



System Administration and Development Guide



January 25, 2011 v.1



Contents

Chapter 1: System Architecture.....	1
Front End Applications	1
Presentation Layer	1
Application Layer.....	2
Service Layer.....	2
Data Access Layer	3
Data Persistence Layer.....	3
Illustration of the Architecture	3
Data Warehouse	4
Development Environment Application Server	4
Development Environment Database Server	7
Staging/QA Database and Application Server	15
Production Environment Application Server.....	17
Production Environment Database Server.....	19
Training Application Server.....	21
Training Database Server	21
Proxy and Backup Server	22
The eRoom.....	23
eRoom Specifications.....	24
Chapter 2: Oracle Operations	25
Configuration.....	25
Performance Pack Setup	25
Rman Backup Configuration.....	26
Schema Deployment Between Servers	26
Step 1. Generate a Pair of Keys	26
Step 2. Configure SSH Key-based Authentication	27
Step 3. Install Perl Modules	28
Step 4. Create Oracle Directories for Export and Import.....	29
Step 5. Export Data from the Development Server	30
Step 6. Move Exported File(s) from the Source Server to the Target	30
Step 7. Prepare for Schema Import	31
Step 8. Import Source Schemas into the Target Database	31
Step 9. Deployment Error Prevention.....	31
Future Enhancements	32
Scripts to Facilitate the Deployment Process	33

Using Oracle impdp and expdp.....	44
Tips.....	44
Schema and Table Export.....	44
Importing the Data.....	45
Additional Information	45
Chapter 3: Amazon Web Services Operations	47
File Transfer Using WinSCP.....	47
Creating a Self-Signed SSL Certificate for Apache.....	47
Shutting Down and Starting Up JBoss	50
Shutting Down and Starting Up the Oracle Database Server	50
Resetting Passwords	51
Resetting Your VNC Password	51
Resetting Your Linux Password.....	51
Setting Up the Production Database Server	51
Installing RPMs	53
Configuring SQL Developer.....	54
Providing VNC Access	54
Step 1: Linux Group/User Creation	54
Step 2: Set Up the Account to Allow VNC Viewer Access	55
Additional Information	55
A Note on Using TOAD on the Cloud	56
Chapter 4: Database Access Security	57
Access Credentials.....	57
Oracle Credentials	57
Credentials for Other Data Sources.....	58
Access Control	58
JNJ_Application Security Group	58
JNJ_Common Security Group.....	59
JNJ_Integration Security Group.....	62
JNJ_Staging_DB Security Group.....	62
JNJ_Warehouse_DB Security Group	63
tranSMART IP Address White List	63
High-Level View of System and Security	64
Chapter 5: Release Management	67
Deployment to the Production Server	67
Application Deployment to Production Server.....	67

Database Deployment to Production Server	68
Tag a New Release in SVN.....	68
Create a Release Note	69
Deployment to the Public Training Server	69
Deployment URLs.....	70
Application Deployment to the Training Server.....	70
Database Deployment to the Training Server	70
Chapter 6: Development Tools	71
Source Control (SVN)	71
Eclipse	71
Eclipse Plugins	71
Grails.....	72
Oracle SQL Developer	72
Indexer Tool	73
Building Indexer.....	73
Usage	73
Dana Farber GCOD Dataset	75
Oracle Users.....	75
Tablespaces.....	75
Oracle Directory	77
Dana Farber GCOD Datasets Loaded	77
Dataset Explorer Development Environment Setup	77
Chapter 7: Coding Standards.....	79
Database Coding Standards.....	79
Database Names	79
Table, View, and Procedure Names	79
Column Names	80
Index Names	80
SQL Code.....	80
Script Headers	81
Grails Coding Standards.....	81
Domain Classes	81
Controllers and Views	82
Services.....	83
Data Structures	83
Java Coding Standards.....	83
Naming Conventions.....	83

Specific Naming Conventions	86
Files	90
Statements	91
Layout and Comments	97

Chapter 8: Other Topics.....105

Object Base Class Diagram	105
Z-Score Calculation	105
Step 1. Data Pre-processing	105
Step 2. Z-Score Calculation	106
Step 3. Data Post-Processing	107
Sample Z-Score Calculation Scripts	107
Stored Procedures to Retrieve Z-Score for Heat Maps	109
Pathway Studio Enterprise Server	115
Users/Info	115
Start/Stop License Server	115
Start/Stop Tomcat Server	115
Creating Schema Objects	115
Populating the Database	115
Post-load Database Optimization	116
GenePattern Server Installation	116
On Unix	116
On Windows	119
Useful Links.....	120
Semantic Browser	120
Java Framework for Semantic Queries	120
Project Management tool at J&J	120
Rembrandt	120
ResNet Server Database	120
Pictor.....	120
Chip (MicroArray Analysis Pipeline).....	121
OmniViz Version 6.0.1	121
Ingenuity	121
SAS Software	121
General J&J Software.....	121

Chapter 1

System Architecture

Front End Applications

tranSMART front-end applications consist of N-tiered web applications and a standalone application. These applications work together in a Service Oriented Architecture (SOA). The web applications are designed as rich internet applications by leveraging AJAX technology. They are built upon industry-standard frameworks including: Model View Controller (MVC), Spring, and Hibernate. These frameworks enable the applications to be scaled up to an enterprise scale solution.

The front-end applications are designed in the tiers or layers described in the following sections.

Presentation Layer

The Presentation layer utilizes Java Server Pages (JSP), Groovy Server Pages (GSP), and a high performance JavaScript library Ext-JS to present interactive user interfaces.

- JSP and GSP serve as the base presentation interface.
- The Groovy on Grails platform implements an MVC architecture by using Java technologies (including Spring and Hibernate) to enable page flow control and separate data presentation from data retrieval.
- Ext-JS is used as a JavaScript framework for the following:
 - Rich GUI development (such as auto-completion boxes)
 - Controlling specialized GUI controls
 - Handling AJAX transactions
- Java-based servlets handle AJAX requests and HTTP proxy requests to other external web services.
- Links connect users to external portal sites, and use a pop-up to a separate browser (such as with Entrez Gene) rather than refreshing a panel within the tranSMART frame.

Application Layer

The Application layer utilizes:

- Grails controllers to provide page flow control and formatting of data for the presentation layer
- Java Spring Framework for service integration
- Java Message Service (JMS) for messaging service
- Portal application for portlet registration and management
- Axis2 web service application that manages RESTful web services

Service Layer

Search Tool

The search tool is comprised of:

- The Federated/External Query Service, built with Grails service classes. This component queries internet Entrez/GO data and Johnson & Johnson Intranet Pictor/Hydra data.
- The Document Search Engine, built with Grails service classes and a Java library. This feature queries a Lucene-based document index and generates links to document repositories. PDF, Microsoft® Office (Word, Excel, PowerPoint), HTML, text documents are currently supported.
- The Internal Analysis Query Services, built with Grails service classes, uses Grails domain objects to query the Data Warehouse and Data Marts.

Dataset Explorer (i2b2 Hive)

The i2b2 Hive Services is built on components of the i2b2 platform, including:

- Project Management (PM) Web Service
- GridSphere Portal Service
- Clinical Research Chart (CRC) Query Service
- Ontology Service

Data Access Layer

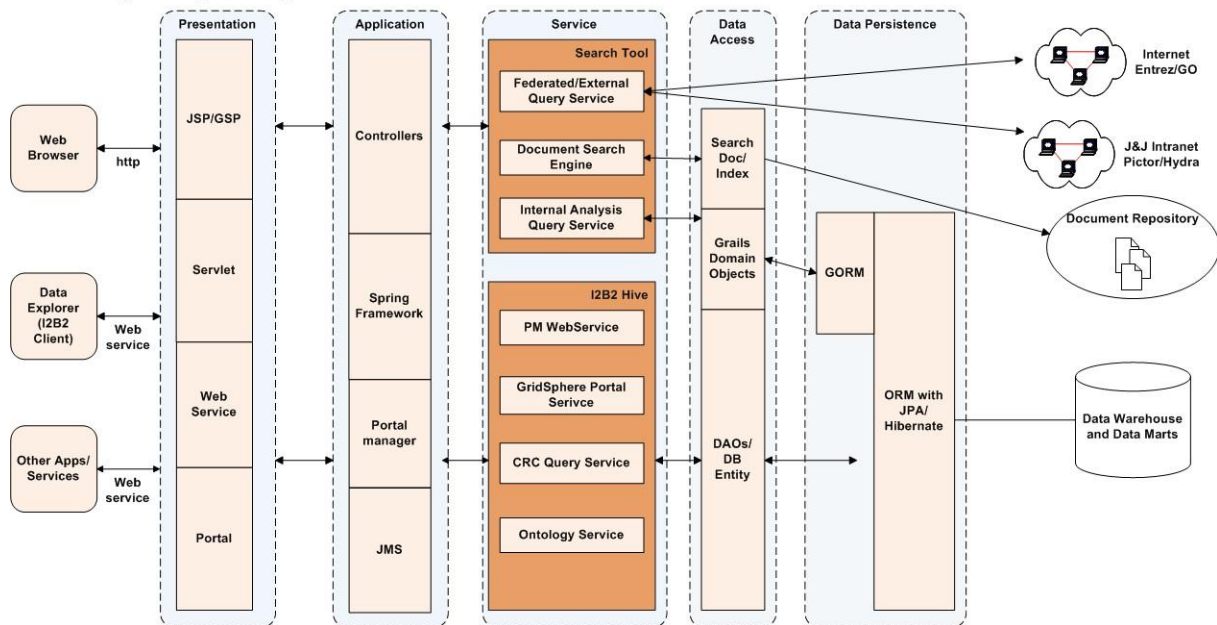
- A Lucene-based document index.
- Grails domain objects for interfacing to interface to persistent data.
- DAO/DB Entities for interfacing between i2b2 services and persistent data.

Data Persistence Layer

Grails Object Relational Mapping (GORM) will be used with Java Persistence API (JPA)/Hibernate Object Relational Mapping (ORM) to interface with the Data Warehouse and Data Mart databases.

Illustration of the Architecture

J&J Knowledge Management Application Architecture Overview



Data Warehouse

The tranSMART data warehouse resides on the Amazon Web Services (AWS) Cloud.

For descriptions of the tables in the core data warehouse, the search data mart, and the Dataset Explorer data mart, see the [tranSMART ETL Analyst's Guide](#).

The following table summarizes the data warehouse servers and the pages where you can find details about each server:

Server Host Name	Description
devapp	Application server for development environment (page 4).
devdb	Database server for development environment (page 7).
tmqa	Database server for staging and QA environment (page 15).
tmsmart	Application server for production environment (page 16).
proddb	Database server for production environment (page 19).
trainingApp	Application server for training environment (page 21).
trainingDB	Database server for training environment (page 21).
backup	Proxy and backup server (page 22).

Development Environment Application Server

External Host Name

<https://devapp.jnj.recomdata.com>

External IP Address

174.129.241.161

Internal IP Address

10.254.177.220

Internal Host Name

domU-12-31-39-00-[AE-12](#).compute-1.internal

Applications

Tomcat, JBoss, i2b2, Hudson, Pathway Studio

Application URLs

- Dev App: <https://devapp.jnj.recomdata.com/transmart> 
- Hudson Build: <https://devapp.jnj.recomdata.com/hudson> 

Server Information

- AMI ID: ami-7ecb2f17
- Instance ID: i-cb8811a2
- Public DNS Name: devapp.jnj.recomdata.com
- IP Address: 174.129.241.161
- Key Name: biomart
- Instance Type: m1.xlarge
- Availability Zone: us-east-1a
- Operating System: Oracle Enterprise Linux v5.1 (2.6.18-53.1.13.9.1.el5xen)
- Security Groups: JNJ_Common, JNJ_Integration

TCP ports opened for inbound traffic:

- 22 (SSH) open to Recombinant Data subnet (75.150.118.80), John Boles (148.177.0.100)

Users

Operating System Users with public key installed and password disabled:

1. root
2. oracle
3. amandel
4. cuhrich
5. dhousman
6. etosch
7. hxia
8. jadler
9. jisikoff
10. jliu
11. krussell

Installed Software

Installed Software under /usr/local

- JDK 1.5.0_17
- JDK 1.6.0_12
- Ant 1.7.1
- Tomcat 5.5.27 (HTTP:7070, AJP: 7009)
- JDK 1.4 Compatibility for Tomcat 5.5.27
- JBoss v4.2.2.GA
- Axis2 v1.1
- sqldeveloper (type **sqldeveloper** from any path)

Installed i2b2 R1.3 final

- URL for Gridsphere: <http://174.129.241.161:7070/gridsphere>
- Two users created from Gridsphere: admin and inforsense, their passwords see Passcodes wiki root/manager
- String for i2b2workbench: I2b2.2=i2b2demo,REST,
<http://67.202.22.37:7070/axis2/rest/PMService/>

Apache Server

Apache server should run as root user.

- Start/Stop:

```
sudo su - root
/etc/init.d/httpd start/stop
```

Passphrase is "biomart".

When asked for the SSL passphrase, type the Unix password.

- Configuration:
 - general configuration file: /etc/httpd/conf/httpd.conf
 - SSL: /etc/httpd/conf.d/ssl.conf
 - Proxy: /etc/httpd/conf.d/proxy.conf
 - AJP proxy: /etc/httpd/conf.d/proxy_ajp.conf. AJP is used to proxy into the tomcat server for the dev app.

- Make sure the following additional RPMs are installed for SSL:
 - mod_ssl-2.2.3-11.el5.0.1.x86_64.rpm
 - distcache-1.4.5-14.1.i386.rpm
 - distcache-1.4.5-14.1.x86_64.rpm

Tomcat Server

The Tomcat server needs to be started with the **appuser** account:

- Dev App and I2B2 Server:
 - /usr/local/tomcat-5.5.27
- Hudson Server
 - /user/local/tomcat-hudson

Development Environment Database Server

External Host Name

https://devdb.jnj.recomdata.com

External IP Address

174.129.237.81

Internal IP Address

10.240.115.47

Internal Host Name

domU-12-31-39-04-6C-C1.compute-1.internal

Applications

Oracle 11g (SID: DW1), eRoom

Server Information

- AMI ID: ami-7ecb2f17
- Instance ID: i-dfbaeeb6
- Public DNS Name: ec2-174-129-237-81.compute-1.amazonaws.com
- IP Address: 174.129.237.81
- Key Name: biomart
- Instance Type: m1.xlarge
- Availability Zone: us-east-1a

- Operating System: Oracle Enterprise Linux v5.1 (2.6.18-53.1.13.9.1.el5xen)
- Security Groups: JNJ_Common, JNJ_Staging_DB

TCP ports opened for inbound traffic:

- 22 (SSH) open to Recombinant Data subnet (75.150.118.80), John Boles (148.177.0.100)

Users

Operating System Users with public key installed and password disabled:

1. root
2. oracle
3. amandel
4. cuhrich
5. dhousman
6. etosch
7. hxia
8. jadler
9. jsikoff
10. jliu
11. krussell

Oracle Tablespaces

1. USERS
2. I2B2_DATA
3. BIOMART
4. BIOMART_LZ
5. BIOMART_USER
6. BIOMART_WZ
7. CENTCLINRD
8. DATA
9. REFERENCE
10. SEARCH_APP
11. CONTROL
12. BIOMKR_D
13. DATASET_EXPLORER
14. MARRAYDAT01
15. INDEX

For more information, see [Tablespace and Directory Details](#) on page 10.

