
- Leveraging imaging to provide an objective assessment of therapeutic response
- Ultimately enabling the development of imaging resources that will lead to improved clinical decision support.

Clinical trials submit images to NCIA so that they can be made available to others. Researchers may then access NCIA to search for and download images. Images are stored in the Digital Imaging and Communications in Medicine (DICOM) standard. A DICOM file stores the digital image along with a series of tags that contain metadata about the image such as patient ID, study ID, patient weight, anatomic site, and many more.

NCIA also allows querying across multiple instances of NCIA using caGRID.

1.2 SYSTEM TECHNICAL OVERVIEW

See Appendix A for a high level technical overview of the NCIA architecture.

1.3 HOW TO USE THE INSTALL GUIDE

Although every effort was made to simply these instructions as much as possible, we feel that this guide is best suited for an experienced developer who is familiar with J2EE and open source technologies.

NCIA was developed to be flexible to be used on a variety on platforms and to be modified by other developers. However, NCIA has not been tested with every possible configuration. See the system requirements in section 2.1 for details on the software that NCIA has been tested with.

Be careful when cutting and pasting text from this document to configuration files. Often, it works better to paste to Notepad first and then to the configuration file.

1.4 WHERE TO FIND MORE INFORMATION

1.4.1 Web Resources

Topic	URL
NCIA General Information and News	http://ncia.nci.nih.gov
NCIA Production Application	https://imaging.nci.nih.gov
NCIA Download (including source code)	http://ncia.nci.nih.gov/ncia/download
Common Security Module (CSM)	http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/csm
caCORE SDK	http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview
MIRC (Medical Imaging Resource Center)	http://www.rsna.org/mirc
Globus Toolkit	http://globus.org
caGRID	https://cabig.nci.nih.gov/workspaces/Architecture/caGrid

1.4.2 Support

If you need assistance with NCIA, please contact NCICB Application Support. Details of how to contact that organization by email or phone are available at the following URL:

http://ncicb.nci.nih.gov/NCICB/support

1.5 NCIA DISTRIBUTION ZIP FILE

All of the files for NCIA Release 2.4 have been placed in a single zip file for download. To obtain the zip file, go to http://ncia.nci.nih.gov/ncia/download. The high level directories under the zip file are as follows:

- Documentation – Documents that describe the installation process along with a user's guide.
- NCIA – Files needed to build and deploy the NCIA application
- NCIA Grid – Files needed to build and deploy the NCIA grid components

- MIRC – Files needed to build and deploy MIRC with NCIA's modifications. Also includes MIRC Field Center software
- SampleNCIADatabase – Oracle export of a sample NCIA database. See section 3.5 for details. The use of the NCIA sample database is optional.

1.6 SECURITY CONFIGURATION

For information on how to configure NCIA security using the Common Security Module (CSM) and the User Provisioning Tool (UPT), please refer to the NCIA CSM Configuration guide found in the Documentation directory of the download package.

2. SYSTEM REQUIREMENTS

2.1 SYSTEM REQUIREMENTS - SOFTWARE

This section describes the third party software that must be installed to use the NCIA software.

2.1.1 Java 2 Standard Edition 5.0

NCIA and MIRC both use Java 2 Standard Edition (J2SE) Development Kit 5.0. This can be downloaded from http://java.sun.com/j2se/1.5.0/download.jsp. After installing Java, you should also set the `JAVA_HOME` environment variable to the directory where you installed Java, and you should add the "`JAVA_HOME/bin`" directory to your PATH environment variable.

2.1.2 J2EE Application Server

NCIA uses Java 2 Enterprise Edition (J2EE) technology and requires a J2EE application server. NCIA was tested using JBoss 4.0.4. The NCIA team strongly recommends using JBoss 4.0.4, which can be downloaded from http://labs.jboss.com/portal/jbossas/download/index.html.

If you are using JBoss 4.0.4, two files must be deleted from the `jbossweb-tomcat55.sar/jsf-libs` directory – specifically the files "myfaces-impl.jar" and myfaces-api.jar" – in order to prevent JBoss's JSF implementation from interfering with NCIA's use of JSF.

On a UNIX platform, jsvc should be used to install JBoss as a service. On a Windows platform, a tool such as the Java Service Wrapper should be used to install JBoss as a service. Additional information can be found at the following links:

http://jboss.org/wiki/Wiki.jsp?page=RunJBossAsAServiceOnWindows

http://sourceforge.net/projects/wrapper/

Another application server such as WebLogic or WebSphere can be used, but will require work on the part of the implementer. Most of this work will be related to configuration of the application with deployment descriptors.

2.1.3 Apache Tomcat 5.5

The Apache Tomcat 5.5 servlet container is required for MIRC T29a. It can be downloaded from http://tomcat.apache.org/download-55.cgi.

On a UNIX platform, jsvc should be used to install Tomcat as a service. On a Windows platform, use of the MSI Installer will result in Tomcat being installed as a service.

2.1.4 Java Image I/O API 1.0

This module improves performance of image I/O and is used by MIRC. It can be downloaded from http://java.sun.com/products/java-media/jai/downloads/download-iio-1_0_01.html. You should download the CLASSPATH install version of the Image I/O API. To make this library available to all applications running in Tomcat (and specifically MIRC), simply extract the "`jai_imageio.jar`" file and place it into your Tomcat instances "`server/lib`" directory.

2.1.5 Globus Tookit 4.0.2

To run the NCIA grid components, the Globus Tookit version 4.0.2 must be installed. This software can be downloaded from

http://www-unix.globus.org/toolkit/survey/index.php?download=ws-core-4.0.2-bin.zip

After installing, set the GLOBUS_LOCATION environment variable to the location where Globus is installed.

To start Globus, run `globus-start-container –nosec` from Globus's `bin` directory.

2.1.6 Apache Ant 1.6.2

The Apache Ant tool version 1.6.2 is used to build the NCIA software. It can be downloaded from http://ant.apache.org/bindownload.cgi. After installing Ant, you should also set the `ANT_HOME` environment variable to the directory where you installed Ant, and you should add the "`ANT_HOME/bin`" directory to your PATH environment variable.

2.1.7 Relational Database

NCIA stores image information and other data in a relational database. NCIA was primarily tested using Oracle 9.2. It may also work with MySQL 5.0 and SQL Server 2005.

A database other than Oracle or MySQL or SQL Server can be used as long as it supports JDBC, but will require work on the part of the implementer. Appendix B provides some guidance on using NCIA with a database other than Oracle or MySQL.

It should be noted that in general, at runtime, there are two critical files for determining the ability of the application to access the database properly. The first file is the datasource XML file, located in the JBoss instances "server/default/deploy" directory – usually named something like "*databaseName*-ds.xml". The second file is the login configuration XML file, located in the JBoss instances "server/default/conf" directory, named "login-config.xml". Both files must point to your database instance using the correct dialect in order to access the database properly.

2.1.8 FTP Server

Files larger than a certain threshold are made available for FTP download by users. An FTP server must be installed to provide this functionality. NCIA was tested using `vsftp`, which can

be downloaded from http://vsftpd.beasts.org. See section **Error! Reference source not found.** for more information on how to configure NCIA to work with the FTP server.

2.1.9 Enterprise Architect (Optional)

NCIA uses the caCore SDK to access most of its domain classes. caCore requires a UML model stored in XMI format. Enterprise Architect (http://www.sparxsystems.com/) is the recommended tool to model and export to XMI. The Enterprise Architect model for NCIA is stored in `NCIA/toolkit/ncia.EAP`.

A modeling tool is only required if changes will be made to the domain classes.

2.2 SYSTEM REQUIREMENTS – HARDWARE

The hardware required for running NCIA depends on lots of factors, including the number of images planned for the repository and the required performance. In general, hardware guidelines provided by the application server and database vendors should be followed.

Image files are stored in the file system under the Tomcat installation. Because image files are very large, a large amount of disk space should be planned to support the desired number of images.

2.3 TYPICAL CONFIGURATION OF SOFTWARE ON HARDWARE

In the past, NCIA was tested using two separate machines: one as the database server and one for everything else (MIRC running on Tomcat, file system for images, NCIA running on JBoss, FTP Server).

More recently, NCIA has been tested in a configuration where each major component is deployed on a separate machine. Under this configuration, all components have access to the image file system.

3. NCIA INSTALLATION

3.1 BUILDING THE NCIA APPLICATION

3.1.1.1 Configuring deploy.properties

The following properties need to be set in the `NCIA/conf/deploy.properties` prior to running the build. The table below describes the properties that need to be set. Please note the recommended practice of using "/" as a path separator character regardless of your OS platform.

Property	Description	Example
`ftp_url`	When users request files over a certain size, the file is placed on a FTP server and the user receives an email with a link to the ftp server. This property specifies the URL to include in the email for your FTP server.	`ftp.yoururl.org` or `localhost`
`ftp_location`	Directory where NCIA will place files that are ready for download via FTP. Your FTP server should be set up to allow access to this directory.	`/local/content/ncia-ftp` or `C:/ncia-ftp`
`server_url`	The hostname and port that the application server is running on. This is used to provide access to the services of the caCore toolkit to the `ImageZippingMDB` and if desired to other applications.	`ncia.yoururl.com:59080`
`quarantine_directory`	Location of the MIRC quarantine directory for this server. Go ahead and create this empty directory on your filesystem.	`/data/trial/quarantine` or `C:/quarantine`
`jboss_mq_url`	Hostname and port number for the JMS service	`localhost:51099`
`image_server_url`	The external URL for the NCIA instance. This URL is used to display image thumbnails locally and over caGRID.	`http://imaging.yoururl.com` or `http://localhost`
`Csm_hibernate_location`	The location of the ncia.csm.hibernate.cfg.xml file in your JBoss instance	`/jboss/server/default/conf/ncia.csm.hibernate.cfg.xml`

3.1.1.2 Configuring hibernate.properties

Based on the database you choose to use, you will need to configure the appropriate entries in the "hibernate.properties" file, which resides in the "NCIA/conf" directory. Below are sample configurations for Oracle, MySQL, and SQL Server. Make sure that only **ONE** configuration is uncommented – and either comment out or delete the other configurations.

```
## Oracle

#hibernate.dialect org.hibernate.dialect.OracleDialect

#If the version of Oracle is 9.x uncomment the following and
comment the above by a '#'

hibernate.dialect org.hibernate.dialect.Oracle9Dialect

hibernate.connection.driver_class
oracle.jdbc.driver.OracleDriver

hibernate.connection.datasource java:/ncia


## MYSQL

hibernate.dialect org.hibernate.dialect.MySQLDialect

hibernate.connection.driver_class com.mysql.jdbc.Driver

hibernate.connection.datasource java:/ncia


## SQL Server 2005

hibernate.dialect org.hibernate.dialect.SQLServerDialect

hibernate.connection.driver_class
com.microsoft.sqlserver.jdbc.SQLServerDriver

hibernate.connection.datasource java:/ncia
```

3.1.1.3 Configuring cacoretoolkit.hibernate.properties.template

Based on the database you choose to use, you will need to configure the appropriate entries in the "cacoretoolkit.hibernate.properties.template" files, which reside in two locations, first in the "NCIA/conf" directory, and second in the "NCIA/conf/templates" directory. Make sure you make the changes in **BOTH** locations. Below are sample configurations for Oracle, MySQL, and SQL Server. Make sure that only **ONE** configuration is uncommented – and either comment out or delete the other configurations.

```
## Oracle

#hibernate.dialect org.hibernate.dialect.OracleDialect

#If the version of Oracle is 9.x uncomment the following and
comment the above by a '#'

hibernate.dialect org.hibernate.dialect.Oracle9Dialect

hibernate.connection.driver_class
oracle.jdbc.driver.OracleDriver

hibernate.connection.datasource java:/ncia


## MYSQL

hibernate.dialect org.hibernate.dialect.MySQLDialect

hibernate.connection.driver_class com.mysql.jdbc.Driver

hibernate.connection.datasource java:/ncia


## SQL Server 2005

hibernate.dialect org.hibernate.dialect.SQLServerDialect

hibernate.connection.driver_class
com.microsoft.sqlserver.jdbc.SQLServerDriver

hibernate.connection.datasource java:/ncia
```

3.1.1.4 Configuring ncia.hibernate.cfg.xml

Based on the database you choose to use, you will need to configure the appropriate dialect in the "ncia.hibernate.cfg.xml" files, which resides in the "NCIA/conf" directory. Below are sample configurations for Oracle, MySQL, and SQL Server. Make sure that only **ONE** configuration is uncommented – and either comment out or delete the other configurations.

```
<!-- Oracle Dialect -->

          <property
name="dialect">org.hibernate.dialect.OracleDialect</property>

<!-- MySQL Dialect -->

          <property
name="dialect">org.hibernate.dialect.MySQLDialect</property>
```

```
<!-- SQL Server Dialect -->

          <property
name="dialect">org.hibernate.dialect.SQLServerDialect</property>
```

3.1.1.5 Extracting, Configuring, and Re-archiving nciaservice.war

The NCIA download comes with a preconfigured nciaservice.war file, which is only preconfigured for Oracle as the database. If you are using MySQL or SQL Server, then you will need to perform this step.