Oracle Database Users:

1. sys
2. system
3. sysman
4. i2b2demodata (migrated from jnjhost2)

5. i2b2metadata (migrated from jnjhost2)
6. i2b2hive (migrated from jnjhost2)
7. i2b2workdata
8. i2b2demodata2
9. i2b2metadata2
10. i2b2workdata2
11. i2b2_lz
12. i2b2_wz
13. OMICSFT_LZ (migrated from jnjhost2)
14. PICTOR (migrated from jnjhost2)
15. BIOMART (migrated from jnjhost2, but triggers need to be recompiled)
16. BIOMART_LZ (migrated from jnjhost2, but procedures need to be recompiled)
17. BIOMART_USER (migrated from jnjhost2)
18. BIOMART_WZ (migrated from jnjhost2)
19. CENTCLINRD (migrated from jnjhost2)
20. JBL_JWB (migrated from jnjhost2)
21. REFERENCE (migrated from jnjhost2)
22. SEARCHAPP (migrated from jnjhost2)
23. SEARCHAPP_USER (migrated from jnjhost2)
24. CONTROL (TBD)
25. DATASET_EXPLORER (TBD)
26. MICROARRAY (TBD)
27. JBL_EXDB (TBD)
28. WEBARRAY (TBD)

Installed Software under /usr/local

- Oracle Database Server 11g Release 1 (11.1.0.6.0)
- JDK 1.6.0_12
- JDK 1.5.0_17
- Ant 1.7.1
- sqldeveloper (type **sqldeveloper** from any path)

Backup

- Use dump with cron job to backup file systems:
 - Sunday: full backup
 - Monday: incremental backup
 - Tuesday: incremental backup
 - Wednesday: incremental backup
 - Thursday: incremental backup
 - Friday: incremental backup
 - Saturday: incremental backup

- Use both RMAN and expdp/impdp with cron job to backup Oracle database:
 - Weekly run expdp to backup databases
 - Use RMAN for database hot backup and archive log files

Oracle Optimizations

- Set [ASYNCHRONOUS IO](#)
- Run [I/O Calibration](#)

Tablespace and Directory Details

External Table space
/u03/biomart/file_mount

Key biomart schemas
-- BIOMART

```
CREATE TABLESPACE BIOMART DATAFILE
  '/u02/oradata/DW1/BIOMART1.DBF'
SIZE 10240M AUTOEXTEND ON NEXT 1024M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;
```

-- BIOMART_LZ

```
CREATE TABLESPACE BIOMART_LZ DATAFILE
  '/u02/oradata/DW1/BIOMART_LZ.DBF'
SIZE 10240M AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
NOLOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;
```

-- BIOMART_USER

```
CREATE TABLESPACE BIOMART_USER DATAFILE
  '/u02/oradata/DW1/BIOMART_USER.DBF'
SIZE 1024M AUTOEXTEND ON NEXT 1M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;
```



```

-- BIOMART_WZ

CREATE TABLESPACE BIOMART_WZ DATAFILE
  '/u02/oradata/DW1/BIOMART_WZ.DBF'
SIZE 10240M AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
NOLOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- CENTCLINRD

CREATE TABLESPACE CENTCLINRD DATAFILE
  '/u02/oradata/DW1/CENTCLINRD.DBF'
SIZE 4096M AUTOEXTEND ON NEXT 1M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- DATA

CREATE TABLESPACE DATA DATAFILE
  '/u01/oradata/DW1/DATA.DBF'
SIZE 4096M AUTOEXTEND ON NEXT 1M MAXSIZE UNLIMITED
NOLOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- REFERENCE

CREATE TABLESPACE REFERENCE DATAFILE
  '/u01/oradata/DW1/REFERENCE.DBF'
SIZE 10240M AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
NOLOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- SEARCH_APP

CREATE TABLESPACE SEARCH_APP DATAFILE
  '/u02/oradata/DW1/SEARCHAPP.DBF'
SIZE 2048M AUTOEXTEND ON NEXT 1M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE

```

Data Warehouse

```
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- CONTROL

CREATE TABLESPACE CONTROL DATAFILE
  '/u01/oradata/DW1/JNJORACLECONTROL.DBF'
SIZE 1024M AUTOEXTEND ON NEXT 1M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- BIOMKR_D

CREATE TABLESPACE BIOMKR_D DATAFILE
  '/u01/oradata/DW1/BIOMKRD.DBF'
SIZE 8096M AUTOEXTEND ON NEXT 1M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- DATASET_EXPLORER

CREATE TABLESPACE DATASET_EXPLORER DATAFILE
  '/u01/oradata/DW1/DATASET_EXPLORER.DBF'
SIZE 1024M AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- JUBILANT

CREATE TABLESPACE JUBILANT DATAFILE
  '/u01/oradata/DW1/JUBILANT.DBF'
SIZE 4096M AUTOEXTEND ON NEXT 1M MAXSIZE UNLIMITED
NOLOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- MARRAYDAT01

CREATE BIGFILE TABLESPACE MARRAYDAT01 DATAFILE
  '/u02/oradata/DW1/MARRAYDAT01.DBF'
SIZE 32G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
```

```
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- WEB_ARRAY_PART1

CREATE BIGFILE TABLESPACE WEB_ARRAY_PART1 DATAFILE
  '/u02/oradata/DW1/WEBARRAY01.DBF'
SIZE 4G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- WEB_ARRAY_PART2

CREATE BIGFILE TABLESPACE WEB_ARRAY_PART2 DATAFILE
  '/u01/oradata/DW1/WEBARRAY02.DBF'
SIZE 4G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- WEB_ARRAY_PART3

CREATE BIGFILE TABLESPACE WEB_ARRAY_PART3 DATAFILE
  '/u01/oradata/DW1/WEBARRAY03.DBF'
SIZE 4G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- WEB_ARRAY_PART4

CREATE BIGFILE TABLESPACE WEB_ARRAY_PART4 DATAFILE
  '/u01/oradata/DW1/WEBARRAY04.DBF'
SIZE 4G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;
```

Data Warehouse

```
-- WEB_ARRAY_PART5

CREATE BIGFILE TABLESPACE WEB_ARRAY_PART5 DATAFILE
  '/u01/oradata/DW1/WEBARRAY05.DBF'
SIZE 4G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- DBADATA_TSP

CREATE BIGFILE TABLESPACE DBADATA_TSP DATAFILE
  '/u01/oradata/DW1/DBADATA_TSP.DBF'
SIZE 4G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- INDEX

CREATE TABLESPACE "INDEX" DATAFILE
  '/u01/oradata/DW1/INDEX01.DBF'
SIZE 1024M AUTOEXTEND ON NEXT 2M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- BIOMART_LZ
CREATE OR REPLACE DIRECTORY BIOMART_LZ AS '/u01/biomart/biomart_lz';

-- FILE_MOUNT

CREATE OR REPLACE DIRECTORY FILE_MOUNT AS '/u01/biomart/file_mount';

-- 04/05/2009

-- DE_USER
CREATE TABLESPACE DE_USER DATAFILE
  '/u02/oradata/DW1/DE_USER.DBF'
SIZE 1024M AUTOEXTEND ON NEXT 1M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;
```

```
-- JNJ_DATA

CREATE TABLESPACE JNJ_DATA DATAFILE
  '/u02/oradata/DW1/JNJ_DATA.DBF'
SIZE 1G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- RESNET

CREATE TABLESPACE RESNET DATAFILE
  '/u02/oradata/DW1/RESNET.DBF'
SIZE 1G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;
```

Staging/QA Database and Application Server

Host Name

tmqa

External IP Address

174.129.234.96

Internal IP Address

10.254.54.127

Internal Host Name

domU-12-31-39-00-31-71.compute-1.internal

Applications

Oracle 11g (SID: tmqa), jboss-4.2.2.GA, tomcat-5.5.27, apache-ant-1.7.1

Server Information

- Instance ID: i-2400144c

Backup

- Use dump with cron job to backup file systems:
 - Sunday: full backup
 - Monday: incremental backup
 - Tuesday: incremental backup
 - Wednesday: incremental backup
 - Thursday: incremental backup
 - Friday: incremental backup
 - Saturday: incremental backup
- Use expdp/impdp with cron job to back up Oracle database every night.

Apache Server

The Apache server should run as root user.

- Start/Stop:

```
sudo su - root
/etc/init.d/httpd start/stop
```

When asked for the SSL passphrase, type the Unix password.

- Configuration:
 - general configuration file: /etc/httpd/conf/httpd.conf
 - SSL: /etc/httpd/conf.d/ssl.conf
 - proxy: /etc/httpd/conf.d/proxy.conf
 - AJP proxy: /etc/httpd/conf.d/proxy_ajp.conf. AJP is used to proxy into the tomcat server for the dev app.

Tomcat Server

The Tomcat server needs to be started with the "appuser" account.

- Dev App and I2B2 server:

/usr/local/tomcat-5.5.27

- Hudson server:

/usr/local/tomcat-hudson

Production Environment Application Server

External Host Name

https://tmsmart.jnj.recomdata.com

External IP Address

174.129.234.170

Internal IP Address

10.248.203.127

Internal Host Name

domU-12-31-39-02-C4-71.compute-1.internal

Applications

Tomcat, JBoss, i2b2

Server Information

- AMI ID: ami-7ecb2f17
- Instance ID: i-cc8118a5
- Public DNS Name: ec2-174-129-234-170.compute-1.amazonaws.com
- IP Address: 174.129.234.170
- Key Name: biomart
- Instance Type: m1.xlarge
- Availability Zone: us-east-1a
- Operating System: Oracle Enterprise Linux v5.1 (2.6.18-53.1.13.9.1.el5xen)
- Security Groups: JNJ_Common, JNJ_Application

TCP ports opened for inbound traffic:

- 22 (SSH) open to Recombinant Data subnet (75.150.118.80), John Boles (148.177.0.100)


Users

Operating System Users with public key installed and password disabled:

1. root
2. oracle
3. amandel
4. cuhrich
5. dhousman
6. etosch
7. hxia

8. jader
9. jisikoff
10. jliu
11. krussell

Installed Software

- Installed Software under /usr/local
 - JDK 1.5.0_17
 - JDK 1.6.0_12
 - Ant 1.7.1
 - Tomcat 5.5.27 (HTTP:7070, AJP: 7009)
 - JDK 1.4 Compatibility for Tomcat 5.5.27
 - JBoss v4.2.2.GA
 - Axis2 v1.1
 - sqldeveloper (type **sqldeveloper** from any path)
- Installed i2b2 R1.3 final
 - URL for Gridsphere: <http://174.129.234.170:7070/gridsphere/gridsphere> 
 - Two users created from Gridsphere: admin and inforsense
 - String for i2b2workbench: I2b2.2=i2b2demo,REST,
<http://174.129.234.170:7070/axis2/rest/PMService/>

Backup

Use dump with cron job:

- Sunday: full backup
- Monday: incremental backup
- Tuesday: incremental backup
- Wednesday: incremental backup
- Thursday: incremental backup
- Friday: incremental backup
- Saturday: incremental backup

Production Environment Database Server

External Host Name

https://proddb.jnj.recomdata.com

External IP Address

174.129.240.115

Internal IP Address

10.249.185.235

Internal Host Name

domU-12-31-39-03-[BA-01](#).compute-1.internal

Applications

Oracle 11g (SID: DW2)

Server Information

- AMI ID: ami-7ecb2f17
- Instance ID: i-1488117d
- Public DNS Name: ec2-174-129-240-115.compute-1.amazonaws.com
- IP Address: 174.129.240.115
- Key Name: biomart
- Instance Type: m1.xlarge
- Availability Zone: us-east-1a
- Operating System: Oracle Enterprise Linux v5.1 (2.6.18-53.1.13.9.1.el5xen)
- Security Groups: JNJ_Common, JNJ_Warehouse_DB

TCP ports opened for inbound traffic:

- 22 (SSH) open to Recombinant Data subnet (75.150.118.80), John Boles (148.177.0.100)

Users

Operating System Users with public key installed and password disabled.

1. root
2. oracle
3. amandel
4. cuhrich
5. dhousman
6. etosch

7. hxia
8. jadler
9. jisikoff
10. jliu
11. krussell

Oracle Database Users:

1. sys
2. system
3. sysman

Installed Software

Installed Software under /usr/local:

- Oracle Database Server 11g Release 1 (11.1.0.6.0)

Backup

- Use dump with cron job to backup file systems:
 - Sunday: full backup
 - Monday: incremental backup
 - Tuesday: incremental backup
 - Wednesday: incremental backup
 - Thursday: incremental backup
 - Friday: incremental backup
 - Saturday: incremental backup
- Use both RMAN and expdp/impdp with cron job to backup Oracle database:
 - Weekly run expdp to backup databases
 - Use RMAN for database hot backup and archive log files

Training Application Server

Host Name

trainingApp

External IP Address

75.101.162.195

Internal IP Address

10.192.170.143

Internal Host Name

domU-12-31-39-0E-A9-61.compute-1.internal

Applications

Tomcat, JBoss, i2b2

Server Information

- Instance ID: i-7fa17614

Training Database Server

Host Name

trainingDB

External IP Address

184.73.224.77

Internal IP Address

10.192.178.207

Internal Host Name

domU-12-31-39-0E-B1-21.compute-1.internal

Applications

Oracle 11g (SID: dw4tn)

Server Information

- Instance ID: i-d35e8bb8

Proxy and Backup Server

External Host Name

https://devapp.jnj.recomdata.com

External IP Address

75.101.128.23

Internal IP Address

10.248.155.79

Server Information

- AMI ID: ami-7ecb2f17
- Instance ID: i-6a8c1503
- Public DNS Name: ec2-75-101-128-23 .compute-1.amazonaws.com
- IP Address: 75.101.128.23
- Key Name: biomart
- Instance Type: m1.xlarge
- Availability Zone: us-east-1a
- Operating System: Oracle Enterprise Linux v5.1 (2.6.18-53.1.13.9.1.el5xen)
Security Groups: JNJ_Common, JNJ_Warehouse_DB

TCP ports opened for inbound traffic:

- 22 (SSH) open to Recombinant Data subnet (75.150.118.80), John Boles (148.177.0.100)

Users

Operating System Users with public key installed and password disabled.

1. root
2. oracle
3. amandel
4. cuhrich
5. dhousman
6. etosch
7. hxia
8. jadler
9. jisikoff
10. jliu
11. krussell

Oracle Database Users

1. sys
2. system
3. sysman

Installed Software

Installed Software under /usr/local

- Oracle Database Server 11g Release 1 (11.1.0.6.0)

Backup

- Use dump with cron job to backup file systems:
 - Sunday: full backup
 - Monday: incremental backup
 - Tuesday: incremental backup
 - Wednesday: incremental backup
 - Thursday: incremental backup
 - Friday: incremental backup
 - Saturday: incremental backup
- Use both RMAN and expdp/impdp with cron job to backup Oracle database:
 - Weekly run expdp to backup databases
 - Use RMAN for database hot backup and archive log files

The eRoom

The eRoom is a transfer point for exchanging data files and other large files between ETL Analysts, Curation Analysts, Data Stewards, subject-matter experts, and others who may be involved in the process of moving raw data files into the data warehouse.

Users who need to upload a file to the eRoom must use the eRoom's private key through sftp or scp from a J&J-approved network or IP address.

The following users have been sent the eRoom private key:

- Jonathan Cornibe
- Robert Gruninger
- Tania Khasanova
- Venkata Koka

eRoom Specifications

Specification	Description
eRoom host IP server	174.129.237.81 (devDB server)
eRoom directory	/eroom
Directory owner	eroom
Permissions	<ul style="list-style-type: none">▪ eRoom owner (user eroom) has Read/Write▪ Users in the eRoom group have Read only▪ No other users have access permission
eRoom group	TBD

Chapter 2

Oracle Operations

The login credentials for Oracle on `jjhost2` are the following:

- User: `system`
- Password: `manager`

Configuration

The following spreadsheet contains the current comprehensive configuration parameter settings for the JNJORACL Database (JNJORACLE service). The more significant parameters are highlighted in yellow.

[injoracleparametersettings.xlsx](#)

This is a living document that should be updated when database parameters are changed. The file can easily be updated by connecting to the database in SQL Developer as the user SYSTEM and executing the following query:

```
Select * from v$parameter;
```

Press F5 to output the data in a text format, save it to a file, then pull the information into the spreadsheet (or copy/paste).

The majority of these parameters reflect default settings with some specific settings where necessary (control file locations, service_names, sga_target, for example).

Performance Pack Setup

1. Sign into `jjhost` using terminal services.
2. Open either IE or the Firefox browser and type the url: <http://jjhost2:5502/em>
3. Sign in using the username/password of `sys/manager` and select 'as sysdba' from the drop down list box.
4. At the top of the page under preferences, click the 'Management Pack Access' link and check the 'Pack Access Agreed' box after selecting the 'All Targets' radio button in the 'View Options' section.

This enables the Automatic statistics gathering features and snapshot captures.

Rman Backup Configuration

1. Under the 'Availability' link, set the backup settings to use compressed backups with a parallelism of 4 (4cpu's on jjhost2)
2. Set the default backup location to be the same as the 'recovery_file_destination' parameter, which is currently configured to 'G:\Backups\ORADATA.'
3. On the same 'Availability' page, set a daily backup to back up to the above location.

Note: Other changes may be required to conform to formatting standards.

Schema Deployment Between Servers

Use the following steps to deploy schemas on the database servers – from the development server to the staging server, and from the staging server to the production server.

For security reasons, the scp with SSH key-based authentication are used.

Step 1. Generate a Pair of Keys

You will log into the three Amazon Web Services servers:

Server and Hostname	External IP Address	Internal IP Address
Development devdb.jnj.recomdata.com	174.129.237.81	10.240.115.47
Staging tmqa.jnj.recomdata.com	174.129.234.96	10.254.54.127
Production proddb.jnj.recomdata.com	174.129.240.115	10.249.185.235

Log in as the user `oracle`, and use the `ssh-keygen` command. You should run it in the command line. You will be asked for a file in which the key should be saved to and for a passphrase (password) for the key.

For example, in the development database server, run the command:

```
oracle@174.129.237.81:DevDB:[/home1/oracle]$ssh-keygen -t rsa

Generating public/private rsa key pair.
Enter file in which to save the key (/home1/oracle/.ssh/id_rsa):
Created directory '/home1/oracle/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home1/oracle/.ssh/id_rsa.
Your public key has been saved in /home1/oracle/.ssh/id_rsa.pub.
The key fingerprint is:
83:6d:c5:73:a9:d6:3c:61:21:f0:2d:72:ac:81:20:38 oracle@domU-12-31-39-
04-6C-C1
```

This will create a private key written to `/home1/oracle/.ssh/id_rsa`, and a public key `/home1/oracle/.ssh/id_rsa.pub`. The passphrase is used to protect your key.

If you enter a passphrase here, you will be asked for it when you connect via SSH or SCP. For convenience, you should leave it blank.

You should be able to use the same command to create a pair of keys for all three database servers under the `oracle` account.

Step 2. Configure SSH Key-based Authentication

1. Configure the production database server to use a key to access the development database server.

Log into the production database server as the user `oracle`, create a file `authorized_keys` in the directory `~/.ssh` (here is `/home/oracle/.ssh`) if not exists, and copy the content of development database server's `oracle` user's public key `id_rsa.pub` into the `authorized_keys`.

2. Configure the staging database server to use a key to access the development database server.

Log into the staging database server as `oracle` user, create a file `authorized_keys` in the directory `~/.ssh` (here is `/home1/oracle/.ssh`) if not exists, and copy the content of development database server's `oracle` user's public key `id_rsa.pub` into the `authorized_keys`.