This step involves extracting the .war file, making changes to two files, and then re-archiving the .war file. The two files which must be updated are "hibernate.properties", located in the "WEB-INF/conf" directory of the extracted archive, and the "orm1.cfg.xml" file, located in the "WEB-INF/classes" directory of the extracted archive. Both files need to be edited to use the proper database settings including the proper dialect class.

The following steps will allow you to extract the archive, configure the necessary changes, and re-archive the .war file:

1. Extract the "nciaservice.war" file, which is located in the "NCIA/toolkit" directory
    a. Execute the following commands from a shell/command prompt:

```
cd NCIA/toolkit

mkdir nciaservice_war

jar xvf nciaservice_war nciaservice.war
```

2. Edit the file "hibernate.properties", which is located in the "nciaservice_war/WEB-INF/conf" directory. Make sure that the dialect, driver_class and datasource are set up properly. (Alternatively you can simply copy your already-configured "NCIA/conf/hibernate.properties" file into the proper directory).

3. Edit the file "orm1.cfg.xml", which is located in the "nciaservice_war/WEB-INF/classes" directory. Make sure that the dialect property is set properly based on the database you have chosen to use, which will be one of the following settings:

```
<!-- Oracle Dialect -->

          <property
name="dialect">org.hibernate.dialect.OracleDialect</pr
operty>

<!-- MySQL Dialect -->

          <property
name="dialect">org.hibernate.dialect.MySQLDialect</pro
perty>
```

```
<!-- SQL Server Dialect -->

          <property
name="dialect">org.hibernate.dialect.SQLServerDialect<
/property>
```

4. Re-archive the .war file.

    a. Execute the following commands from a shell/command prompt:

```
cd NCIA/toolkit/nciaservice_war

jar cvf0 nciaservice.war *

copy nciaservice.war ..
```

You will now have a properly configured nciaservice.war file, which is needed during the NCIA application build process.

3.1.1.6 Configuring your JDBC library

Your JBDC Driver library (such as "ojdbc14.jar" for Oracle 9.x, "mysql-connector-java-5.0.7.jar for MySQL 5.0, or "sqljdbc.jar" for SQL Server 2005) needs to be copied into the "NCIA/Webroot/WEB-INF/lib" directory prior to the build, and it is also a good idea to make it available to your JBoss instance by copying it into your "JBoss/lib" directory.

3.1.1.7 Other Properties To Consider

As you customize or debug NCIA, you will want to consider properties that are set in `conf/ncia.properties` as well as `conf/mail.properties`. For a basic installation, you should not need to be concerned with those properties files.

The `date_format` property in `ncia.properties` controls the date string format used when querying against the database using dates. The default is dd-MMM-yyyy. Change as needed to match your database's configuration.

3.1.2 Install Any Necessary Critical Patch Files

If there are any critical patch files that need to be installed, they should be installed at this point of the installation process, prior to conducting the actual build.

3.1.3 Build Application

To build NCIA, change to the NCIA directory of the download package, which is the directory where `build.xml` is located. In that directory, run the following command, making sure to use all 3 of the command line arguments:

```
ant build-system
```

```
      –Dftp_url=yourFtpUrl

      –Dserver_url=http://yourServerUrl:59080

-
Dcsm_hibernate_location=pathToJBoss/server/default/conf/ncia.csm.hiber
nate.cfg.xml
```

After the Ant script has finished running, the file `ncia.ear` will appear in the output directory.

3.2 CONFIGURE COMMON SECURITY MODULE

3.2.1 Configure the ncia.csm.hibernate.cfg.xml file

Make sure that the dialect property is set properly based on the database you have chosen to use, which will be one of the following settings:

```
<!-- Oracle Dialect -->
          <property
name="dialect">org.hibernate.dialect.OracleDialect</property>

<!-- MySQL Dialect -->
          <property
name="dialect">org.hibernate.dialect.MySQLDialect</property>

<!-- SQL Server Dialect -->
          <property
name="dialect">org.hibernate.dialect.SQLServerDialect</property>
```

3.2.2 Copy CSM Hibernate Configuration File

The CSM requires a Hibernate 3 configuration file. Because of the way the CSM is designed, the configuration file cannot be packaged in a WAR or EAR file.

Determine a location where the configuration file will reside. It is highly recommended to utilize the JBoss "`server/default/conf`" directory for this purpose. Copy the file `conf/csm/ncia.csm.hibernate.cfg.xml` to that location.

3.2.3 Configure ApplicationSecurityConfig.xml

Edit the ApplicationSecurityConfig.xml file located in the "`NCIA/conf/csm`" directory. Make sure that both `hibernate-config-file` tags are configured to point to your `ncia.csm.hibernate.cfg.xml` file that you copied in the previous step. The directory you configure should read something like "`pathToJBoss/server/default/conf/ncia.csm.hibernate.cfg.cml`".

Note that there are two sets of `hibernate-config-file` tags; one each for the NCIA application, and one for the NCIA-UPT application; the change needs to be made for both.

3.2.4 Copy ApplicationSecurityConfig.xml

Copy the file `conf/csm/ApplicationSecurityConfig.xml` file to a configuration directory of your choice. For JBoss, it is recommended that the file be placed in the jBoss "`server/default/conf`" directory.

3.3 CONFIGURE APPLICATION SERVER

3.3.1 Configure port number

Configure the application server to run on the appropriate port number. It is recommended to run your application server on port 59080. You can configure this by editing the JBoss file: "server/default/deploy/jbossweb-tomcat55.sa/server.xml". The HTTP 1.1 Connector which normally runs on port 8080 should be configured to run on port 59080.

Make sure there are no conflicts with Tomcat and Globus if those components are running on the same machine. (Globus uses port 8080 by default, and this is the recommended setting for Globus).

3.3.2 Configure datasource

A data source must be configured in the application server with a JNDI name of `ncia`. In jBoss, this can be configured by creating a file named `oracle-ds.xml` (or `mysql-ds.`xml or `ms-sql-ds.xml`) in the jBoss `deploy` directory with the following contents:

```
<datasources>
  <local-tx-datasource>
    <jndi-name>ncia</jndi-name>
    <connection-url>{DB_URL}</connection-url>
    <user-name>{DB_USER}</user-name>
    <password>{DB_PASS}</password>
    <driver-class>{DRIVER_CLASS}</driver-class>
  </local-tx-datasource>
</datasources>
```

Fill in the parameters as follows:

Parameter	Value

DB_URL	JDBC URL for the database. For example,
	`jdbc:oracle:thin:@dbserver.yourorg.org:1521:dbid`
	
	or
	
	`jdbc:sqlserver://localhost:1433;databaseName=NCIA`
	
	or similar.

DB_USER	Database user name
DB_PASS	Database password

DRIVER_CLASS	Driver class name, for example:
	
	For Oracle, it will be:
	`oracle.jdbc.driver.OracleDriver`
	For MySQL, it will be:
	`com.mysql.jdbc.Driver`
	For SQL Server, it will be:
	`com.microsoft.sqlserver.jdbc.SQLServerDriver`

3.3.3 Add JAAS Login Modules

NCIA uses the Common Security Module (CSM) for authentication. CSM performs authentication using JAAS (Java Authentication and Authorization Service) login modules. The application server must be configured to use a login module for the NCIA and NCIA-UPT applications. NCIA-UPT represents the CSM User Provisioning Tool (UPT) that is used to administer security.

In jBoss, this can be configured by adding the following to `conf/login-config.xml`:

```
<application-policy name="NCIA">
  <authentication>
   <login-module
     code="gov.nih.nci.security.authentication.loginmodules.RDBMSLoginModule"
     flag="required">
         <module-option name="driver">
            {DRIVER_CLASS}
          </module-option>
          <module-option name="url">{DB_URL}</module-option>
          <module-option name="user">{DB_USER}</module-option>
          <module-option name="passwd">{DB_PASS}</module-option>
          <module-option name="query">
             SELECT * FROM CSM_USER WHERE LOGIN_NAME=? and PASSWORD=?
          </module-option>
   </login-module>
  </authentication>
</application-policy>

<application-policy name="NCIA-UPT">
  <authentication>
   <login-module
     code="gov.nih.nci.security.authentication.loginmodules.RDBMSLoginModule"
     flag="required">
         <module-option name="driver">
            {DRIVER_CLASS}
```

```
        </module-option>
        <module-option name="url">{DB_URL}</module-option>
        <module-option name="user">{DB_USER}</module-option>
        <module-option name="passwd">{DB_PASS}</module-option>
        <module-option name="query">
           SELECT * FROM CSM_USER WHERE LOGIN_NAME=? and PASSWORD=?
        </module-option>
    </login-module>
  </authentication>
</application-policy>
```

Fill in the parameters as follows:

Parameter	Value

{DRIVER_CLASS}	Driver class name, for example:
	
	For Oracle, it will be:
	`oracle.jdbc.driver.OracleDriver`
	For MySQL, it will be:
	`com.mysql.jdbc.Driver`
	For SQL Server, it will be:
	`com.microsoft.sqlserver.jdbc.SQLServerDriver`

{DB_USER}	Database user name
{DB_PASS}	Database password

{DB_URL}	JDBC URL for the database. For example,
	
	`jdbc:oracle:thin:@dbserver.yourorg.org:1521:dbid`
	
	or
	
	`jdbc:sqlserver://localhost:1433;databaseName=NCIA`
	
	or similar.

3.3.4 Configure log4j

Application servers sometimes have their own log4j configuration. If your application server uses log4j (like jBoss does), add the following to `log4j.xml`:

```
<category name="org.hibernate">
   <priority value="FATAL"/>
</category>
```

```
<category name="gov.nih.nci">
   <priority value="ERROR"/>
</category>
```

3.3.5 Configure System Properties Service

The CSM looks for a JVM parameter to determine where to look for its configuration file. Different application servers have different ways of setting JVM parameters. Set the parameter `gov.nih.nci.security.configFile` to the full path of the file `ApplicationSecurityConfig.xml` (see section 3.2.3).

In jBoss, this can be done by adding the following under the `org.jboss.varia.property.SystemPropertiesService` MBean in the `properties-service.xml` file located in the `deploy` directory.

```
<attribute name="Properties">
     gov.nih.nci.security.configFile=../server/default/conf/Applicatio
nSecurityConfig.xml
</attribute>
```

Note: Make sure that when cutting and pasting that the text within the XML tags is all on one line.

Other parameters that should be configured in this file include the imaging server URL, the quarantine directory, the FTP and ZIP directories, the image path head, the image path pattern, and the JBoss MQ URL.

3.3.6 Configure JMS

A JMS Queue must be configured for the `ImageZippingMDB` to function. The queue must be named `imageQueue`. In jBoss, this can be configured by placing a file named `ncia-jbossmq-destinations-service.xml` in the `deploy/jms` directory with the following contents:

```
<server>
 <mbean code="org.jboss.mq.server.jmx.Queue"
       name="jboss.mq.destination:service=Queue,name=imageQueue">
    <depends optional-attribute-name="DestinationManager">
           jboss.mq:service=DestinationManager
    </depends>
 </mbean>
 <mbean code="org.jboss.mq.server.jmx.Queue"
       name="jboss.mq.destination:service=Queue,name=
curationQueue">
    <depends optional-attribute-name="DestinationManager">
           jboss.mq:service=DestinationManager
    </depends>
  </mbean>
</server>
```

3.3.7 Memory

The application server software needs to be configured to use at least one gigabyte of memory. For jBoss using the Sun JVM, this is configured by adjusting the `JAVA_OPTS` parameter in the script that starts jBoss.

3.4 DEPLOY NCIA APPLICATION

3.4.1 Deploy NCIA EAR

The EAR file for the NCIA application must be deployed. Deploy the file output/ncia.ear to the application server. In jBoss, this is done by copying ncia.ear to the deploy directory.

3.4.2 Deploy CSM UPT WAR

The WAR file for the CSM User Provisioning Tool must also be deployed. Deploy the file `conf/csm/upt.war` to the application server. In jBoss, this is done by copying `upt.war` to the `deploy` directory.

Please note: The CSM User Provisioning Tool is currently not supported in any system using a SQL Server database – this is due to an issue within the UPT application itself. Ideally, this defect will be fixed in a future release. If you choose to use SQL Server, you will have to perform user provisioning manually in the CSM database tables.