3. Validate the key-based authentication.

The command `ssh` can be used to validate whether the key-based authentication is correct. If the configuration is correct, when you open an SSH session to the remote system, you should be automatically logged in:

```
oracle@174.129.240.115:ProdDB:[/home/oracle/.ssh]$ssh 174.129.237.81

Address 174.129.237.81 maps to ec2-174-129-237-81.compute-
1.amazonaws.com, but this does not map back to the address -
POSSIBLE BREAK-IN ATTEMPT!
```

```
Last login: Tue Jul 28 08:43:47 2009 from 174.129.240.115

oracle@174.129.234.96:tmqa:[/home1/oracle]$ssh 174.129.237.81

Address 174.129.237.81 maps to ec2-174-129-237-81.compute-
1.amazonaws.com, but this does not map back to the address -
POSSIBLE BREAK-IN ATTEMPT!
Last login: Tue Jul 28 09:06:24 2009 from 174.129.234.96
```

Step 3. Install Perl Modules

The following steps need to be performed by `root`:

1. Set up CPAN update environment.

Enter the command and then follow the on-screen instructions:

```
# perl -MCPAN -eshell
... ..
cpan> get DBI
cpan> get DBD::Oracle
cpan> exit
```

Set up environment variables to compile and install DBI and DBD::Oracle modules.

```
# export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1
# export ORACLE_SID=<Oracle SID>
# export ORACLE_USER=<Oracle tester>/<Oracle tester passwd>
# export LD_LIBRARY_PATH=$ORACLE_HOME/lib:$ORACLE_HOME
```

2. Install DBI.

Change the current directory to DBI download source directory:

```
# cd /root/.cpan/build/DBI-1.609
```

Execute the following command:

```
# perl Makefile.PL
# make
# make test
```

If the test results look good, install it:

```
# make install
```

3. Install DBD::Oracle.

Change the current directory to DBI download source directory:

```
# cd /root/.cpan/build/ DBD-Oracle-1.23
```

Execute the following command:

```
# perl Makefile.PL
# make
# make test
```

If the test results look good, install it:

```
# make install
```

4. Test the installations.

The following can be used to test whether the installation of DBI and DBD::Oracle are working.

```
use DBI;
use DBD::Oracle qw(:ora_types ORA_OCI);
use strict;
use vars qw($dbh $sth);

# get a database handle
$dbh = DBI->connect('dbi:Oracle:host=localhost;sid=<SID>',
                  '<User>', '<Password>');

# get a statement handle
$sth=$dbh->prepare('SELECT table_name FROM cat');

# execute the statement handle
$sth->execute();

# loop through the results
while (my ($table) = $sth->fetchrow())
{
    print " $table \n";
}

$sth->finish();
$dbh->disconnect();
```

If a list of table names are displayed after you run the above script, the installation is successful. Otherwise, fix any errors and reinstall.

Step 4. Create Oracle Directories for Export and Import

1. Create an Oracle directory for exporting from the development server.

Log in as `oracle`, and log in the development database as the user `system`:

```
$ sqlplus system/xxx
SQL> create directory BKUP as '/backup/database';
SQL> grant read, write on directory BKUP to public;
```

The physical directory `/backup/database` should be created using the operating system's command.

2. Create an Oracle directory for importing to the staging server.

Log in as `oracle`, and log in the development database as the user `system`:

```
$ sqlplus system/xxx
SQL> create directory DW1 as '/backup/dw1dump/dump';
SQL> grant read, write on directory DW1 to public;
```

The physical directory `/backup/dw1dump/dump` should be created using the operating system's command.

3. Create an Oracle directory for importing to the production server.

Log in as `oracle`, and log in the development database as the user `system`:

```
$ sqlplus system/<password>
SQL> create directory BKUP as '/backup/database';
SQL> grant read, write on directory BKUP to public;
```

The physical directory `/backup/database` should be created using the operating system's command.

Step 5. Export Data from the Development Server

Use Oracle's `expdp` program to export schema data from the development database.

```
$ expdp system/<password>
    dumpfile=<Oracle directory name>:<dump file name>
    logfile=<Oracle directory name>:<log file name>
    schemas=<list of schemas>
```

For example, the following command will export the development database's `deapp` schema with data into the file `deapp_2009-07-28.dump` under `/backup/database` (Oracle directory `BKUP`):

```
$ expdp system/xxx dumpfile=BKUP:deapp_2009-07-28.dump
    logfile=BKUP:deapp_2009-07-28.log schemas=deapp
```

Step 6. Move Exported File(s) from the Source Server to the Target

Use `SCP` to move dumped schema files from the development server to the staging server or to the production server:

```
$scp <source server IP>:<source path>/<source dump file>
    <destination path>/.
```

For example, the following command will `scp` the dump file `deapp_2009-07-28.dump` from the development database server's `/backup/database` to the staging database server's `/backup/dw1dump/dump` directory.

```
$ scp 10.240.115.47:/backup/database/deapp_2009-07-28.dump
    /backup/dw1dump/dump/.
```

Step 7. Prepare for Schema Import

All existing sequence names need to be renamed or dropped.

Step 8. Import Source Schemas into the Target Database

Use Oracle's `impdp` program to import schema data from the development database:

```
$ impdp system/<password>
  dumpfile=DW1:<dump file name>
  logfile=DW1:<log file name>
  schemas=<list of schemas>
  table_exists_action=replace
```

For example, the following command will export the development database's `deapp` schema with data into the file `deapp_2009-07-28.dump` under `/backup/database` (Oracle directory `BKUP`):

```
$ expdp system/xxx dumpfile=BKUP:deapp_2009-07-28.dump
  logfile=BKUP:deapp_2009-07-28.log schemas=deapp
```

Step 9. Deployment Error Prevention

1. Check whether any indexes are invalid.

Check all imported schema's indexes and rebuild them if they are invalid.

2. Rebuild the schema `searchapp`'s materialized view.

Drop and rebuild the materialized view `SEARCH_BIO_MKR_CORREL_FAST_MV`, and also drop and recreate the synonym for `biomart_user`.

The following scripts are run once the schema `searchapp` is deployed:

```
DROP SYNONYM BIOMART_USER.SEARCH_BIO_MKR_CORREL_FAST_MV;

CREATE SYNONYM BIOMART_USER.SEARCH_BIO_MKR_CORREL_FAST_MV FOR
SEARCHAPP.SEARCH_BIO_MKR_CORREL_FAST_MV;

DROP MATERIALIZED VIEW SEARCHAPP.SEARCH_BIO_MKR_CORREL_FAST_MV;

CREATE MATERIALIZED VIEW SEARCHAPP.SEARCH_BIO_MKR_CORREL_FAST_MV
TABLESPACE SEARCH_APP
CACHE
LOGGING
NOCOMPRESS
NOPARALLEL
BUILD IMMEDIATE
USING NO INDEX
REFRESH COMPLETE ON COMMIT
WITH PRIMARY KEY
```

```

AS
SELECT
    i.SEARCH_GENE_SIGNATURE_ID AS domain_object_id,
    i.BIO_MARKER_ID AS asso_bio_marker_id,
    'GENE_SIGNATURE_ITEM' AS correl_type,
    CASE
        WHEN i.FOLD_CHG_METRIC IS NULL THEN 1
        ELSE i.FOLD_CHG_METRIC
    END AS value_metric,
    3 AS mv_id
FROM
    SEARCH_GENE_SIGNATURE_ITEM i,
    SEARCH_GENE_SIGNATURE gs
WHERE
    i.SEARCH_GENE_SIGNATURE_ID = gs.SEARCH_GENE_SIGNATURE_ID AND gs.
        DELETED_FLAG = 0;

COMMENT ON MATERIALIZED VIEW SEARCHAPP.SEARCH_BIO_MKR_CORREL_FAST_MV
IS
    'expose transactional domain objects to search framework (gene
signature for now)';

CREATE INDEX SEARCHAPP.SEARCH_BIO_MKR_CORREL_MV_IDX ON
SEARCHAPP.SEARCH_BIO_MKR_CORREL_FAST_MV
(DOMAIN_OBJECT_ID, CORREL_TYPE)
NOLOGGING
TABLESPACE SEARCH_APP
NOPARALLEL;

```

Future Enhancements

The following functions are not implemented:

- Table-based deployment.

The current implementation can only be used to deploy a schema, not an individual table or tables.

- Review and display possible errors generated in export or import stage.

All possible errors created in the export or import stage can only be reviewed manually at the current implementation.

Scripts to Facilitate the Deployment Process

Configure file used in the deployment (deployment.conf):

```
#####
##
# If you make any changes on Oracle Directory, please modify here
# accordingly, the schema deployment will fail.
#
# Note:
# 1. *_DUMP_PATH is case sensitive
# 2. make sure user's passwords are in low case
#
#####
#
# devDB:
# IP: 10.240.115.47 or 174.129.237.81
# SID: DW1
# Directory:
#     BKUP -> /backup/database
#     DMP -> /dpdump/dump
#     DATA_PUMP_DIR -> /data/dpdump
#
# prodDB:
# IP: 10.249.185.235 or 174.129.240.115
# SID: DW2
# Directory:
#     TM -> /transmart/dump
#     DMP -> /dwldump/dump
#     DATA_PUMP_DIR -> /backup/dpdump
#
# tmqa:
# IP: 10.254.54.127 or 174.129.234.96
# SID: tmqa
# Directory:
#     DMP -> /transmart/dpdump
#     DATA_PUMP_DIR -> /backup/dpdump
#####
#

ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

# parameters for source schemas
SOURCE_IP=10.240.115.47
SOURCE_SID=DW1
SOURCE_DIRECTORY_NAME=BKUP
SOURCE_DUMP_PATH=/backup/database

# parameters for target schemas
TARGET_IP=10.254.54.127
TARGET_SID=tmqa
TARGET_DIRECTORY_NAME=DMP
TARGET_DUMP_PATH=/transmart/dpdump
```

```
#####
# Database deployments can be in schema level, table level
# or mixed with both
#####

# parameter for schema level deployment
# <source schema1>::<target schema1>, <source schema2>::<target
schema2>, ...
SCHEMAS=i2b2demodata::i2b2demodata, i2b2hive::i2b2hive,
searchapp::searchapp

# parameter for table level deployment
# This level is not implemented in this version, if requested,
# this could be good exercise for co-development
#TABLES=DEPLOY:OBSERVATION_FACT::DEPLOY1:OBSERVATION_FACT,
DEPLOY:CONCEPT_DIMENSION::DEPLOY1:CONCEPT_DIMENSION
```

The deployment program

```
use DBI;
use strict;

# Identify running Operating System
# Windows -> "MSWin" Linux -> "linux"
my $os = $^O;

#####
# Read in the deployment configure file from
# the current directory, so this program should
# be launched from the installation directory and
# as the user oracle
#####

my %config = &configure();
foreach my $key (keys %config){
    #print "$key  $config{$key} \n";
}

my (%src_dbh, %trgt_dbh) = ();

# timestamp used for file name
my $dt = &getTimestamp();

# get Oracle Home from the configure file
my $ORACLE_HOME = $config{"ORACLE_HOME"};

# get parameters for source database server
$src_dbh{"host"} = $config{"SOURCE_IP"};
$src_dbh{"sid"} = $config{"SOURCE_SID"};
$src_dbh{"directory"} = $config{"SOURCE_DIRECTORY_NAME"};
$src_dbh{"port"} = "1521";
$src_dbh{"srcdir"} = $config{"SOURCE_DUMP_PATH"};

# get parameters for target database server
$trgt_dbh{"host"} = $config{"TARGET_IP"};
$trgt_dbh{"sid"} = $config{"TARGET_SID"};
$trgt_dbh{"directory"} = $config{"TARGET_DIRECTORY_NAME"};
```



```

$trgt_dbh{"port"} = "1521";
$trgt_dbh{"trgtmdir"} = $config{"TARGET_DUMP_PATH"};
$trgt_dbh{"srcdir"} = $config{"SOURCE_DUMP_PATH"};
$trgt_dbh{"srchost"} = $config{"SOURCE_IP"};

#####
# schema level deployment: one by one
#####

print "Starting the schema level deployment process .... \n";
print "    Source Database Server: $src_dbh{'host'} \n";
print "    Source Database SID: $src_dbh{'sid'} \n";
print "    Source Dump File Localtion: $src_dbh{'srcdir'} \n";
print "\n";

print "    Tagart Database Server: $trgt_dbh{'host'} \n";
print "    Target Database SID: $trgt_dbh{'sid'} \n";
print "    Target Dump File Localtion: $trgt_dbh{'trgtmdir'} \n";
print "\n";

print "If this is not what you are anticipated, \n ";
print " please enter \"Ctrl + C\" to terminate this process. \n\n";

sleep 1;

# extract schemas from the configuration file
my $schemas = $config{"SCHEMAS"};
my @schemas = split(/,/ , $schemas);
for(my $i=0; $i<@schemas; $i++){
    # Remove all space from schema name
    $schemas[$i] =~ s/ //g;
    my ($srcSchema, $trgtSchema) = split(/::/, $schemas[$i]);

    print "Staring the deployment: \n";
    print "    $src_dbh{'sid'}:$srcSchema --> $trgt_dbh{'sid'}:$trgtSchema \n";

    # connecting to source database
    $src_dbh{"user"} = $srcSchema;
    my $src_dbh = &createDBH(\%src_dbh);

    # exporting schema from source database
    print "    Start exporting $src_dbh{'sid'}:$srcSchema .... \n";
    &exportSchema(\%src_dbh);
    print "    Exporting $src_dbh{'sid'}:$srcSchema is done. \n\n";

    #####
    # 1. move schemas's dump file to the
    #    target database server
    # 2. connecting to target database
    # 3. rename all existing sequences
    # 4. import schema
    # 5. rebuild all invalid indexes
    # 6. rebuild materialized view
    #####

```

```

    $trgt_dbh{"user"} = $trgtSchema;

# scp source schema's dump file from the source database
# server to the target database server
print "  Start moving the dump file for schema $srcSchema... \n";
&scpSchemaDump(\%trgt_dbh);
print "  Move the dump file for schema $srcSchema is done. \n\n";

# create database handler
my $trgt_dbh = &createDBH(\%trgt_dbh);

# rename sequences
print "  Renaming the target schema's sequences ... \n";
&renameSequenceOld($trgt_dbh);
print "  Rename the target schema's sequences is done. \n\n";

# importing a schema into the target database
print "  Start importing the dump file into the schema
$trgtSchema... \n";
&importSchema(\%trgt_dbh);
print "  Import the dump file into the schema $trgtSchema is
done.\n\n";

# drop old sequence if need or left for checking
&dropOldSequence($trgt_dbh);
&renameOldSequence($trgt_dbh);

# rebuild invalid indexes
print "  Rebuilding all invalid indexes in the imported schema ...
\n";
&rebuildInvalidIndex($trgt_dbh);
print "  Rebuild all invalid indexes is done. \n\n";

# rebuild Materialized View if the schema is searchapp
if(lc($trgtSchema) =~ /searchapp/){
    print "  Rebuilding Materialized Views ... \n";
    &rebuildMaterializedView($trgt_dbh);
    print "  Rebuild Materialized Views is done. \n\n";
}
}

#####
# Table level deployment: one by one
#   if this function is requested,
#       it can be used as a co-development
#####

my $tables = $config{"TABLES"};
my @tables = split(/,/ , $tables);
for(my $j=0; $j<@tables; $j++){
    # Remove all space from schema name
    $tables[$j] =~ s/ //g;
    my ($srcTable, $trgtTable) = split(/::/, $tables[$j]);
    print "$srcTable, $trgtTable \n";
}

```

```
#####
# prepare a schema for deployment or update:
#   attach "_OLD" to each Sequence
#####

sub renameSequenceOld
{
    my $dbh = $_[0];
    my ($sql, $sth, $sql1, $sth1, $flag) = ();

    $sql = "select table_name from cat where table_type='SEQUENCE'";
    $sth = $dbh->prepare($sql);
    $sth->execute();
    while(my ($seq) = $sth->fetchrow()){
        $flag = 1;
        my $seq1 = $seq."_OLD";
        print "\t Rename $seq to $seq1 ... \n";
        $sql1 = "rename $seq to $seq1";
        $sth1 = $dbh->prepare($sql1);
        $sth1->execute();
    }
    if($flag) {
        $sth1->finish();
    }
    $sth->finish();
}

#####
# Remove "_OLD" from each Sequence
#   for testing purpose only
#####

sub renameOldSequence
{
    my $dbh = $_[0];
    my ($sql, $sth, $sql1, $sth1, $flag) = ();

    $sql = "select table_name from cat where table_type='SEQUENCE'";
    $sth = $dbh->prepare($sql);
    $sth->execute();
    while(my ($seq) = $sth->fetchrow()){
        $flag = 1;
        my $seq1 = $seq;
        $seq1 =~ s/_OLD//;
        print "\t rename $seq to $seq1 ... \n";
        $sql1 = "rename $seq to $seq1";
        $sth1 = $dbh->prepare($sql1);
        $sth1->execute();
    }
    if($flag){
        $sth1->finish();
    }
    $sth->finish();
}