3.5 DATABASE

NCIA requires a database. After creating a schema, run the DDL script. If you are using Oracle, follow run the script located in `database/I2PRODschema_final.sql` in the download package. The script `database/seedData.sql` must also be run to populate the database with initial data.

If you are using MySQL, run the scripts located in the `database/mysql` directory.

These scripts will create all of the required tables, views, and indexes.

For more detailed information about how to configure your database for optimal performance, refer to Appendix C.

If you are not using Oracle or MySQL, you will need to manually convert the script to work with the database you are using. Refer to Appendix B.

3.5.1 Sample Oracle Database

A sample database is provided as part of the download zip file in the SampleNCIADatabase directory. Use the sample database if you do not want to set up your own or if needed for debugging purposes.

The sample database includes GENERAL_IMAGE records as examples, but does not include the corresponding image files. Each instance of NCIA needs to supply its own image files.

The database is in Oracle 9 export format inside of a zip file.

Once you have pointed your CSM (via `login-config.xml`) and application server (`oracle-ds.xml`) to the sample database, you can log in using the following credentials:

- Login: john.doe@hotmail.com
- Password: johndoe

3.6 DEPLOY NCIA GRID COMPONENTS

3.6.1 Prerequisities

Prior to deploying the NCIA grid components, the following steps must be performed:

1. Install Globus Toolkit (see section 2.1.4).
2. Set GLOBUS_LOCATION environment variable (see section 2.1.4).
3. Install Ant 1.6 and Java 1.5.
4. Make sure that the bin directories of Ant and Java are in the system path

3.6.2 Staging and Configuration

Copy all files from the NCIA Grid directory of the download package to a temporary location on the server where Globus is installed.

The file `build/classes/nciaClientContext.xml` must be configured to contain the hostname and port number where that the NCIA application will be listening on. The default is `localhost:59080`. If NCIA is running on a different hostname or port, change the file to contain the correct values.

3.6.3 Deploying NCIA Grid Services

To deploy the NCIA grid services to Globus, run `ant deployGlobus` in the directory where the NCIA Grid component's build.xml is located.

3.6.4 Starting Globus

To start Globus, run `globus-start-container –nosec` from Globus's `bin` directory.

3.6.5 Configure Grid Node List (Optional)

To make an NCIA instance aware of other grid nodes, populate the NCIA_GRID table. The columns are as follows:

- GRID_NODE_PK_ID – the primary key of the grid node – a unique number
- NODE_NAME – string describing the node
- URL – URL to the grid services. It will follow this format:
    - `http://{HOSTNAME}:{PORT}/wsrf/services/cagrid/NCIAQueryService`
        - `HOSTNAME` = the hostname of the Globus server
        - `PORT` = the port number that Globus runs on. This is typically 8080.

4. MIRC INSTALLATION

Medical Imaging Resource Center (MIRC) is open source software developed by the Radiological Society of North America (RSNA) that allows for submission of DICOM image files and annotation files. More information about MIRC can be found in section at http://www.rsna.org/mirc.

NCIA uses version T29a of the MIRC software.

The NCIA team has extended the standard MIRC software to provide functionality that inserts data about each image into the database. Having information about each image in the database facilitates searching.

4.1 INSTALL REQUIRED SOFTWARE

MIRC T29a requires the following software to be installed (see section 2.1 for details):

- Ant 1.6.2
- Java 1.5
- JAI (Java Image I/O API)
- Tomcat 5.5
    - Ensure that the port that Tomcat runs on will not conflict with Globus or the NCIA application server

4.2 BUILD MIRC APPLICATION

The MIRC application including the NCIA customizations is built using Ant 1.6.2.

First, change to the MIRC directory of the download package. Then, run the following command:

```
ant build-prod -Ddb_user={DB_USER} -Ddb_pass={DB_PASS} -Ddb_url={DB_URL}
```

Fill in the parameters as follows. The parameter values are placed into a Hibernate configuration file used by NCIA's MIRC:

Parameter	Value
DB_USER	Database user name
DB_PASS	Database password
DB_URL	JDBC URL for the database. For example, `jdbc:oracle:thin:@dbserver.yourorg.org:1521:dbid`

The result of the build will be a file named `MIRCSite-Installer.jar` located in the output directory under the MIRC directory of the download package.

4.3 INSTALL AND CONFIGURE MIRC

To install NCIA's MIRC to Tomcat, refer to the MIRC Installation directions located in the Documentation directory of the download package. Refer to the section titled "Installing MIRCsite Software". These directions use the `MIRCsite-installer.jar` file as a starting point. `MIRCInstaller.jar` can be run by typing `java –jar MIRCsite-installer.jar` at the command prompt.

5. MIRC FIELD CENTER SOFTWARE INSTALLATION

MIRC Field Center is used to submit files from a field center (hospital, university, research center, etc) to the MIRC server (installed in section 3.6).

Instructions for installing MIRC Field Center are contained in a separate document. Please refer to the document located in the Documentation directory of the NCIA installation zip file. The installation files for the field center are in the MIRCFieldCenter directory of the download package. After building MIRC, the Field Center install file is located in the `MIRC/output` directory.

After installing MIRC Field Center, be sure to properly configure anonymization to ensure that no private health information is submitted to the server.

By default, images are set to a QA status of "Not Yet Reviewed." To make images available for searching, use the QA Tool feature to change the QA status to "Visible".

6. REPORTING MODULE

NCIA captures data about how users use the system. Implementations of NCIA can optionally choose to view reports based on this data. This includes:

- Timestamps of logins
- Download history
- Query history
    - When queries were run
    - What criteria was used

This data is made available using several views. A reporting tool is not provided. This approach allows implementations of NCIA to use the appropriate reporting tool that meets their needs. The views were tested using Crystal Reports. The trivial case would be to not have a reporting tool and just use SQL to query the views.

The reporting views are as follows:

Name	Description
TOTAL_COUNT_LOGINS	Number of logins by all users
TOTAL_COUNT_QUERY	Number of queries by all users
AVERAGE_DOWNLOAD_SIZE	Average size of files requested for download
ANATOMIC_SITE_PATTERN	Number of queries run where each anatomic site was included as criteria
MODALITY_PATTERN	Number of queries run where each modality was included as criteria
PROJECT_PATTERN	Number of queries run where each project was included as criteria

Each of these views provides data over three time periods:

- Last 30 Days
- Year to Date
- Total from Beginning

The DDL scripts run in section 3.5 will create these views. Reporting tools should be set up to access the reporting views using a separate user that is restricted to only select on the reporting views. The scripts will also attempt to grant permission for a user named `nciareport` that can be used for that purpose. A reporting tool should only have access to the reporting views, not all of the data.