```

```
#####
# After a schema is replaced, drop sequences with "_OLD"
#####

sub dropOldSequence
{
    my $dbh = $_[0];
    my ($sql, $sth, $sql1, $sth1, $flag) = ();

    $sql = "select table_name from cat
            where table_type='SEQUENCE' and table_name like '%_OLD'";
    $sth = $dbh->prepare($sql);
    $sth->execute();
    while(my ($seq) = $sth->fetchrow()){
        $flag = 1;
        print "\t Drop retired sequence $seq ... \n";
        $sql1 = "drop sequence $seq";
        $sth1 = $dbh->prepare($sql1);
        $sth1->execute();
    }
    if($flag){
        $sth1->finish();
    }
    $sth->finish();
}

#####
# After a schema deployed, rebuild any invalid indexes
#####

sub rebuildInvalidIndex
{
    my $dbh = $_[0];
    my ($sql, $sth, $sql1, $sth1, $flag) = ();

    $sql = "select table_name, index_name from user_indexes
            where status='INVALID' and index_type!='LOB'";
    $sth = $dbh->prepare($sql);
    $sth->execute();
    while(my ($table, $index) = $sth->fetchrow()){
        $flag = 1;
        print "\t Index $index in $table is invalid, rebuilding ...
\n";
        $sql1 = "alter index $index rebuild";
        $sth1 = $dbh->prepare($sql1);
        $sth1->execute();
        print "\t Rebuild index $index is done. \n";
    }
    if($flag){
        $sth1->finish();
    }
    $sth->finish();
}
```

```
#####
# After a schema is replaced, rebuild the materialized view
#####

sub rebuildMaterializedView
{
    my $dbh = $_[0];
    my ($sql, $sth, $sql1, $sth1, $flag) = ();

    # drop the synonym for the materialized view:
    #     SEARCH_BIO_MKR_CORREL_FAST_MV
    $sql = "DROP SYNONYM BIOMART_USER.SEARCH_BIO_MKR_CORREL_FAST_MV";
    $sth = $dbh->prepare($sql);
    $sth->execute();

    # drop the materialized view: SEARCH_BIO_MKR_CORREL_FAST_MV
    $sql = "DROP MATERIALIZED VIEW
SEARCHAPP.SEARCH_BIO_MKR_CORREL_FAST_MV";
    $sth = $dbh->prepare($sql);
    $sth->execute();

    # create the materialized view: SEARCH_BIO_MKR_CORREL_FAST_MV
    $sql = "CREATE MATERIALIZED VIEW
SEARCHAPP.SEARCH_BIO_MKR_CORREL_FAST_MV ";
    $sql .= "TABLESPACE SEARCH_APP ";
    $sql .= "CACHE ";
    $sql .= "LOGGING ";
    $sql .= "BUILD IMMEDIATE ";
    $sql .= "USING NO INDEX ";
    $sql .= "REFRESH COMPLETE ON COMMIT ";
    $sql .= "WITH PRIMARY KEY ";
    $sql .= "AS ";
    $sql .= "SELECT ";
    $sql .= "    i.SEARCH_GENE_SIGNATURE_ID AS domain_object_id,";
    $sql .= "    i.BIO_MARKER_ID AS asso_bio_marker_id,";
    $sql .= "    'GENE_SIGNATURE_ITEM' AS correl_type,";
    $sql .= "    CASE ";
    $sql .= "        WHEN i.FOLD_CHG_METRIC IS NULL THEN 1 ";
    $sql .= "        ELSE i.FOLD_CHG_METRIC ";
    $sql .= "    END AS value_metric, ";
    $sql .= "    3 AS mv_id ";
    $sql .= "FROM ";
    $sql .= "    SEARCH_GENE_SIGNATURE_ITEM i, ";
    $sql .= "    SEARCH_GENE_SIGNATURE gs ";
    $sql .= "WHERE ";
    $sql .= "    i.SEARCH_GENE_SIGNATURE_ID =
gs.SEARCH_GENE_SIGNATURE_ID AND ";
    $sql .= "    gs.DELETED_FLAG = 0";
    $sth = $dbh->prepare($sql);
    $sth->execute();

    # create a comment for the materialized view:
    #     SEARCH_BIO_MKR_CORREL_FAST_MV
    $sql = "COMMENT ON MATERIALIZED VIEW
SEARCHAPP.SEARCH_BIO_MKR_CORREL_FAST_MV IS ";
    $sql .= "'expose transactional domain objects to search framework
(gene signature for now)'"';

```

```

$sth = $dbh->prepare($sql);
$sth->execute();

# create index for the materialized view:
#     SEARCH_BIO_MKR_CORREL_FAST_MV
$sql = "CREATE INDEX SEARCHAPP.SEARCH_BIO_MKR_CORREL_MV_IDX ";
$sql .= " ON SEARCHAPP.SEARCH_BIO_MKR_CORREL_FAST_MV ";
$sql .= " (DOMAIN_OBJECT_ID, CORREL_TYPE) ";
$sql .= " NOLOGGING TABLESPACE SEARCH_APP";
$sth = $dbh->prepare($sql);
$sth->execute();

# create a synonym for the materialized view:
#     SEARCH_BIO_MKR_CORREL_FAST_MV
$sql = "CREATE SYNONYM BIOMART_USER.SEARCH_BIO_MKR_CORREL_FAST_MV
";
$sql .= " FOR SEARCHAPP.SEARCH_BIO_MKR_CORREL_FAST_MV";
$sth = $dbh->prepare($sql);
$sth->execute();
}

#####
# Read configuration file and return a hash table
#####

sub configure
{
    my %config;

    open(IN, "./deployment.conf") || die "$! \n";
    while(<IN>){
        if(($_ =~ /=/) && ($_ !~ /\#/)){
            # if OS is Windows
            if($os =~ /MSWin/){
                chomp;
            }elseif($os =~ /linux/){
                $_ =~ s/\r\n//;
            }else{
                print "Need to find a way removing newline \n";
            }

            my ($key, $val) = split(/=/, $_);
            $config{$key} = $val;
        }
    }
    close(IN);
    return %config;
}

```

```
#####
#   Create a db handler
#####

sub createdb
{
    my ($ref) = @_ ;

    my $host = $$ref{"host"};
    my $port = $$ref{"port"};
    my $sid = $$ref{"sid"};
    my $user = $$ref{"user"};
    my $passwd = $$ref{"passwd"};

    if(!$port){
        $port = "1521";
    }

    if(!$passwd){
        $passwd = lc($user);
    }

    my $conn = "dbi:Oracle:host=$host;port=$port;sid=$sid";
    return DBI->connect($conn,$user,$passwd);
}

#####
#   export a source database's schema
#####

sub exportSchema
{
    my ($ref) = @_ ;

    my $host = $$ref{"host"};
    my $sid = $$ref{"sid"};
    my $user = lc($$ref{"user"});
    my $dir = $$ref{"directory"};

    my $expdp = "$ORACLE_HOME/bin/expdp system/rec0m23\@$sid
schemas=$user";
    my $dump = $user."_".$dt.".dump";
    my $log = $user."_".$dt.".log";

    my $cmd;
    if($dir =~ /DATA_PUMP_DIR/){
        $cmd = "$expdp dumpfile=$dump logfile=$log";
    } else{
        $cmd = "$expdp dumpfile=$dir:$dump logfile=$dir:$log";
    }
    print "   Run command: $cmd \n";
    system($cmd);
}

```

```
#####
#  import a schema in the target database
#####

sub importSchema
{
    my ($ref) = @_ ;
    my $host = $$ref{"host"};
    my $sid = $$ref{"sid"};
    my $user = lc($$ref{"user"});
    my $dir = $$ref{"directory"};
    my $impdp = "$ORACLE_HOME/bin/impdp system/rec0m23\@$sid";
    $impdp .= " schemas=$user table_exists_action=replace";
    my $dump = $user."_".$dt.".dump";
    my $log = $user."_".$dt.".log";
    my $cmd;
    if($dir =~ /DATA_PUMP_DIR/){
        $cmd = "$impdp dumpfile=$dump logfile=$log";
    } else{
        $cmd = "$impdp dumpfile=$dir:$dump logfile=$dir:$log";
    }
    print "  Run command: $cmd \n";
    system($cmd);
}

#####
#  return the current timestamp
#####

sub getTimestamp
{
    my $date = &getDate();
    my $time = &getTime();
    my $timestamp = $date.$time;
    return $timestamp
}

#####
#  return the current system time
#####

sub getDate
{
    my ($day, $month, $year) = (localtime) [3,4,5];
    $year = 1900 + $year;
    $month = $month + 1;

    if(length($month)<2){
        $month = "0".$month;
    }

    if(length($day)<2){
        $day = "0".$day;
    }

    my $date = $year.$month.$day;
    $date =~ s/^20//;
    return $date;
}
```



```
#####
# return the current system date
#####

sub getTime
{
    my ($sec, $min, $hrs) = (localtime) [0,1,2];

    if(length($sec)<2){
        $sec = "0".$sec;
    }
    if(length($min)<2){
        $min = "0".$min;
    }
    if(length($hrs)<2){
        $hrs = "0".$hrs;
    }
    #my $time = $hrs.$min.$sec;
    my $time = $hrs.$min;
    return $time;
}

#####
# scp schema dump file from the source database server
# to the target database
#####

sub scpSchemaDump
{
    my ($ref) = @_ ;

    my $host = $$ref{"host"};
    my $sid = $$ref{"sid"};
    my $user = lc($$ref{"user"});
    my $trgtdir = $$ref{"trgtdir"};

    my $srchost = $$ref{"srchost"};
    my $srcdir = $$ref{"srcdir"};

    my $scp = "/usr/bin/scp";
    my $dump = $user."_".$dt.".dump";
    my $log = $user."_".$dt.".log";

    my $cmd1 = "$scp $srchost:$srcdir/$dump $trgtdir/.";
    print " Run command: $cmd1 \n";
    system($cmd1);

    my $cmd2 = "$scp $srchost:$srcdir/$log $trgtdir/.";
    print " Run command: $cmd2 \n";
    system($cmd2);
}

#####
# Possible co-development efforts for the following areas:
# 1. table level deployment
# 2. import/export log file check
# 3. schema or table comparison
#####
```

Using Oracle impdp and expdp

Tips

- For a full list of settings, type `impdp help=yes` or `expdp help=yes`.
- Check out <http://www.orafaq.com/wiki/Datapump>.
- The location of the file for import or export cannot be set directly in the utility. It uses a parameter loaded into Oracle for security reasons (stored in `dba_directories.directory_name`).
- Use the `VERSION` parameter to move data from different versions of ORACLE instances – for example, from 11 to 10.2.
- To change the Tablespace of a dumpfile, use:

```
REMAP_TABLESPACE=source_tablespace:target_tablespace
```

 - The `source_tablespace` does not have to exist on the target.
 - This feature is available starting in 10g.

Schema and Table Export

The following procedure exports data from the JNJ Dev server to a local Oracle Express Edition:

1. Decide where the dumpfile should land on the hard drive:

```
select * from dba_directories
```

The `DIRECTORY_NAME` is what is used for the 'DIRECTORY' Parameter: In this case, 'BACKUPS' was used, which was set up on the Dev Server. For your local instance it would most likely be: `DATA_PUMP_DIR`.

The directory points to where the file will be created.

2. From a command line, export the table or schema that you want.

```
TABLE: expdp system/manager DIRECTORY=BACKUPS  
DUMPFILE=KR_TEST_TABLE.dmp TABLES=biomart.bio_assay
```

```
SCHEMA: expdp system/manager DIRECTORY=BACKUPS  
DUMPFILE=KR_TEST_SCHEMA.dmp SCHEMAS=BIOMART
```

3. Copy the file to the machine hosting the server where you want to load the data:

(From `d:\datadump\` on `jjhost2` to `C:\oracle\app\oracle\admin\XE\dpdump\` on my local machine.)

The local directory was set in parameter 'DATA_PUMP_DIR' during installation.

Importing the Data

1. Create the needed Tablespace on the Database. (Same as the original.)
In this case:

```
CREATE TABLESPACE BIOMKR_D
DATAFILE 'biomkr_d.dbf' SIZE 20M
NO LOGGING;
```

Note: If only loading the single table, a user needs to be created called 'biomart'. The schema load created it automatically.

2. Create the objects:

```
TABLE: impdp system/password DIRECTORY=DATA_PUMP_DIR
DUMPFILE=KR_TEST_TABLE.dmp
```

```
SCHEMA: impdp system/password DIRECTORY=DATA_PUMP_DIR
DUMPFILE=KR_TEST_SCHEMA.dmp
```

Note: A few errors will be reporting during the load due to reference to nonexistent constraints or tables. You can ignore these.

Example of ignoring grants (no errors):

```
SCHEMA: impdp system/password DIRECTORY=DATA_PUMP_DIR
DUMPFILE=KR_TEST_SCHEMA.dmp EXCLUDE=ROLE_GRANT
```

Additional Information

If you run impdp with the SQLFILE parameter, you will get a list of all DDL commands that are executed to import the database. Looking through the SQL for the most recent dump file, it appears that you should create two tablespaces as shown below:

```
create tablespace BIOMART
datafile 'biomart.dbf'
size 20m
autoextend on
next 20m maxsize 1024m;

create tablespace SEARCH_APP
datafile 'searchapp.dbf'
size 20m
autoextend on
next 20m maxsize 1024m;
```

The only other tablespace is the BIOMART_WZ, and that can be mapped to the BIOMART using the REMAP_TABLESPACE parameter as shown below:

```
C:\oracle\app\oracle\admin\XE\dpdump>impdp system/password
DIRECTORY=DATA_PUMP_DIR DUMPFILE=BIOMARTAPP0313.DMP LOGFILE=impdp.log
REMAP_TABLESPACE=BIOMART_WZ:BIOMART
```


Amazon Web Services Operations

Miscellaneous operations performed on the Amazon Cloud.

File Transfer Using WinSCP

Note: Our data upload server(75.101.128.23) is the backup server.
Large data files should be uploaded to the "/data" folder.

File transfer using WinSCP:

1. Download WinSCP. WinSCP can be downloaded from:
<http://winscp.net/eng/download.php>
2. Install SCP. Double click the downloaded package and follow the instructions.
3. Connect SCP to AWS instance.

Once you launch WinSCP, click the **New** button and the next screen will appear. You should specify the AWS instance's IP address or public domain name in the **Host Name** field, your login name in **User Name** field, and your private key file in the **Private key file** field.

After you fill the needed authentication info as before, you can log in.

Creating a Self-Signed SSL Certificate for Apache

1. Generate a Private Key.

The **openssl** toolkit is used to generate an **RSA Private Key** and **CSR (Certificate Signing Request)**. It can also be used to generate self-signed certificates which can be used for testing purposes or internal usage.

The first step is to create your RSA Private Key. This key is a 1024-bit RSA key which is encrypted using Triple-DES and stored in a PEM format so that it is readable as ASCII text.

```
openssl genrsa -des3 -out server.key 1024
Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
```

2. Generate a CSR (Certificate Signing Request)

Once the private key is generated, a Certificate Signing Request can be generated. The CSR is then used in one of two ways. Ideally, the CSR will be sent to a Certificate Authority, such as Thawte or Verisign who will verify the identity of the requestor and issue a signed certificate. **The second option is to self-sign the CSR, which will be demonstrated in the next section.**

During the generation of the CSR, you will be prompted for several pieces of information. These are the X.509 attributes of the certificate. One of the prompts will be for "Common Name (e.g., YOUR name)". It is important that this field be filled in with the fully qualified domain name of the server to be protected by SSL. If the website to be protected is <https://public.akadia.com>, enter public.akadia.com at this prompt. The command to generate the CSR is as follows:

```
openssl req -new -key server.key -out server.csr
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:Massachusetts
Locality Name (eg, city) [Newbury]:Waltham
Organization Name (eg, company) [My Company Ltd]:Recombinant Data Corp
Organizational Unit Name (eg, section) []:Information Technology
Common Name (eg, your name or your server's hostname)
[]:tmsmart.jnj.recomdata.com
Email Address []:hxia@recomdata.com
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

3. Remove Passphrase from Key.

One unfortunate side-effect of the passphrased private key is **that Apache will ask for the pass-phrase each time the web server is started**. Obviously this is not necessarily convenient, as someone will not always be around to type in the passphrase, such as after a reboot or crash. mod_ssl includes the ability to use an external program in place of the built-in passphrase dialog; however, this is not necessarily the most secure option. **It is possible to remove the Triple-DES encryption from the key**, thereby no longer needing to type in a passphrase.

If the private key is no longer encrypted, it is critical that this file only be readable by the root user! If your system is ever compromised and a third party obtains your unencrypted private key, the corresponding certificate will need to be revoked. With that being said, use the following command to remove the pass-phrase from the key:

```
cp server.key server.key.org
openssl rsa -in server.key.org -out server.key
```

4. Generating a Self-Signed Certificate.

At this point you will need to generate a self-signed certificate because you either don't plan on having your certificate signed by a CA, or you wish to test your new SSL implementation while the CA is signing your certificate. This temporary certificate will generate an error in the client browser to the effect that the signing certificate authority is unknown and not trusted.

To generate a temporary certificate which is good for 365 days, issue the following command:

```
openssl x509 -req -days 365 -in server.csr -signkey server.key -out
server.crt
Signature ok
... ..
Getting Private key
```

5. Installing the Private Key and Certificate.

When Apache with `mod_ssl` is installed, several directories in the Apache config directory are created. The location of this directory will differ depending on how Apache was compiled.

```
cp server.crt /etc/httpd/conf.d/server.crt
cp server.key /etc/httpd/conf.d/server.key
```

6. Configuring SSL Enabled Virtual Hosts.

```
SSLEngine on
SSLCertificateFile /usr/local/apache/conf/ssl.crt/server.crt
SSLCertificateKeyFile /usr/local/apache/conf/ssl.key/server.key
```

7. Restart Apache and Test.

EITHER

```
/etc/init.d/httpd stop
/etc/init.d/httpd stop
```

OR

```
service httpd start
service httpd stop
```

Shutting Down and Starting Up JBoss

- Log into the Cloud server as appuser using PuTTY with appuser's private key.
- Shutdown: `"/usr/local/jboss-4.2.2.GA/bin/shutdown.sh -s 0.0.0.0"`.

Be sure to check whether the JBoss java process disappears. Sometime its process hangs up there after shutdown. In this situation, manually kill the JBoss java process using `"kill -9 <pid>"`.

- Startup: `"/usr/local/jboss-4.2.2.GA/bin/run.sh -b 0.0.0.0 &"`

Shutting Down and Starting Up the Oracle Database Server

1. Log into the Cloud server using Putty with your private key.
2. Gain root privilege: `"sudo su - root"`.
3. su oracle for root: `"sudo su - oracle"`.
4. Log into the Oracle server: `"sqlplus /nolog"`.
5. Under SQL> prompt, type `"connect / as sysdba"`.
6. Enter `"shutdown transactional;"` to shut down the database gracefully.
7. Once it shuts down, type `"restart;"` to start the database server.

The following is a sample screen display:

```
$ sqlplus /nolog
SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 14 15:20:12
2009
Copyright (c) 1982, 2007, Oracle. All rights reserved.
SQL> connect / as sysdba
Connected.
SQL> shutdown transactional;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup
ORACLE instance started.
Total System Global Area 1.0689E+10 bytes
Fixed Size 2147392 bytes
Variable Size 6912216000 bytes
Database Buffers 3758096384 bytes
Redo Buffers 17014784 bytes
Database mounted.
Database opened.
SQL> exit
```



```

Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application
Testing options

```

Resetting Passwords

Resetting Your VNC Password

Steps to reset your VNC password:

1. Log into the Cloud server using PuTTY with your private key.
2. Run the command: **cd .vnc.**
3. Run the command: **rm passwd.**
4. Type: **vncpasswd**, then enter your new vnc password.

Resetting Your Linux Password

This password is used to unlock your locked VNC session only. You cannot use this password to log into any AWS instances.

Steps to reset your Linux account password:

1. Log into the Cloud server using PuTTY with your private key.
2. Type: **passwd**, then follow the instruction and reset your Linux account password.

Setting Up the Production Database Server

```

-- PSE6: Pathway Studio
CREATE TABLESPACE PSE6 DATAFILE
  '/u02/oradata/dw2/pse6.DBF'
SIZE 20G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

```

Setting Up the Production Database Server

```
-- PSE6 User
create user pse6_resnet
  identified by pathway61
  default tablespace pse6
  temporary tablespace temp
  quota unlimited on pse6;
grant resource to pse6_resnet;
grant connect to pse6_resnet;
grant create table to pse6_resnet;
grant create sequence to pse6_resnet;
grant create view to pse6_resnet;
grant execute on CTXSYS.CTX_DDL to pse6_resnet;
grant create procedure to pse6_resnet;
grant create function to pse6_resnet;
grant create trigger to pse6_resnet;
grant create materialized view to pse6_resnet;
grant dba to pse6_resnet;

-- BIOMART
CREATE TABLESPACE BIOMART DATAFILE
  '/u02/oradata/dw2/BIOMART1.DBF'
SIZE 10240M AUTOEXTEND ON NEXT 1024M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

-- BIOMART_WZ
CREATE TABLESPACE BIOMART_WZ DATAFILE
  '/u02/oradata/dw2/BIOMART_WZ.DBF'
SIZE 10240M AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
NOLOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;
-- BIOMART_INDX
CREATE TABLESPACE BIOMART_INDX DATAFILE
  '/u01/oradata/dw2/BIOMART_INDX.DBF'
SIZE 10240M AUTOEXTEND ON NEXT 1024M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;
```

Installing RPMs

1. Download the following RPMs from the URL: <http://cran.r-project.org/>:

- R-2.8.1-1.rh5.x86_64.rpm
- R-devel-2.8.1-1.rh5.x86_64.rpm
- libRmath-2.8.1-1.rh5.x86_64.rpm
- libRmath-devel-2.8.1-1.rh5.x86_64.rpm

2. Log into the Cloud server, su root using "sudo su - root" and install RPMs:

- rpm -ivh libRmath-2.8.1-1.rh5.x86_64.rpm
- rpm -ivh libRmath-devel-2.8.1-1.rh5.x86_64.rpm
- rpm -ivh R-2.8.1-1.rh5.x86_64.rpm

The following message appears:

```
warning: R-2.8.1-1.rh5.x86_64.rpm: Header V3 DSA signature: NOKEY, key ID
99b62126
error: Failed dependencies:
    libgfortran.so.1()(64bit) is needed by R-2.8.1-1.rh5.x86_64
    libtcl8.4.so()(64bit) is needed by R-2.8.1-1.rh5.x86_64
    libtk8.4.so()(64bit) is needed by R-2.8.1-1.rh5.x86_64
    xdg-utils is needed by R-2.8.1-1.rh5.x86_64
```

The message means that you need install these dependencies first in order to install R-2.8.1-1.rh5.x86_64.rpm.

- rpm -ivh R-devel-2.8.1-1.rh5.x86_64.rpm

The following RPMs must be installed first:

```
warning: R-devel-2.8.1-1.rh5.x86_64.rpm: Header V3 DSA signature: NOKEY, key
ID 99b62126
error: Failed dependencies:
    R = 2:2.8.1-1.rh5 is needed by R-devel-2.8.1-1.rh5.x86_64
    bzip2-devel is needed by R-devel-2.8.1-1.rh5.x86_64
    cairo-devel is needed by R-devel-2.8.1-1.rh5.x86_64
    gcc-gfortran is needed by R-devel-2.8.1-1.rh5.x86_64
    libICE-devel is needed by R-devel-2.8.1-1.rh5.x86_64
    libSM-devel is needed by R-devel-2.8.1-1.rh5.x86_64
    libX11-devel is needed by R-devel-2.8.1-1.rh5.x86_64
    libXmu-devel is needed by R-devel-2.8.1-1.rh5.x86_64
    libXt-devel is needed by R-devel-2.8.1-1.rh5.x86_64
    libjpeg-devel is needed by R-devel-2.8.1-1.rh5.x86_64
    libpng-devel is needed by R-devel-2.8.1-1.rh5.x86_64
    ncurses-devel is needed by R-devel-2.8.1-1.rh5.x86_64
    readline-devel is needed by R-devel-2.8.1-1.rh5.x86_64
    tcl-devel is needed by R-devel-2.8.1-1.rh5.x86_64
    tetex-latex is needed by R-devel-2.8.1-1.rh5.x86_64
    texinfo is needed by R-devel-2.8.1-1.rh5.x86_64
    texinfo-tex is needed by R-devel-2.8.1-1.rh5.x86_64
    tk-devel is needed by R-devel-2.8.1-1.rh5.x86_64
```

There are two ways to install the dependencies:

- Using yum if the server connects to a repository
- Find and download needed RPMs and install them one by one

Configuring SQL Developer

1. Start **sqldeveloper** in the Cloud server.
2. Create a new connection.
3. Configure the connection:
 - ☐ Connection Name: <assign any name that is meaningful to you>
 - ☐ Username: <database user you want to log in>
 - ☐ Password: <database user's password>
 - ☐ Hostname: localhost or DB server's public IP address if remote
 - ☐ Port: 1521
 - ☐ SID: DW1 for the staging db server; DW2 for the production db server
 - ☐ Service name: Leave as is
4. Test the configuration. If successful, check the **Save Password** box to save it.

Providing VNC Access

Step 1: Linux Group/User Creation

1. Log in as root.
2. Create a new group with a command in the following format:

```
# groupadd -g <group_id> <group_name>
```

For example:

```
# groupadd -g 505 oracle
```

Note: Note: <group_id> should not exist in /etc/group.*

3. Create a new user with a command in the following format:

```
useradd -g <group_id> -u <user_id> -s /bin/bash -m -k /etc/skel -d /home/<login_name> <login_name>
```

For example:

```
# useradd -g 505 -u 505 -s /bin/bash -m -k /etc/skel -d /home/oracle oracle
```

Note: <group_id> is the one created in the previous step, <user_id> should not exist in /etc/passwd.

4. Set up the initial password:

```
passwd oracle
```

5. Type the password twice.

Users should change their passwords using the same command.

Step 2: Set Up the Account to Allow VNC Viewer Access

1. Log in as user.
2. Enter the command **vncserver**, then type in the vncserver password.

This password will be used to permit you to connect to **vncserver**.

The vncserver's password is stored in the **<home_directory>/.vnc/passwd** file. If you need to change your vncserver password, delete this password file and run vncserver command again.

Additional Information

For details on tunneling VNC over SSH, see:

<https://www.intriniumsecurity.com/resources/white-papers/Tunneling%20VNC%20over%20SSH.pdf/view>

For details on implementing key-based SSH authentication, see:

<https://www.intriniumsecurity.com/resources/white-papers/Implementing%20Key-based%20SSH%20Authentication.pdf/view>

A Note on Using TOAD on the Cloud

TOAD converts your login user name and password to upper case before logging into Oracle. If the new Oracle 11 parameter `sec_case_sensitive_logon` is enabled and set to `MIXED`, you may receive an `Invalid Username/Invalid Password` error when attempting to log into the Cloud with TOAD.

If you receive this error, try disabling the parameter with the following command:

```
alter system set sec_case_sensitive_logon=false scope=both;
```

Chapter 4

Database Access Security

Access Credentials

Oracle Credentials

The following table lists the Oracle user IDs and passwords for zones and key databases in the data warehouse:

Zone/Database	User ID	Password	Description
biomart	biomart	biomart	Main application data.
biomart_lz	biomart_lz	biomart_lz	Landing zone for new data.
biomart_user	biomart_user	biomart_user	Entry point for tranSMART into the rest of the databases.
biomart_wz	biomart_wz	biomart_wz	Working zone where data is manipulated and tested prior to loading to the biomart.
control	control	control	Storage for all ETL information and code.
de_app	de_app	de_app	Entry point for Dataset Explorer into the rest of the databases.
reference	reference	reference	Stores reference data such as Genes, pathways, etc...
searchapp	searchapp	searchapp	All user information.

Credentials for Other Data Sources

Oracle Schema	Password	Description
jbl_jwb	jbl_jwb	Jubilant Oncology Data released on 03-03-2009.
jbl_exdb	jbl_exdb	Reference Omic data used by Jubilant Oncology Data released on 03-03-2009.
PICTOR	PICTOR	Pathways from Pictor.
MICROARRAY	MICROARRAY	Beerse data.
omicsoft_lz	omicsoft	Landing zone for Omicsoft data.

Access Control

JNJ_Application Security Group

The security group JNJ_Application is dedicated to controlling access to the production application server (174.129.234.170).

Incoming traffic for all ports is blocked unless opened in the following circumstances:

- The following list of ports or services are allowed to be tunneled through the port 22 (ssh):
 - 443 (https, Apache)
 - 5901 - 5920 (VNC server)
 - 7070 (Tomcat server)
 - 9090 (JBoss server)
 - 80 (http, Apache)
 - 8080 (Tomcat)
 - 8088 (Tomcat)
 - 8073 (Tomcat, Pathway Studio Enterprise)
 - 8081 (Tomcat)
- Port 9090 is opened for Recombinant subnet 72.85.238.162, 72.85.238.162, and 96.237.9.104.

JNJ_Common Security Group

JNJ_Common is used to control common permissions for the development application server (174.129.241.161), the development Oracle database server (174.129.237.81), the production application server (174.129.234.170), the production Oracle database server (174.129.240.115), and the backup server (75.101.128.23). The following rules are implemented in this security group:

- All ports are closed for incoming traffic unless opened under the rules below.
- Port 22 (ssh) is opened for the following IP addresses:
 - J&J's AWS instances, including the development application server (174.129.241.161), the development Oracle database server (174.129.237.81), the production application server (174.129.234.170), the production Oracle database server (174.129.240.115), the backup server (75.101.128.23), and the Inforsense development server (174.129.105.99).
 - Recombinant Data Corp: 96.237.9.104, 72.85.238.163 and 72.85.238.162, 68.162.251.169.
 - Johnson & Johnson's public IPs:

Raritan: 148.177.1.210-213, 215-216, and 218-219

=====

148.177.1.210

148.177.1.211

148.177.1.212

148.177.1.213

148.177.1.215

148.177.1.216

148.177.1.218

148.177.1.219

Cincinnati: 148.177.69.210-213

=====

148.177.69.210

148.177.69.211

148.177.69.212

148.177.69.213

Toronto: 199.243.112.37-38

=====

199.243.112.37

199.243.112.38

Raritan FW PAT: 148.177.0.100-101

=====

148.177.0.100

148.177.0.101

Access Control

Cincinnati FW PAT: 148.177.68.50

=====

148.177.68.50

Canada, Toronto FW PAT: 199.243.112.10

=====

199.243.112.10

and

Belgium (Beerse): 148.177.129.210-219

=====

148.177.129.210

148.177.129.211

148.177.129.212

148.177.129.213

148.177.129.214

148.177.129.215

148.177.129.216

148.177.129.217

148.177.129.218

148.177.129.219

Belgium (Beerse): 148.177.128.81-84

=====

148.177.128.81

148.177.128.82

148.177.128.83

148.177.128.84

Belgium (Mechelen): 212.190.70.38.7

=====

212.190.70.38

212.190.70.7

- Port 443 (https) is opened for the following IP addresses:

Johnson & Johnson's public IPs:

Raritan: 148.177.1.210-213, 215-216, and 218-219

=====

148.177.1.210

148.177.1.211

148.177.1.212

148.177.1.213

148.177.1.215

148.177.1.216

148.177.1.218

148.177.1.219

Cincinnati: 148.177.69.210-213

=====

148.177.69.210

148.177.69.211

148.177.69.212

148.177.69.213

Toronto: 199.243.112.37-38

=====

199.243.112.37

199.243.112.38

Raritan FW PAT: 148.177.0.100-101

=====

148.177.0.100

148.177.0.101

Cincinnati FW PAT: 148.177.68.50

=====

148.177.68.50

Canada, Toronto FW PAT: 199.243.112.10

=====

199.243.112.10

and

Belgium (Beerse): 148.177.129.210-219

=====

148.177.129.210

148.177.129.211

148.177.129.212

148.177.129.213

148.177.129.214

148.177.129.215

148.177.129.216

148.177.129.217

148.177.129.218

148.177.129.219

Belgium (Beerse): 148.177.128.81-84

=====

148.177.128.81

148.177.128.82

148.177.128.83

148.177.128.84