7. VERIFICATION

7.1 MIRC

MIRC can be tested by attempting to submit images to the NCIA MIRC using Field Center. The submission was successful if:

- Image is stored in the file system
- Data about the image is stored in the database
- DICOM header data was anonymized properly.

MIRC has an administration page that can be used to verify that services are running and to view logs. See the MIRC web site for more information.

7.2 NCIA

To verify that NCIA is working correctly, test the following once images have been submitted and made visible:

- Register as a new user
- Log in as that user
- Perform a search
- Add data to the data basket
- Download the data from the data basket (be sure to test data basket sizes both less than and greater than the FTP threshold)

-

8. LICENSE

8.1 NCIA SOFTWARE LICENSE

Copyright 2001-2006 Science Applications International Corporation ("SAIC"). The software subject to this notice and license includes both human readable source code form and machine readable, binary, object code form ("National Cancer Imaging Archive (NCIA) web application"). The NCIA Software was developed in conjunction with the National Cancer Institute ("NCI") by NCI employees and employees of SAIC. To the extent government employees are authors, any rights in such works shall be subject to Title 17 of the United States Code, section 105.

This NCIA Software License (the "License") is between NCI and You. "You (or "Your") shall mean a person or an entity, and all other entities that control, are controlled by, or are under common control with the entity. "Control" for purposes of this definition means (i) the direct or indirect power to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

This License is granted provided that You agree to the conditions described below. NCI grants You a non-exclusive, worldwide, perpetual, fully-paid-up, no-charge, irrevocable, transferable and royalty-free right and license in its rights in the NCIA Software to (i) use, install, access, operate, execute, copy, modify, translate, market, publicly display, publicly perform, and prepare derivative works of the NCIA Software; (ii) distribute and have distributed to and by third parties the NCIA Software and any modifications and derivative works thereof; and (iii) sublicense the foregoing rights set out in (i) and (ii) to third parties, including the right to license such rights to further third parties. For sake of clarity, and not by way of limitation, NCI shall have no right of accounting or right of payment from You or Your sublicensees for the rights granted under this License. This License is granted at no charge to You.

1. Your redistributions of the source code for the Software must retain the above copyright notice, this list of conditions and the disclaimer and limitation of liability of Article 6, below. Your redistributions in object code form must reproduce the above copyright notice, this list of conditions and the disclaimer of Article 6 in the documentation and/or other materials provided with the distribution, if any.

2. Your end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by SAIC and the National Cancer Institute." If You do not include such end-user documentation, You shall include this acknowledgment in the Software itself, wherever such third-party acknowledgments normally appear.

3. You may not use the names "The National Cancer Institute", "NCI" "Science Applications International Corporation" and "SAIC" to endorse or promote products derived from this Software. This License does not authorize You to use any trademarks, service marks, trade names, logos or product names of either NCI or SAIC, except as required to comply with the terms of this License.

4. For sake of clarity, and not by way of limitation, You may incorporate this Software into Your proprietary programs and into any third party proprietary programs. However, if You incorporate the Software into third party proprietary programs, You agree that You are solely responsible for obtaining any permission from such third parties required to incorporate the Software into such third party proprietary programs and for informing Your sublicensees, including without limitation Your end-users, of their obligation to secure any required permissions from such third parties before incorporating the Software into such third party proprietary software programs. In the event that You fail to obtain such permissions, You agree to indemnify NCI for any claims against NCI by such third parties, except to the extent prohibited by law, resulting from Your failure to obtain such permissions.

5. For sake of clarity, and not by way of limitation, You may add Your own copyright statement to Your modifications and to the derivative works, and You may provide additional or different license terms and conditions in Your sublicenses of modifications of the Software, or any derivative works of the Software as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

6. THIS SOFTWARE IS PROVIDED "AS IS," AND ANY EXPRESSED OR IMPLIED WARRANTIES, (INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE) ARE DISCLAIMED. IN NO EVENT SHALL THE NATIONAL CANCER INSTITUTE, SAIC, OR THEIR AFFILIATES BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

8.2 MIRC AND MIRC FIELD CENTER LICENSES

MIRC software is governed by the RSNA Public License: http://mirc.rsna.org/rsnapubliclicense.