```
Belgium (Mechelen): 212.190.70.38.7  
=====
```

```
212.190.70.38  
212.190.70.7
```

Recombinant Data Corp: 96.237.9.104, 72.85.238.163 and 72.85.238.162, 68.162.251.169.

- Port 80 (http) is opened for the following IP addresses:
 - Recombinant Data Corp: 96.237.9.104, 72.85.238.163 and 72.85.238.162, 68.162.251.169

JNJ_Integration Security Group

The security group JNJ_Integration is dedicated to controlling access to the development application server (174.129.241.161). The following rules are implemented:

- All port's incoming traffic are blocked unless they are opened under the following rules.
- The following list of ports or services are allowed to be tunneled through the port 22 (ssh):
 - 443 (https, Apache)
 - 5901 - 5920 (VNC server)
 - 7070 (Tomcat server)
 - 9090 (JBoss server)
 - 8081 (Tomcat)
- Port 9090 are opened for Recombinant subnet, 72.85.238.162, 72.85.238.162, and 96.237.9.104

JNJ_Staging_DB Security Group

The security group JNJ_Staging_DB is dedicated to controlling access to the development database server (174.129.237.81). The following rules are implemented:

- All port's incoming traffic are blocked unless they are opened under the following rules.
- Port 1521 (Oracle TNS Listener) is allowed access from the development application server (174.129.241.161) and the production database server (174.129.240.115).

- The following list of ports or services are allowed to be tunneled through the port 22 (ssh):
 - 5901 - 5920 (VNC server)
 - 1521 (Oracle TNS Listener)

JNJ_Warehouse_DB Security Group

The security group JNJ_Warehouse_DB is dedicated to controlling access to the production database server (174.129.240.115). The following rules are implemented:

- All port's incoming traffic are blocked unless they are opened under the following rules.
- Port 1521 (Oracle TNS Listener) is allowed access from the production application server (174.129.234.170).
- The following list of ports or services are allowed to be tunneled through the port 22 (ssh):
 - 5901 - 5920 (VNC server)
 - 1521 (Oracle TNS Listener)

tranSMART IP Address White List

This is the list of IP addresses from which Johnson & Johnson and external users are allowed to connect to tranSMART:

J&J IP Addresses:

- Raritan 148.177.1.210-213, 215-216, and 218-219
- Cincinnati 148.177.69.210-213
- Toronto 199.243.112.37-38
- Raritan FW PAT 148.177.0.100-101
- Cincinnati FW PAT 148.177.68.50
- Canada, Toronto FW PAT 199.243.112.10
- Belgium (Beerse) 148.177.129.210-219
- Belgium (Beerse) 148.177.128.81-84
- Belgium (Mechelen) 212.190.70.38.7

J&J External

- 173.62.218.247 (Greg)

St. Jude's External

- 192.55.208.10

Recombinant Data Corp.

- 71.174.229.66 (Jinlei)
- 71.174.86.101 (Kevin)
- 173.76.205.58 (Chris)
- 71.255.161.41 (Jeremy)
- 24.218.125.72 (Jeff)
- 76.195.183.40 (Joseph Adler)
- 71.184.149.228 (Jean)

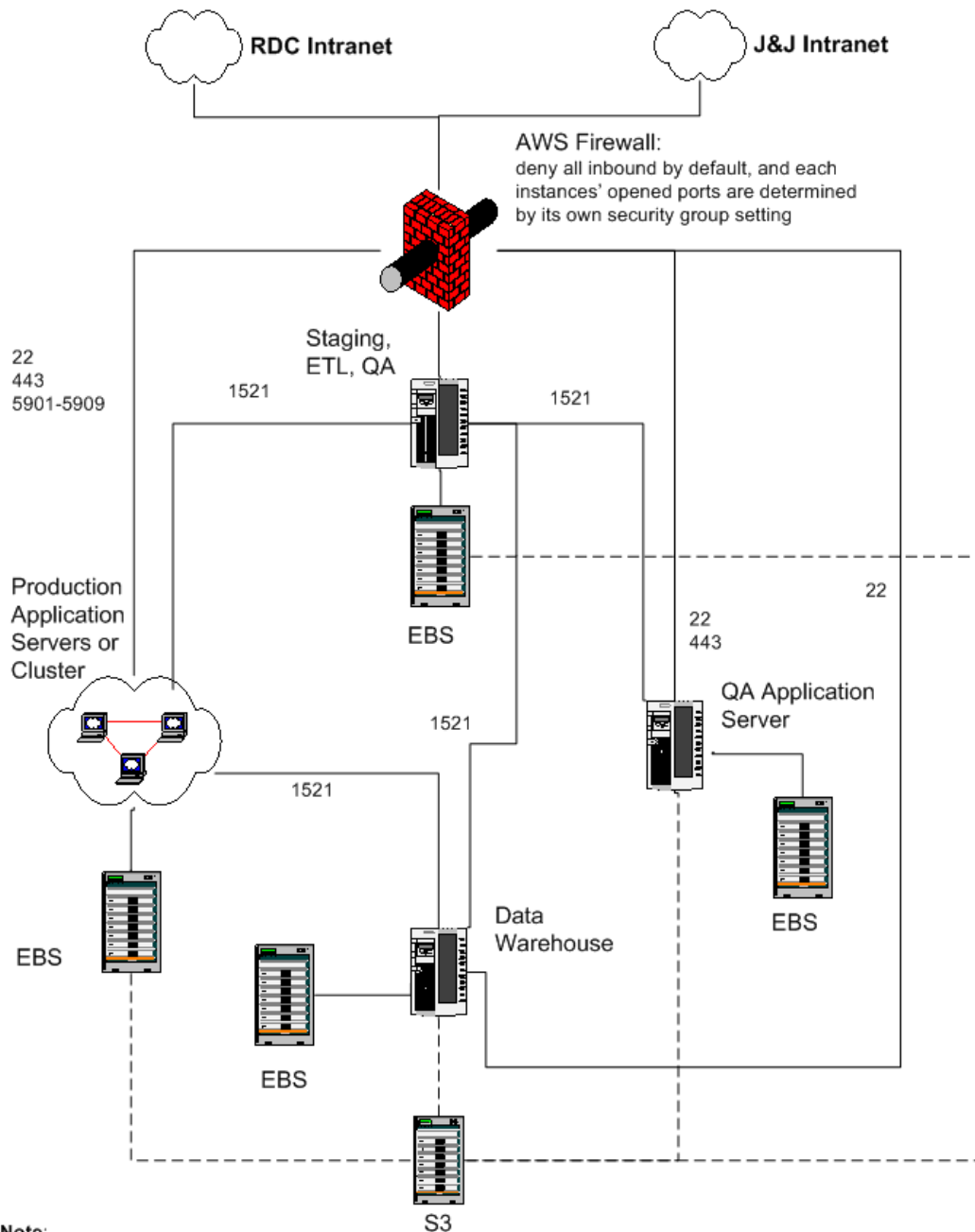
Inforsense

- 87.224.22.169 (Inforsense Cambridge)
- 64.61.143.137 through 64.61.143.142 (Inforsense London)
- 74.233.6.213(Faruk Kay)

High-Level View of System and Security

Only J&J's Intranet and Recombinant Data Corp's Intranet are allowed to access the J&J BIO-Mart Data Warehouse environment through SSH or HTTPS. All non-encrypted data transfer protocol ports are disabled.

AWS Environment for J & J Biomart Data Warehouse



Note:

1. Numbers are TCP ports needed to be opened, either for Internet or internal use. Port 22 for SSH, 443 for HTTPS, 1521 for Oracle, 590x for VNC Server.
2. S3 is used for instances and database backup, a tape replacement
3. EBS is a off-instance storage with high availability and reliability

Version: 20090213
Recombinant Data Corp.

Chapter 5

Release Management

Deployment to the Production Server

URLs for application and database deployment on the production server are located as follows:

- [Application deployment](#): page 16
- [Database deployment](#): page 19

Application Deployment to Production Server

Example of migrating tranSMART from the development server to the production server:

1. Migrate the following schemas as needed:
 - ☐ biomart
 - ☐ biomart_user
 - ☐ searchapp
 - ☐ deapp
 - ☐ i2b2metadata
 - ☐ i2b2demodata
 - ☐ i2b2hive
2. Copy documents from the devAPP's /usr/local/tomcat-5.5.27/appdata to the prodAPP.
3. Deploy the packaged tmsmart.war to /usr/local/tomcat-5.5.27/webapps.
4. Restart Tomcat server.

The shell script `deploy.sh` under `/home/appuser/script` will automatically remove the exiting version of `tmsmart.war` from the prodApp's Tomcat, pick up `tmsmart.war` from Hudson's destination at the devApp server, and move to the prodApp's `/usr/local/tomcat-5.5.27/webapp` directory using SCP for you, and then stop and start Tomcat as needed.

Use your private key to log in the prodApp server, and enter the command `"sudo su - appuser"` or directly log in as the appuser. Then run the command `"cd /home/appuser/script"`, and type `"sh deploy.sh"`.

Database Deployment to Production Server

Example of deploying databases from the development server to the production server:

1. Log into the prodDB server using your private key.
2. Enter the command "sudo su - oracle" and become oracle user.
3. cd /dw1dump/scripts.
4. Modify expdp.sh's dbs variable as needed, and then enter "sh expdp.sh".

This shell script will use Oracle tool expdp to export databases or schemas listed in the variable dbs. The dump file will be stored in the devDB's /backup/database directory, the dump file with a name <schema_name><MM><YYYY><DD>.log.

5. Modify mvdp.sh's dbs variable as needed, and then enter "sh mvdp.sh".

This shell scripts will use SCP to move dump files and log files from the devDB's /backup/database into the prodDB's /dw1dump/dump directory.

6. Modify impdp.sh's dbs variable as needed, and then enter "sh impdp.sh"

This shell scripts will use Oracle's impdp to import dump files from the prodDB's /dw1dump/dump into the production database DW2, and generated log files will be placed in /dw1dump/log directory.

Note: Before importing, it's better to drop the users which will be imported using "drop user <username> cascade;" as system.

Tag a New Release in SVN

Both the tranSMART search tool and the i2b2WebClient (Dataset Explorer) need to be tagged for a new release.

Steps to create a new release tag using "TortoiseSVN":

1. Do a svn checkout <https://svn.recomdata.com/repo1/jnj/tags> to your local svn repository.
2. Create a new folder in the tags folder - **The folder name is the tag name!**
3. Do svn add & commit on the newly created folder (the tag folder).
4. Get the latest copy of the search tool and the i2b2WebClient from the svn trunk:
URLs: <https://svn.recomdata.com/repo1/jnj/trunk/searchTool>
<https://svn.recomdata.com/repo1/jnj/trunk/i2b2WebClient>
5. Right click on the "searchTool" folder on your local svn repository -> "TortoiseSVN" -> "Branch/Tag" in the copy/branch window.

Create a Release Note

JIRA can be used to generate a preliminary release note:

1. Log into to JIRA, Browse to the "JnJ BioMarkers" project and Click "Release Note".



2. Select the current release version and "Text" as the style.
3. Use "release note template" to create a new release note page.

[Click to create a new release note using the template](#)

Deployment to the Public Training Server

The public training server supports internal JnJ trainees and non-JnJ trainees.

The tranSMART resources that a trainee can access depends on the login credentials, as follows:

Trainee Type	Login Credentials	Resources
JnJ employees	jnjuser1 through jnjuser30 Password training	Resources that JnJ users can access and non-JnJ users cannot include: <ul style="list-style-type: none"> ■ JnJ clinical trials
Non-JnJ employees	publicuser1 through publicuser30 Password training	<ul style="list-style-type: none"> ■ Pictor ■ ResNet ■ GeneGo ■ JnJ documents <p>In Dataset Explorer, the only studies available to both groups are those in the Public Studies node.</p>

Deployment URLs

URLs for application and database deployment on the public training server are located as follows:

- [Application deployment](#): page 21
- [Database deployment](#): page 21

Application Deployment to the Training Server

1. Log in to the JNJ [devApp](#) server (URL on page 4) as `appuser`.
2. Make a tranSMART build for JNJ public training application server.
3. Change to the following directory:

```
/home/appuser/scripts
```

4. Run the following command:

```
sh deploy4training.sh
```

The tranSMART `war` file will be copied to the correct location on the JNJ public training server, shown below, and replace the out-of-date version of the `war` file:

```
/usr/local/tomcat-5.5.27/webapps
```

5. Log in to the JNJ application training server as `appuser`.
6. Shut down the Tomcat server.
7. Delete the directory `webapps/trasmart`.
8. Start the Tomcat server.

Database Deployment to the Training Server

1. Log in to the JNJ [devDB](#) server (URL on page 7) as `oracle`.
2. Using `expdp` to export `biomart`, `biomart_user`, `deapp`, `i2b2metadata`, `i2b2demodata`, `searchapp` and `control`.
3. Copy these files to the database training server using the command:

```
scp <file name> 10.192.178.207:/u04/dpdump/.
```

4. Log in to the database training server and use `impdp` to import databases as needed:

```
imp <user/passwd> directory=bk dumpfile=<dmp file>  
logfile=<log file> schemas=<schema>
```

Chapter 6

Development Tools

Source Control (SVN)

tranSMART source files are stored in Recombinant's Subversion (SVN) source control system. The files are in the following location:

<https://svn.recomdata.com/repo1/jnj>

Eclipse

Download Eclipse from the following location:

<http://www.eclipse.org/downloads/>

Eclipse Plugins

You might find the following plugins helpful:

Eclipse checkstyle plugin - Code Inspection

1. Within Eclipse go to Help->Software Updates->Find and Install.
2. Choose Search for new features to install and press Next.
3. Create a New Remote Site...
4. Input a name to your liking (for instance Checkstyle Plug-in) and input the following URL: <http://eclipse-cs.sourceforge.net/update>.
5. Click your way through the following pages to install the plug-in.

Enerjy - Another Code Inspection tool

The Enerjy Eclipse plug-in can be downloaded and installed via the Automatic Software Update feature within Eclipse. In Eclipse, go to Help, Software Updates. Select Find and Install. Click the New Remote Site button, and add Enerjy Software to the name and the following URL:

<http://update.enerjy.com/eclipse>

Follow the on-screen instructions.

Grails

1. Download 1.1-RC1 from grails: <http://www.grails.org/Download> or copy from <\\jjhost2\data\Utilities\grails-bin-1.1-RC1.zip>.
2. Unzip the download to a local folder.
3. Update your environment variable GRAILS_HOME to point to the new grails 1.1-RC1 folder.
4. Open a command window and run "grails." Make sure grails is running with the new version.
5. Reinstall Acegi security plugin:

```
grails install-plugin acegi
```

Note: You might need to upgrade the hibernate plugin. Run the following script from your biomart app folder:

```
grails upgrade
```

Change the default grails http port. The default port number can be found in your \${GRAILS_HOME}\scripts _GrailsSettings.groovy.

6. Make sure that the classpath variable for GRAILS_HOME is changed in Eclipse (Window->Preferences, Java->Build Path->Classpath Variables).

Oracle SQL Developer

Required files:

- SQL Developer - <\\jjhost2\share\Utilities\sqldeveloper.zip>
- JTDS JDBC Driver - <\\jjhost2\share\Utilities\jtds-1.2.2-dist.zip>
- SQL Server JDBC Driver - \\jjhost2\share\Utilities\sqljdbc_1.2.2828.100_enu.exe

Installation procedure:

1. Unzip sqldeveloper.zip to C:\Program Files to install SQL Developer.
2. Create a shortcut to C:\Program Files\sqldeveloper\sqldeveloper.exe.
3. Unzip jtds-1.2.2-dist.zip to C:\Program Files to install the JTDS JDBC drivers.
4. Execute sqljdbc_1.2.2828.100_enu.exe to install the SQL Server JDBC drivers.
5. Start sqldeveloper and when prompted, point to you installation of the JDK.
6. Follow the instructions from the [SQL Developer User's Guide](#) to install the JTDS driver.

7. Use these same instructions to install the SQL Server JDBC drivers, but point to the following Jar file: C:\Program Files\Microsoft SQL Server 2005 JDBC Driver\sqljdbc_1.2\enu\sqljdbc.jar.

Indexer Tool

The Indexer tool is a Java console application which uses [Lucene](#) to index disk-based document repositories.

Building Indexer

The indexer project is an Eclipse project and is available at the following location:

<https://svn.recomdata.com/repo1/jnj/trunk/util/indexer>

Check out or update the project from Subversion, import it as an existing project into Eclipse, build the project, and then do an Ant build on the indexer.xml file (right-click on this file and select Run as... -> Ant Build). This will produce an indexer.jar with all the required libraries.

Usage

Copy the indexer.jar, indexer.bat, merger.bat, and finder.bat files to a host that has access to the directories you want to index. Make sure Java 1.6 is on the host and that its bin directory is on the PATH.

Building Repository Indexes

Edit the indexer.bat file, then run it to build indexes. You may include one or more commands in the batch file like the following:

```
java -Xms256m -Xmx1024m -cp indexer.jar
    com.recomdata.search.Indexer -create -index "indexes\Biomarker" -
repository "Biomarker" -path Z:\
```

This command creates a new index in the indexes\Biomarker folder of all the files found under the network share Z:\ drive, and sets each indexed document's repository field to "Biomarker".

Three files should be created in the indexes\Biomarker folder with names similar to the following: _wwc.cfs, segments.gen, and segments_tm0. Create a zip file of the Biomarker files named Biomarker-yyyyymmdd.zip and copy it to devapp.jnj.recomdata.com server:/data/indexes.

If you are indexing new Conference files, copy the new files to subdirectories with meaningful names under /usr/local/tomcat-5.5-27/appdata/transmart/Documents/Conferences on the devapp server. Then build a new Conferences index by indexing that directory.

```
java -Xms256m -Xmx1024m -cp indexer.jar
    com.recomdata.search.Indexer -create -index
"/data/indexes/Conferences" -repository "Conferences"
    -path "/usr/local/tomcat-5.5-
27/appdata/transmart/Documents/Conferences"
```

Testing an Index

Edit the finder.bat file, then run it to test indexes. You may include one or more commands in the batch file like the following:

```
java -cp indexer.jar
    com.recomdata.search.Finder -index "C:\\indexes\\Biomarker" met
```

This command searches for the term "met" in the Biomarker index.

After you have created and tested new indexes, zip them up and copy them to the devapp.jnj.recomdata.com:/data/indexes.

Building the Master Index

tranSMART uses one Master index as its document index. It is built by merging the individual Repository indexes into one Master index. This process should be run on the devapp.jnj.recomdata.com server.

Edit the merger.bat file, then run it to create the Master index.

```
mkdir indexes\Master
cp indexes\Biomarkers\* indexes\Master\*
java -Xms256m -Xmx1024m -cp indexer.jar
    com.recomdata.search.Merger -index "indexes\Master"
"indexes\Conferences"
java -Xms256m -Xmx1024m -cp indexer.jar
    com.recomdata.search.Merger -index "indexes\Master" "indexes\DIP"
java -Xms256m -Xmx1024m -cp indexer.jar
    com.recomdata.search.Merger -index "indexes\Master"
"indexes\Jubilant Oncology"
```


Dana Farber GCOD Dataset

Information for loading data from the GeneChip Oncology Database.

Oracle Users

The following users are created to hold Dana Farber GCOD datasets:

```
- WEBARRAY
CREATE USER WEBARRAY
IDENTIFIED BY WEBARRAY
DEFAULT TABLESPACE MARRAYDAT01
TEMPORARY TABLESPACE TEMP
PROFILE DEFAULT
ACCOUNT UNLOCK;
GRANT DBA TO WEBARRAY;
ALTER USER WEBARRAY DEFAULT ROLE ALL;
GRANT UNLIMITED TABLESPACE TO WEBARRAY;
- WEBARRAY_READ
CREATE USER WEBARRAY_READ
IDENTIFIED BY WEBARRAY_READ
DEFAULT TABLESPACE MARRAYDAT01
TEMPORARY TABLESPACE TEMP
PROFILE DEFAULT
ACCOUNT UNLOCK;
GRANT DBA TO WEBARRAY;
ALTER USER WEBARRAY DEFAULT ROLE ALL;
GRANT UNLIMITED TABLESPACE TO WEBARRAY_READ;
```

Tablespaces

The following tablespaces are created for loading Dana Farber GCOD dataset:

```
- MARRAYDAT01
CREATE BIGFILE TABLESPACE MARRAYDAT01 DATAFILE
'/u02/oradata/DW1/MARRAYDAT01.DBF'
SIZE 32G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;
- WEB_ARRAY_PART1
CREATE BIGFILE TABLESPACE WEB_ARRAY_PART1 DATAFILE
'/u02/oradata/DW1/WEBARRAY01.DBF'
SIZE 4G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
```

```

EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;
- WEB_ARRAY_PART2
CREATE BIGFILE TABLESPACE WEB_ARRAY_PART2 DATAFILE
'/u01/oradata/DW1/WEBARRAY02.DBF'
SIZE 4G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;
- WEB_ARRAY_PART3
CREATE BIGFILE TABLESPACE WEB_ARRAY_PART3 DATAFILE
'/u01/oradata/DW1/WEBARRAY03.DBF'
SIZE 4G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;
- WEB_ARRAY_PART4
CREATE BIGFILE TABLESPACE WEB_ARRAY_PART4 DATAFILE
'/u01/oradata/DW1/WEBARRAY04.DBF'
SIZE 4G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;
- WEB_ARRAY_PART5
CREATE BIGFILE TABLESPACE WEB_ARRAY_PART5 DATAFILE
'/u01/oradata/DW1/WEBARRAY05.DBF'
SIZE 4G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;
- DBADATA_TSP
CREATE BIGFILE TABLESPACE DBADATA_TSP DATAFILE
'/u01/oradata/DW1/DBADATA_TSP.DBF'
SIZE 4G AUTOEXTEND ON NEXT 128M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO;

```

Oracle Directory

The directory needs to be created for loading the datasets:

```
- FILE_MOUNT
CREATE OR REPLACE DIRECTORY FILE_MOUNT AS '/u01/biomart/file_mount';
```

Dana Farber GCOD Datasets Loaded

Four dmp files are loaded:

- 11G ddata10.dmp
- 11G ddata11.dmp
- 11G ddata12.dmp
- 94M webarray.dmp

Total size of used tablespaces is about 112G.

Dataset Explorer Development Environment Setup

How to set up your workstation for Dataset Explorer development:

1. Install Java EE SDK (version 1.6 or above).
2. Install Eclipse.
3. Install SVN client.
 - Most Recombinant developers use <http://tortoisesvn.tigris.org/>.
 - An alternative is Subclipse, which is an Eclipse plugin. It's available at <http://subclipse.tigris.org/>. (Some folks have experienced problems with this plugin when working on multiple projects.)
4. Install Oracle Client. (Make sure you install JDBC drivers.)
5. Install Grails.
 - Follow these directions to install grails: <http://grails.org/Installation>.
 - Follow these directions for Grails - Eclipse integration: <http://grails.org/Eclipse+IDE+Integration>.
 - Make sure you set the environment variables JAVA_HOME and GRAILS_HOME in Windows.
 - Make sure you set the CLASSPATH variable GRAILS_HOME in Eclipse (under project properties).

6. Check out the latest dev build.
7. Establish an ssh tunnel to the database server mapping local port 1520 to port 1521 on the remote server.
8. Open a command line window and do the following:
 - a. Change to the directory containing your local copy of Dataset Explorer. For example – "`~/workspace/jnj trunk/datasetExplorer.`"
 - b. Type "`grails install-plugin acegi`" to install the security framework plugin.
 - c. Type "`grails run-app`" to compile the code and start grails.

Chapter 7

Coding Standards

Database Coding Standards

Database Names

Database names are prefixed with "bio" to indicate that they are incorporated into the BIO Data Warehouse project:

Database Name	Description
bio_Loading_Zone	Loading Zone
bio_Working_Zone	Working Zone
bio_Data_Warehouse	Data Warehouse Zone
bio_Quality_Staging	Quality Staging
bio_Production	Production Zone

Table, View, and Procedure Names

These names are prefixed with an abbreviated form of their "zone" in uppercase. The names also use "X" to indicate executable procedures, and "T" to indicate temp tables:

Database Name	Table Name	Description
bio_Loading_Zone	src<Table>	Destination for SSIS/Mirth file/feed loads, truncated before each load
bio_Loading_Zone	LZ<Table>	Common load tables where data is copied in preparation for processing by the Working Zone
bio_Loading_Zone	LZX<Procedure>	Load procedures which copy data from src tables to LZ tables
bio_Working_Zone	WZ<Table>	Tables and Views

Database Name	Table Name	Description
bio_Working_Zone	WZX<Procedure>	Procedures
bio_Quality_Staging, bio_Production	QZ<Table>	Tables and Views
bio_Quality_Staging, bio_Production	QZX<Procedure>	Tables and Views

Column Names

Use underscore characters to separate each word – for example, Event_Type_Code.

Index Names

Index names should only use the column name to which they refer. They should NOT include the work "index" and should NOT indicate that they are clustered. They may include a number when necessary.

SQL Code

- Should be formatted with each new keyword on a new line.
- Table names should be aliased.
- Column names should be fully qualified.
- Table names should be fully qualified with the database name, with the exception of tables referenced within the staging and production databases.

```
select
    p.F_Name,
    p.L_Name,
    m.Medication_Name,
    m.Dosage
from
    rdc_BIO..Bio_Patients p,
    rdc_BIO..Bio_Fact_Medications m
where
    m.PtID = p.PtID
    and m.MedicationType = 'Generic'
```

Script Headers

Scripts should include the following header information:

- Script name
- Purpose
- Change history: Date, author, and brief description of changes
- Copyright info and license info, if applicable

Note: The source control system will document the change history along with this standard header automatically.

Example header:

```
-- WZX_Load_BIO_Patient.sql
-- Loads transformed data into Bio_Patient table.
-- 04/01/2008 A. Mandel - Created script.
-- 04/20/2005 A. Mandel - Updated with new column names.
-- Copyright © 2008 Recombinant Data Corp.
```

Grails Coding Standards

Domain Classes

The Grails Object Relational Mapping (GORM) capability uses Hibernate 3 internally. The domain classes have objects that are mapped to the database. The domain classes can be linked via relationships and provide very powerful CRUD operations.

Relationships between tables should be explicitly stated to improve readability and to make the relation obvious. Each variable is explicitly mapped to a column in a table, with one variable reserved for the "id" field. The version is set to "false" to avoid creating versions of the table.

```
class GeneExprAnalysis {
    Long    id
    String  contentID
    Long    contentTypeID
    String  probeSet
    Double  ratio
    CellLineDiseaseMapping cellLineDiseaseMapping
    SearchSubject searchsubject
    GeneExprAnalysisContent geneExprAnalysisContent
}
```

```

static mapping = {
    table 'GeneExpressionAnalysis'
    version false
    id column:'id'
    columns {
        probeSet column:'ProbeSet'
        contentID column:'ContentID'
        contentTypeID column:'ContentTypeID'
        genBankAccession column:'GenBankAccession'
        ratio column:'Ratio'
        cellLineDiseaseMapping column:'cellLineName'
        geneExprAnalysisContent column:'geaContentID'
    }
}

```

Controllers and Views

The controllers are responsible for handling requests from a particular domain class. The first part of the controller name is mapped to a URI, and each action defined within your controller maps to URI within the controller name URI. A controller is a class whose name ends in the word "Controller".

```

class ShirtController {
    def index = { redirect(action:list,params:params) }

    def list = {
        [ shirtList: Shirt.list( params ) ]
    }

    def show = {
        [ shirt : Shirt.get( params.id ) ]
    }

    def delete = {
        def shirt = Shirt.get( params.id )
        if(shirt) {
            shirt.delete()
            flash.message = "Shirt ${params.id} deleted."
            redirect(action:list)
        }
        else {
            flash.message = "Shirt not found with id ${params.id}"
            redirect(action:list)
        }
    }

    // ...
}

```

In the above example, the list closure in the ShirtController will handle requests where the URI is /shirt/list, and so on. ServletContext, sessions, and requests should be worked on in the controllers.

Services

A service is a class that holds one or more methods that implement business logic. Logical parts of the business logic are contained in separate service classes. Services in Grails are seen as the place to put the majority of the logic in the application, leaving controllers responsible for handling request flow via redirects.

Service class names should start with the name of the service and end with "Service". A country service would thus be called "CountryService".

```
class CountryService {
    static transactional = false
}
```

Native SQL should be avoided in the groovy services. Criteria builders allow querying databases to retrieve results as well as create projections. The use of HQL is advised only in circumstances when criteria builders may produce some overhead.

```
def c = Account.createCriteria()
def results = c {
    like("holderFirstName", "Fred%")
    and {
        between("balance", 500, 1000)
        eq("branch", "London")
    }
    maxResults(10)
    order("holderLastName", "desc")
}
```

Data Structures

Data structures should be created in their individual groovy classes. The structured data returned from the responses should be bound to the closures using the generic `bindData()` method.

```
bindData(session.searchFilter.trialFilter, params)
```

Java Coding Standards

Naming Conventions

Packages

Names representing packages should be entirely in lower case.

```
mypackage, com.company.application.ui
```

This is Sun's package naming convention for the Java core packages. The initial package name representing the domain name must be in lower case.

Data Types

Names representing types must be nouns and written in mixed case starting with upper case.

`Line, AudioSystem`

This is the common practice in the Java development community. It is also Sun's type naming convention for the Java core packages.

Variables

Variable names must be in mixed case starting with lower case.

`line, audioSystem`

This is the common practice in the Java development community. It is also Sun's naming convention for variables for the Java core packages. This makes variables easy to distinguish from types, and effectively resolves potential naming collision as in the `Line line;` declaration.

Constants

Names representing constants (final variables) must be entirely in upper case, using an underscore to separate words.

`MAX_ITERATIONS, COLOR_RED`

This is the common practice in the Java development community. It is also Sun's naming convention for the Java core packages.

In general, the use of such constants should be minimized. In many cases implementing the value as a method is a better choice:

```
int getMaxIterations() // NOT: MAX_ITERATIONS = 25
{
    return 25;
}
```

This form is both easier to read, and it ensures a uniform interface towards class values.

Methods

Names representing methods must be verbs and written in mixed case, starting with lower case.

`getName(), computeTotalWidth()`

This is the common practice in the Java development community. It is also Sun's naming convention for the Java core packages. This is identical to variable names, but methods in Java are already distinguishable from variables by their specific form.

Abbreviations and Acronyms

Abbreviations and acronyms should not be upper case when used as names.

```
exportHtmlSource(); // NOT: exportHTMLSource();
openDvdPlayer();    // NOT: openDVDPlayer();
```

Using all upper case for the base name will result in conflicts with the naming conventions given above. A variable of this type would have to be named dVD, hTML etc. which obviously is not very readable. Another problem is illustrated in the examples above. When the name is connected to another, readability is seriously reduced. The word following the acronym does not stand out as it should.

Generic Variables

Generic variables should have the same name as their type.

```
void setTopic(Topic topic) // NOT: void setTopic(Topic value)
                           // NOT: void setTopic(Topic aTopic)
                           // NOT: void setTopic(Topic t)

void connect(Database database) // NOT: void connect(Database db)
                                // NOT: void connect(Database oracleDB)
```

Reduce complexity by reducing the number of terms and names used. This convention also makes it easy to deduce the type given a variable name only.

If for some reason this convention doesn't seem to fit, it is a strong indication that the type name is badly chosen.

Non-generic variables have a role. These variables can often be named by combining role and type:

```
Point  startingPoint, centerPoint;
Name   loginName;
```

Objects Names and Method Names

The name of the object is implicit, and should be avoided in a method name.

```
line.getLength(); // NOT: line.getLineLength();
```

The latter might seem natural in the class declaration, but proves superfluous in use, as shown in the example.

Specific Naming Conventions

get... and set... Prefixes

The prefixes `get` and `set` must be used where an attribute is accessed directly.

```
employee.getName();  
employee.setName(name);  
  
matrix.getElement(2, 4);  
matrix.setElement(2, 4, value);
```

This is the common practice in the Java development community. It is also Sun's naming convention for the Java core packages.

is... Prefix

The `is` prefix should be used for Boolean variables and methods.

```
isSet, isVisible, isFinished, isFound, isOpen
```

This is the naming convention for Boolean methods and variables used by Sun for the Java core packages. Using the `is` prefix solves a common problem of choosing bad Boolean names like `status` or `flag`. `isStatus` or `isFlag` simply doesn't fit, and the programmer is forced to choose more meaningful names.

Setter methods for Boolean variables must have the `set` prefix, as in:

```
void setFound(boolean isFound);
```

There are a few alternatives to the `is` prefix that fit better in some situations. These are the prefixes `has`, `can` and `should`:

```
boolean hasLicense();  
boolean canEvaluate();  
boolean shouldAbort = false;
```

The initialize... Prefix

The prefix `initialize` can be used where an object or a concept is established.

```
printer.initializeFontSet();
```

The American "initialize" is preferred over the English "initialise." The abbreviation "init" must be avoided.

Object Collections

Plural form should be used for names representing a collection of objects.

```
Collection<Point>  points;
int[]              values;
```

This convention enhances readability since the name gives the user an immediate clue of the type of the variable and the operations that can be performed on its elements.

Iterator Variables

Iterator variables should be called *i*, *j*, *k*, etc.

```
for (Iterator i = points.iterator(); i.hasNext(); ) {
    ...
}

for (int i = 0; i < nTables; i++) {
    ...
}
```

This notation for indicating iterators is an established convention in mathematics.

Variables named *j*, *k*, etc. should be used for nested loops only.

Complement Entities

Complement names must be used for complement entities.

get/set, add/remove, create/destroy, start/stop, insert/delete,
increment/decrement, old/new, begin/end, first/last, up/down, min/max,
next/previous, old/new, open/close, show/hide, suspend/resume, etc.

Reduce complexity by symmetry.

Avoid Abbreviations in Names

Abbreviations in names should be avoided.

```
computeAverage();           // NOT: compAvg();
ActionEvent event;          // NOT: ActionEvent e;
catch (Exception exception) { // NOT: catch (Exception e) {
```

There are two types of words to consider. First are the common words listed in a language dictionary. These must never be abbreviated. Never write:

```
cmd   instead of  command
comp  instead of  compute
cp    instead of  copy
e      instead of  exception
init  instead of  initialize
pt    instead of  point
etc.
```

Then there are domain specific phrases that are more naturally known through their acronym or abbreviations. These phrases should be kept abbreviated. Never write:

```
HypertextMarkupLanguage  instead of  html
CentralProcessingUnit     instead of  cpu
PriceEarningRatio        instead of  pe
etc.
```

Avoid Negated Boolean Variable Names

Negated Boolean variable names must be avoided.

```
boolean isError; // NOT: isNoError
boolean isFound; // NOT: isNotFound
```

Confusion may occur when the logical `not` operator is used and double negative arises. It is not immediately apparent what `!isNoError` means.

Related Constants

Associated constants (final variables) should be prefixed by a common type name.

```
final int  COLOR_RED    = 1;
final int  COLOR_GREEN  = 2;
final int  COLOR_BLUE   = 3;
```

This construction indicates that the constants belong together, and what concept the constants represents.

An alternative to this approach is to put the constants inside an interface, effectively prefixing their names with the name of the interface:

```
interface Color
{
    final int RED    = 1;
    final int GREEN  = 2;
    final int BLUE   = 3;
}
```

Exception Classes

Exception classes should be suffixed with `Exception`.

```
class AccessException extends Exception
{
    ...
}
```

Exception classes are really not part of the main design of the program, and naming them like this makes them stand out relative to the other classes. This standard is followed by Sun in the basic Java library.

Default Interface Implementations

Default interface implementations can be prefixed by `Default`.

```
class DefaultTableCellRenderer
    implements TableCellRenderer
{
    ...
}
```

It is not uncommon to create a simplistic class implementation of an interface providing default behavior to the interface methods. The convention of prefixing these classes by `Default` has been adopted by Sun for the Java library.

Singleton Classes

Singleton classes should return their sole instance through the method `getInstance`.

```
class UnitManager
{
    private final static UnitManager instance = new UnitManager();

    private UnitManager()
    {
        ...
    }

    public static UnitManager getInstance() // NOT: get() or
instance() or unitManager() etc.
    {
        return instance;
    }
}
```

This is the common practice in the Java community, though it is not consistently followed by Sun in the JDK. The above layout is the preferred pattern.

Factory Classes

Classes that create instances on behalf of others (factories) can do so through the method `new<ClassName>`.

```
class PointFactory
{
    public Point newPoint(...)
    {
        ...
    }
}
```

This construction indicates that the instance is created by `new` inside the factory method, and that the construct is a controlled replacement of `new Point()`.

Functions and Procedures

Functions (methods returning an object) should be named after what they return. Procedures (void methods) should be named after what they do.

This convention increases readability. It also makes clear what the unit should do, and especially all the things it is not supposed to do. This again makes it easier to keep the code clean of side effects.

Files

Classes

Classes should be declared in individual files, with the file name matching the class name. Secondary private classes can be declared as inner classes and reside in the file of the class they belong to.

File Content

File content must be kept within 80 columns.

80 columns is the common dimension for editors, terminal emulators, printers, and debuggers, and files that are shared between several developers should keep within these constraints. It improves readability when unintentional line breaks are avoided when passing a file between programmers.

Avoid Special Characters

Special characters such as TAB and page break must be avoided.

These characters are bound to cause problem for editors, printers, terminal emulators or debuggers when used in a multi-programmer, multi-platform environment.

Break Long Lines Consistently

The incompleteness of split lines must be made obvious.

```
totalSum = a + b + c +  
          d + e;  
  
method(param1, param2,  
        param3);  
  
setText ("Long line split" +  
        "into two parts.");  
  
for (int tableNo = 0; tableNo < nTables;  
    tableNo += tableStep) {  
    ...  
}
```


Split lines occur when a statement exceeds the 80 column limit given above. It is difficult to give rigid rules for how lines should be split, but the examples above should give a hint.

In general:

- Break after a comma.
- Break after an operator.
- Align the new line with the beginning of the expression on the previous line.

Statements

Package

The package statement must be the first statement of the file. All files should belong to a specific package.

The package statement location is enforced by the Java language. Letting all files belong to an actual (rather than the Java default) package enforces Java language object oriented programming techniques.

Import

The import statements must follow the package statement. Import statements should be sorted with the most fundamental packages first, and grouped with associated packages together and one blank line between groups.

```
import java.io.IOException;
import java.net.URL;

import java.rmi.RmiServer;
import java.rmi.server.Server;

import javax.swing.JPanel;
import javax.swing.event.ActionEvent;

import org.apache.server.SoapServer;
```

The import statement location is enforced by the Java language. The sorting makes it simple to browse the list when there are many imports. It also makes it simple to determine the dependencies of the present package. The grouping reduces complexity by collapsing related information into a common unit.

Explicitly List Imported Classes

```
import java.util.List;           // NOT: import java.util.*;
import java.util.ArrayList;
import java.util.HashSet;
```

Importing classes explicitly gives an excellent documentation value for the class at hand, and makes the class easier to comprehend and maintain. Appropriate tools should be used in order to always keep the import list minimal and up to date.

Class and Interface Declarations

Class and Interface declarations should be organized in the following manner:

- Class/Interface documentation.
- class or interface statement.
- Order class (static) variables as follows: public, protected, package (no access modifier), private.
- Order instance variables as follows: public, protected, package (no access modifier), private.
- Constructors.
- Methods (no specific order).

Reduce complexity by making the location of each class element predictable.

Order of Method Modifiers

Method modifiers should be given in the following order:

```
<access> static abstract synchronized <unusual> final native
```

The <access> modifier (if present) must be the first modifier.