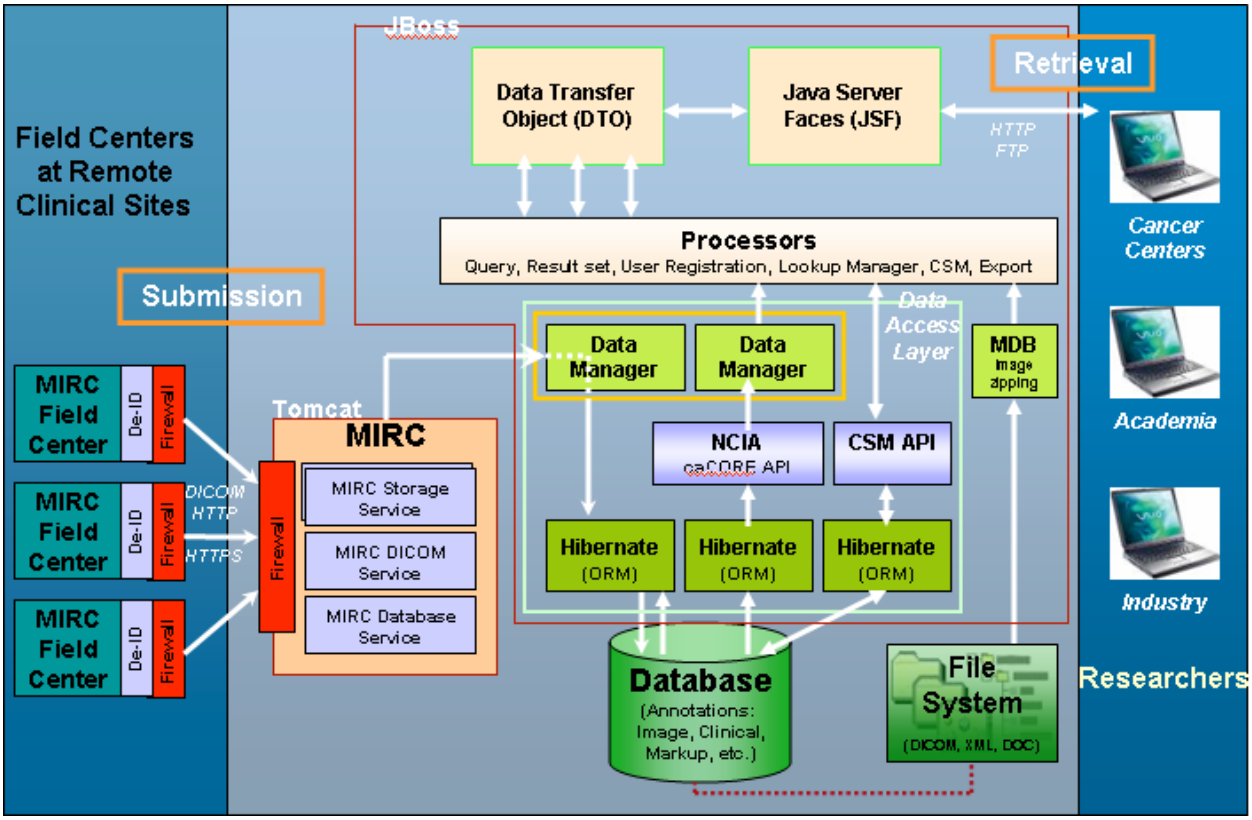
8.3 THIRD PARTY TOOLS

See Appendix D.

9. APPENDIX A – TECHNICAL OVERVIEW

The purpose of this section is to give an overview of the NCIA architecture. The following diagram shows the overall architecture of NCIA. An overview of each component is provided in the sections immediately following the diagram.

The purpose of this section is to give an overview of the NCIA architecture. The following diagram shows the overall architecture of NCIA. An overview of each component is provided in the sections immediately following the diagram.



9.1.1 User Client Software

NCIA is a web based application. Users (cancer centers, academia, industry, etc) access NCIA through commonly available web browser software using the HTTPS protocol. JavaScript must be enabled in the browser.

9.1.2 User Interface

NCIA's web-based user interface is implemented as HTML and JavaScript. The use of JavaScript is limited to improving usability only.

JavaServer Pages (JSPs) are used to generate dynamic HTML based on user actions. JSPs are a combination of HTML, Java Code, and custom JSP tags that get compiled into servlet classes.

JavaServer Faces (JSF) is used to simplify development of the JSPs. JSF is helpful by providing:

- Custom tags that are used to generate form input fields such as text boxes, checkboxes and radio buttons.
- Configurable mechanism for mapping input in HTML forms into JavaBeans
- An event model that maps button or link clicks to methods on JavaBeans
- Configurable navigation model

The Tiles framework is used to create reusable page components.

JavaBeans known as "backing beans" interact with the JSF components in JSPs. Examples are SearchWorkflowBean, BasketBean, and SearchResultsBean. Each of these beans manages the UI interaction in certain areas of the application.

Data Transfer Objects (DTOs) represent lightweight versions of domain classes used for presentation. When data is retrieved from the database for presentation in the UI, it is placed into DTOs.

9.1.3 Middle Tier

The middle tier is responsible for bridging the gap between the user interface that presents data and the back end that retrieves and stores the data. Middle tier components are called processors or managers.

The most important middle tier component is the ResultSetManager, which processes search requests. Based on the type of search requested, it builds a query based on the search criteria, sends the query to the back end, gets the results and converts the results into DTOs that the user interface layer can understand.

9.1.4 Back End

The NCIA back end components provide access to persistence services for retrieving and storing data in the database. Hibernate, an open source tool, is used as the Object-Relational Mapping (ORM) software.

9.1.4.1 DataAccessProxy

Access to back end components goes through the DataAccessProxy, which provides methods for searching and storing data. Each instance of DataAccessProxy is initialized to use the appropriate DataAccess class based on the way in which data is being accessed.

9.1.4.2 caCORE SDK Data Access

caCORE provides a robust set of tools for accessing data using Java. Based on a Unified Modeling Language (UML) model of the domain classes and database structure, the caCORE

SDK generates a web application that provides access to the data represented by the modeled domain classes. The SDK simplifies the process of creating a back end because application developers do not need to worry about how data is being obtained from the database or generating mapping files for modeled domain classes. Access provided using the caCORE SDK is read only. caCORE leverages Hibernate for object / relational mapping.

The caCORE generated API can be accessed either remotely or locally. Accessing the toolkit remotely involves using the caCORE client classes to call a web application. The request and response is tunneled using HTTP. Local access involves directly calling the caCORE classes and avoids the overhead of HTTP tunneling.

Whenever possible, local access to the caCORE API is used to improve performance. Local access is used by the search infrastructure since the searching code has easy access to the SDK classes. Remote access is used by the ImageZippingMDB. Because of complexities with classloaders and details of the SDK's implementation, it is difficult to call the SDK locally from the MDB, so remote access is used.

When searching, NCIA initializes DataAccessProxys to use either the local or remote caCore data access classes.

9.1.4.3 Hibernate 3 Data Access

When data needs to be updated instead of just read from the database, the caCORE API cannot be used. Instead, NCIA uses Hibernate version 3. When dealing with data that needs to be updated (with the exception of CSM data – see below), NCIA initializes DataAccessProxys to use the Hibernate 3 data access class. When there are requirements to update data, NCIA uses its own hand created domain classes and Hibernate mapping files.

9.1.4.4 Image Zipping Message Driven Bean

Users can download large sets of images from NCIA. To facilitate easy downloads, the image files are zipped up into one large file. Since the system often deals with several hundred megabytes or gigabytes of data, the zipping process can take a very long time. Because of this, zipping of very large files is performed using a Message Driven Bean (MDB).

MDBs are a special type of Enterprise Java Bean (EJB) that is called every time a message is placed on a queue. When a user requests a download of large amounts of data, NCIA places a message on a Java Message Service (JMS) queue. The MDB then accesses the message on the queue and processes it. The use of MDBs provide not only asynchronous behavior, but also allow for transactional integrity and failover.

Once the zipping process is complete, the zip file is placed on the NCIA FTP site for download and the user is sent an email. For small downloads (less than a configurable threshold), files are zipped synchronously while the user waits.

9.1.4.5 Curation Spreadsheet Message Driven Bean

Users can upload a spreadsheet of curation data to NCIA. Because there could be large amounts of data in the spreadsheet, the processing is done asynchronously. When a spreadsheet is uploaded, its data is placed on a queue which is then accessed by the CurationSpreadsheetMDB. The MDB processes the data and then sends an email when processing is complete.

9.1.5 Common Security Module (CSM)

The Common Security Module (CSM) is a NCICB component that is integrated into NCIA to provide authentication and authorization services. When users log in, NCIA calls the CSM to determine if the user is authentic. For authorization, NCIA calls CSM to determine the protection elements and roles to which the user has access. It can then filter results according to the user's authorization.

The CSM also provides a web application called the User Provisioning Tool (UPT) that allows for maintenance of user authorization data.

9.1.6 jBoss Application Server

The user interface, middle tier, and back end components are run within the jBoss Java 2 Enterprise Edition (J2EE) application server. jBoss is an open source application server that provides a servlet container, an EJB container, the full range of J2EE 1.4 features as well as extended enterprise services including clustering, caching, and persistence. NCIA runs on jBoss 4.0.2.

9.1.7 File System

DICOM files and any associated annotations files are stored in the file system. The MIRC software places files on the file system when they are submitted in a directory structure that reflects the organization of images by patient, study, and series. Each image record in the database stores the image's file system path.

Zipped images files that are ready for download are also placed in the file system.

9.1.8 Database

Oracle Relational Database Management System (RDBMS) software is used as the persistent store for NCIA application data. Java classes access the database using the Java Database Connectivity (JDBC) API.

9.1.9 MIRC

Medical Imaging Resource Center (MIRC) software was developed by the Radiological Society of North America (RSNA). The goal of MIRC is to provide tools for sharing radiological images for education and research purposes. MIRC offers a simple way to identify, index and retrieve images, teaching files and other radiology information.

For NCIA, MIRC handles the submission of images, storing images in the file system, and exporting images to the database. After images are submitted to the database, the private DICOM fields are anonymized so that they are not available to users who download the images.

MIRC is run on Tomcat, an open source servlet container developed by the Apache Software Foundation.

9.1.9.1 MIRC DICOM Service

The MIRC DICOM Service allows for DICOM images to be submitted from field centers using the DICOM protocol.

9.1.9.2 MIRC Storage Service

The MIRC Storage Service places submitted DICOM files into a directory structure in a directory structure that reflects the organization of images by patient, study, and series.

9.1.9.3 MIRC Database Service

The MIRC Database Service provides a hook for developers to include custom code that will export the DICOM information to a database. For NCIA, the database export process extracts selected data from the DICOM header fields and stores it in the NCIA table structure.

Hibernate 3 is used to interact with the database. Mapping files are generated using the caCORE SDK, but are then modified to allow for updates instead of reads.

9.1.10 MIRC Field Center

MIRC Field Center along with MIRC File Sender software is used to submit images to MIRC from the sites where they are generated (the “field center”). Typically, images are generated as part of a clinical trial. MIRC Field Center is used by researchers at hospitals, cancer centers, and universities to send their images to a MIRC site for sharing.

Due to privacy regulations, personally identifiable information must be removed from images before they can leave the field center. MIRC Field Center software provides tools to anonymize any personally identifiable data that is found in the DICOM tags.

10. APPENDIX B – USING A DIFFERENT DATABASE

As mentioned above, NCIA was tested primarily using Oracle 9.2. If you are using a different database, you will need to make the following changes:

- Change the JDBC drivers packaged with each application to be the appropriate ones for your database
- Modify DDL (see section 3.5) as needed to create a script that will create tables, views and object database objects on your database.
- Change the Hibernate configuration files used by NCIA and MIRC to use the appropriate dialect for your database.
- Modify the Hibernate configuration files contained in `NCIA/toolkit/nciaservice.war` to use the proper dialect for your database.

11. APPENDIX C – ADDITIONAL ORACLE CONFIGURATION

Create NCIA database (two options)

I. General Oracle database configuration for NCIA database.

Create Oracle database schema account and appropriate tablespace needs to be done by an Oracle database administrator. Please work with your Oracle DBA for the database creation and data import.

1. Configure Oracle database init.ora parameter file.

- Open your Oracle database init.ora parameter file.
- Change
  - query_rewrite_enabled=true
  - query_rewrite_integrity=trusted
- Save and close init.ora parameter file.
- Shutdown and restart the Oracle database to allow the parameters to take effect.

2. Check to make sure that your system has at least 40 GB of free disk space on the database server.

3. Create a NCIA tablespace in an Oracle database.

- Log on to Oracle database as an Oracle DBA.
- Create tablespace “NCIA”.
- Turn on autoextend on the tablesapce.
- Check to make sure that there is at least 40 GB of free disk space available for tablespace “NCIA”.
  - Example of create tablespace (please change the path for the datafile appropriate to your Operating System and availability of storage space). Following is a sample for UNIX:

```
CREATE TABLESPACE ncia
LOGGING
DATAFILE '/data/oracle/oradata/ncia_01.dbf' size 1900 m,
         '/data/oracle/oradata/ncia_02.dbf' size 1900 m,
         '/data/oracle/oradata/ncia_03.dbf' size 1900 m,
         '/data/oracle/oradata/ncia_04.dbf' size 1900 m,
         '/data/oracle/oradata/ncia_05.dbf' size 1900 m,
         '/data/oracle/oradata/ncia_06.dbf' size 1900 m,
         '/data/oracle/oradata/ncia_07.dbf' size 1900 m,
               .
               .
               .
         '/data/oracle/oradata/ncia_20.dbf' size 1900 m;
```

4. Create an NCIAprod database user.

- Log on to Oracle database as an Oracle DBA.

- Create database user account using “NCIA” tablespace as the default tablespace. The user name must be consistent with what was defined in the property files (described later in this document). “NCIAprod” is suggested as the user name.
- Grant “resource”, “query rewrite” to the user.
  - Example of creating user.
    a. create user nciapord identified by xxxxxx
       default tablespace NCIA
       temporary tablespace temp
       quota unlimited on NCIA;

    b. grant connect, resource, query rewrite to NCIAprod;

II. NCIA database schema creation options.

1. Option 1: Create database schema without data using caCORE schema creation SQL.

   - Run the DDL in the NCIAprod database user account defined above.

2. Option 2 (NOT required): Create database schema and import data using the Oracle dump file

   - Unzip “ncia_db.dmp.zip” file.
   - Perform a database import using the database user defined above and “ncia_db.dmp” as the data files.
   - Please keep a log of Oracle import.
     - Example of import script.

       nohup imp system/password fromuser=cabiostage touser=cabioprod filesize=6GB file=ncia_db.dmp log=cabio_db.log &