```
public static double square(double a); // NOT: static public double
square(double a);
```

<access> is one of public, protected or private, while <unusual> includes volatile and transient. The most important lesson here is to keep the access modifier as the first modifier. Of the possible modifiers, this is by far the most important, and it must stand out in the method declaration. For the other modifiers, the order is less important, but it make sense to have a fixed convention.

Make Type Conversions Explicitly

Type conversions must always be done explicitly. Never rely on an implicit type conversion.

```
floatValue = (int) intValue; // NOT: floatValue = intValue;
```

By this, the programmer indicates that he is aware of the different types involved, and that the mix is intentional.

Variable Initialization

Variables should be initialized where they are declared, and they should be declared in the smallest scope possible.

This ensures that variables are valid at any time. Sometimes it is impossible to initialize a variable to a valid value where it is declared. In these cases it should be left uninitialized, rather than initialized to some phony value.

Unique Variable Meanings

Variables must never have dual meanings.

This convention enhances readability by ensuring all concepts are represented uniquely. It also reduces the chance of error by side effects.

Never Declare Class Variables as Public

Class variables should never be declared public.

The concept of Java information hiding and encapsulation is violated by public variables. Use private variables and access functions instead. One exception to this rule is when the class is essentially a data structure, with no behavior (equivalent to a C++ struct). In this case it is appropriate to make the class' instance variables public.

Array Declarations

Arrays should be declared with their brackets next to the type.

```
double[] vertex; // NOT: double vertex[];
int[]    count;  // NOT: int    count[];

public static void main(String[] arguments)

public double[] computeVertex()
```

The reason for is twofold. First, the array is a feature of the class, not the variable. Second, when returning an array from a method, it is not possible to have the brackets with other than the type (as shown in the last example).

Variable Scope

Variables should be kept alive for as short a time as possible.

Keeping the operations on a variable within a small scope, it is easier to control the effects and side effects of the variable.

for() Construction

Only loop control statements must be included in the `for()` construction.

```
sum = 0;                                // NOT: for (i = 0, sum = 0; i < 100;
i++) {
for (i = 0; i < 100; i++) {              sum += value[i];
    sum += value[i];                    }
}
```

This convention increases maintainability and readability. It also makes a clear distinction of what controls and what is contained in the loop.

Initialization of Loop Variables

Loop variables should be initialized immediately before the loop.

```
isDone = false;                        // NOT: bool isDone = false;
while (!isDone) {                      //    ...
    ...                               //    while (!isDone) {
}                                     //    ...
//    }
```

Avoid do-while Loops

The use of `do-while` loops can be avoided.

`do-while` loops are less readable than ordinary `while` loops and `for` loops, since the conditional is at the bottom of the loop. The reader must scan the entire loop in order to understand the scope of the loop.

In addition, `do-while` loops are not needed. Any `do-while` loop can easily be rewritten into a `while` loop or a `for` loop. Reducing the number of constructs used enhance readability.

Avoid using break and continue in Loops

The use of `break` and `continue` in loops should be avoided.

These statements should only be used if they enhance readability over their structured counterparts.

Avoid Complex Conditional Expressions

Complex conditional expressions must be avoided. Introduce temporary Boolean variables instead.

```
bool isFinished = (elementNo < 0) || (elementNo > maxElement);
bool isRepeatedEntry = elementNo == lastElement;
if (isFinished || isRepeatedEntry) {
    ...
}

// NOT:
if ((elementNo < 0) || (elementNo > maxElement) ||
    elementNo == lastElement) {
    ...
}
```

By assigning Boolean variables to expressions, the program is automatically documented. The construction will be easier to read, debug and maintain.

Exceptions in if/else Statements

The nominal case should be put in the `if` part of the `if-else` statement, and the exception in the `else` part of the statement.

```
boolean isOk = readFile(fileName);
if (isOk) {
    ...
}
else {
    ...
}
```

Makes sure that the exception does not obscure the normal path of execution. This is important for both readability and performance.

Conditionals on Separate Lines

The conditional should be put on a separate line.

```
if (isDone)          // NOT: if (isDone) doCleanup();
    doCleanup();
```

This is for debugging purposes. When writing on a single line, it is not apparent whether the test is really true or not.

Avoid Executable Statements in Conditionals

Executable statements in conditionals must be avoided.

```
InputStream stream = File.open(fileName, "w");
if (stream != null) {
    ...
}

// NOT:
if (File.open(fileName, "w") != null) {
    ...
}
```

Conditionals with executable statements are simply very difficult to read. This is especially true for programmers new to Java.

Avoid Magic Numbers

The use of magic numbers in the code should be avoided. Numbers other than 0 and 1 can be declared as named constants instead.

```
private static final int TEAM_SIZE = 11;
:
Player[] players = new Player[TEAM_SIZE]; // NOT: Player[] players =
new Player[11];
```

If the number does not have an obvious meaning by itself, the readability is enhanced by introducing a named constant instead.

Floating Point Constants

- Floating point constants should always be written with a decimal point and at least one decimal.

```
double total = 0.0;    // NOT: double total = 0;
double speed = 3.0e8;  // NOT: double speed = 3e8;

double sum;
...
sum = (a + b) * 10.0;
```

This emphasize the different nature of integer and floating point numbers. Mathematically the two models are completely different and non-compatible concepts.

Also, as in the last example above, it emphasizes the type of the assigned variable (sum) at a point in the code where this might not be evident.

- Floating point constants should always be written with a digit before the decimal point.

```
double total = 0.5; // NOT: double total = .5;
```

The number and expression system in Java is borrowed from mathematics, and one should adhere to mathematical conventions for syntax wherever possible. Also, 0.5 is more readable than .5; there is no way it can be confused with the integer 5.

Referencing Status Variables and Methods

Static variables or methods must always be referenced through the class name, and never through an instance variable.

```
Thread.sleep(1000); // NOT: thread.sleep(1000);
```

This emphasizes that the element referenced is static and independent of any particular instance. For the same reason the class name should also be included when a variable or method is accessed from within the same class.

Layout and Comments

Note: Where applicable, code formatting tools included with Eclipse should be used to handle default code formatting conventions.

Indentation

Basic indentation should be 4.

```
for (i = 0; i < nElements; i++)
    a[i] = 0;
```

Indentation is used to emphasize the logical structure of the code. Indentation of 1 is too small to achieve this. Indentation larger than 4 results in deeply nested code that is difficult to read and that increases the chance that the lines must be split.

Block Layout

Block layout should be as illustrated below:

```
while (!done) {
    doSomething();
    done = moreToDo();
}
```

Class and Interface Declarations

The class and interface declarations should have the following form:

```
class Rectangle extends Shape
    implements Cloneable, Serializable
{
    ...
}
```

This follows from the general block rule above. Note that it is common in the Java developer community to have the opening bracket at the end of the line of the class keyword. This is not recommended.

Method Definitions

Method definitions should have the following form:

```
public void someMethod()
    throws SomeException
{
    ...
}
```

See comment on class statements above.

if-else Statements

The `if-else` class of statements should have the following form:

```
if (condition) {
    statements;
}

if (condition) {
    statements;
} else {
    statements;
}

if (condition) {
    statements;
} else if (condition) {
    statements;
} else {
    statements;
}
```


for Statements

The `for` statement should have the following form:

```
for (initialization; condition; update) {  
    statements;  
}
```

This follows from the general block rule above.

Empty for Statements

An empty `for` statement should have the following form:

```
for (initialization; condition; update)  
    ;
```

This emphasize the fact that the `for` statement is empty, and makes it obvious that the statement is empty intentionally.

while Statements

The `while` statement should have the following form:

```
while (condition) {  
    statements;  
}
```

This follows from the general block rule above.

do-while Statements

The `do-while` statement should have the following form:

```
do {  
    statements;  
} while (condition);
```

This follows from the general block rule above.

switch Statements

The `switch` statement should have the following form:

```
switch (condition) {
    case ABC :
        statements;
        // FALLTHROUGH

    case XYZ :
        statements;
        break;

    default :
        statements;
        break;
}
```

This differs slightly from the Sun recommendation both in indentation and spacing. In particular, each `case` keyword is indented relative to the `switch` statement as a whole. This makes the entire `switch` statement stand out. Note also the extra space before the `:` character. The explicit `Fallthrough` comment should be included whenever there is a `case` statement without a `break` statement. Leaving the `break` out is a common error, and it must be made clear that it is intentional when it is not there.

try-catch Statements

A `try-catch` statement should have the following form:

```
try {
    statements;
} catch (Exception exception) {
    statements;
}

try {
    statements;
} catch (Exception exception) {
    statements;
} finally {
    statements;
}
```

This follows partly from the general block rule above. This form differs from the Sun recommendation in the same way as the `if-else` statement described above.

Use Brackets with Single-Statement if-else, for, or while

Single statement `if-else`, `for`, or `while` statements should be written with brackets.

```
if (condition) {
    statement;
}

while (condition) {
    statement;
}

for (initialization; condition; update) {
    statement;
}
```

It is a common recommendation (Sun Java recommendation included) that brackets should always be used in all these cases.

Whitespace

- Operators should be surrounded by a space character.
- Java reserved words should be followed by a white space.
- Commas should be followed by a white space.
- Colons should be surrounded by white space.
- Semicolons in `for` statements should be followed by a space character.

```
a = (b + c) * d; // NOT: a=(b+c)*d
while (true) { // NOT: while(true){
    ...
doSomething(a, b, c, d); // NOT: doSomething(a,b,c,d);
case 100 : // NOT: case 100:
for (i = 0; i < 10; i++) { // NOT: for(i=0;i<10;i++){
    ...
```

These rules make the individual components of the statements stand out and enhances readability. It is difficult to give a complete list of the suggested use of whitespace in Java code. The examples above, however, should give a general idea of the intentions.

Logical Units in a Block

Logical units within a block should be separated by one blank line.

```
// Create a new identity matrix
Matrix4x4 matrix = new Matrix4x4();

// Precompute angles for efficiency
double cosAngle = Math.cos(angle);
double sinAngle = Math.sin(angle);

// Specify matrix as a rotation transformation
matrix.setElement(1, 1, cosAngle);
matrix.setElement(1, 2, sinAngle);
matrix.setElement(2, 1, -sinAngle);
matrix.setElement(2, 2, cosAngle);

// Apply rotation
transformation.multiply(matrix);
```

This convention enhances readability by introducing white space between logical units. Each block is often introduced by a comment, as indicated in the example above.

Don't Substitute Comments for Clear Code

Tricky code should not be commented, but rewritten.

In general, the use of comments should be minimized by making the code self-documenting by appropriate name choices and an explicit logical structure.

Javadoc Comments

Javadoc comments should have the following form:

```
/**
 * Return lateral location of the specified position.
 * If the position is unset, NaN is returned.
 *
 * @param x      X coordinate of position.
 * @param y      Y coordinate of position.
 * @param zone   Zone of position.
 * @return       Lateral location.
 * @throws IllegalArgumentException If zone is <= 0.
 */
public double computeLocation(double x, double y, int zone)
    throws IllegalArgumentException
{
    ...
}
```

A readable form is important because this type of documentation is typically read more often inside the code than as processed text.

Note in particular:

- The opening `/**` characters appear on a separate line.
- Subsequent `*` characters are aligned with the first one above.
- A space is inserted after each `*` character.
- An empty line is inserted description and parameter sections.
- Parameter descriptions are vertically aligned.
- Appropriate punctuation appears at the end of each parameter description.
- No blank line appears between the documentation block and the method/class.

Javadoc of class members can be specified on a single line, as follows:

```
/** Number of connections to this database */
private int nConnections_;
```

Use a Space after the Comment Identifier

There should be a space after the comment identifier.

<code>// This is a comment</code>	NOT: <code>//This is a comment</code>
<code>/**</code>	NOT: <code>/**</code>
<code> * This is a javadoc</code>	<code>*This is a javadoc</code>
<code> * comment</code>	<code>*comment</code>
<code> */</code>	<code>*/</code>

This convention improves readability by making the text stand out.

Use `//` for non-Javadoc Comments

Use `//` for all non-Javadoc comments, including multi-line comments.

```
// Comment spanning
// more than one line.
```

Since multilevel Java commenting is not supported, using `//` comments ensure that it is always possible to comment out entire sections of a file using `/* */` for debugging purposes, etc.

Indent Comments Relative to the Code

Comments should be indented relative to their position in the code.

```
while (true) {           // NOT:  while (true) {  
    // Do something      // Do something  
    something();         something();  
}
```

This is to avoid that the comments break the logical structure of the program.

Comments for Anonymous Collection Variables

The declaration of anonymous collection variables should be followed by a comment stating the common type of the elements of the collection.

```
private Vector  points_;    // of Point  
private Set     shapes_;    // of Shape
```

Without the extra comment, it can be hard to figure out what the collection consists of, and thereby how to treat the elements of the collection. In methods that take collection variables as input, the common type of the elements should be given in the associated Javadoc comment.

Whenever possible one should qualify the collection with the type to make the comment superfluous:

```
private Vector<Point>  points_;  
private Set<Shape>     shapes_;
```

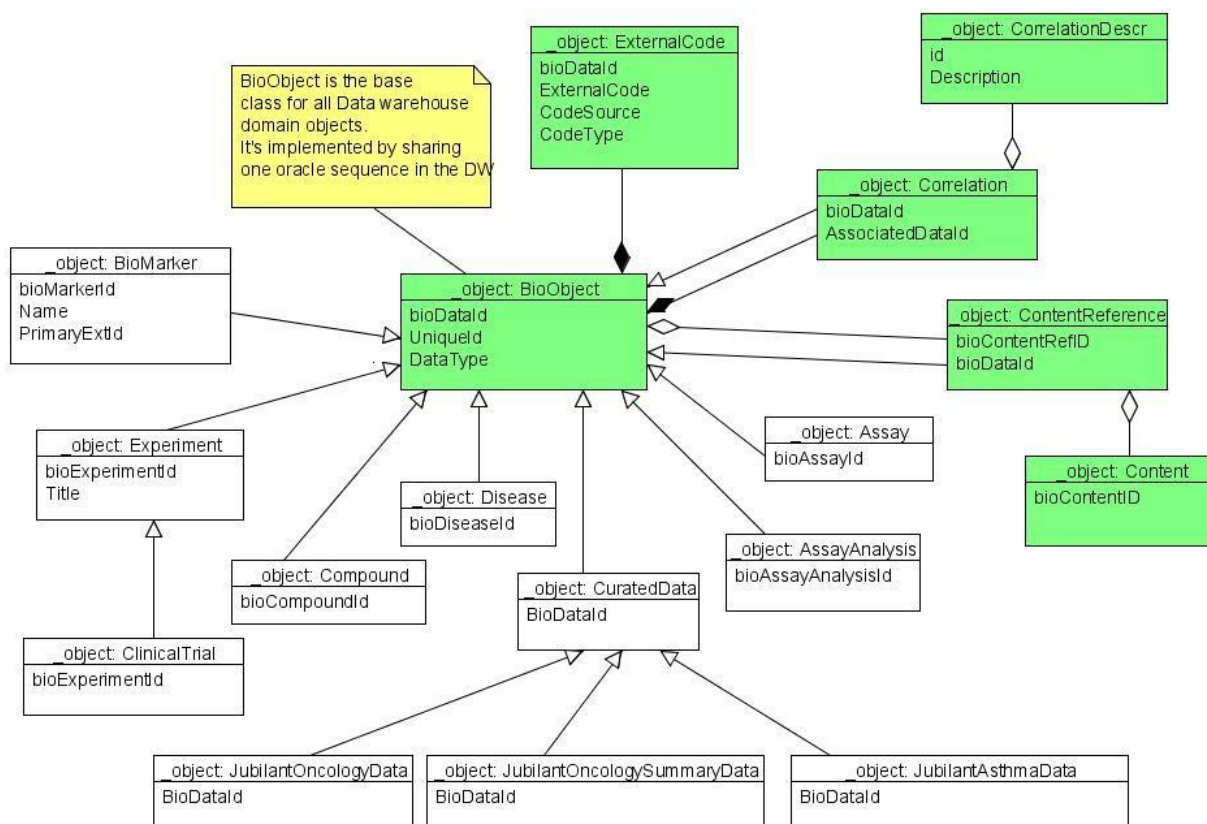
Comments for Public Classes and Public/Protected Functions

All public classes, and public and protected functions within public classes, should be documented using the Java documentation (Javadoc) conventions. This makes it easy to keep up-to-date online code documentation.

Chapter 8

Other Topics

Object Base Class Diagram



Z-Score Calculation

Step 1. Data Pre-processing

Gene expression data

If a probeset's raw intensity is 0 or null, its raw intensity will be set to:
 $0.5 * \min(\text{the probeset's raw intensity in its trial})$.

RBM data

If a probeset's value is 0 or null, its value will be set to:
 $0.5 * \min(\text{abs}(\text{the probeset's value in its trial})) * \text{sign of the probeset's value in its trial}.$

Protein Data

If a probeset's intensity is 0 or null, its raw intensity will be set to:
 $0.5 * \min(\text{abs}(\text{the probeset's raw intensity in its trial}.$

Transcript data

If a probeset's raw intensity is 0 or null, its raw intensity will be set to:
 $0.5 * \min(\text{the probeset's raw intensity in its trial}).$

Step 2. Z-Score Calculation

The formula used for standard score or Z score is:

$$z = \frac{x - \mu}{\sigma},$$

where:

- **x** is a raw score to be standardized
- **μ** is the [- **σ** is the [

Gene expression data

The population is a combination of trial, time point, gene, and probeset.

RBM data

The population is a combination of trial, time point, and antigen.

Protein Data

The population is a combination of trial, time point, and component.

Transcript data

The population is a combination of trial, time point, and gene.

Step 3. Data Post-Processing

Make a copy of the original calculated Z Score to the zscore_org column, and the column zscore will be used for the heat map.

If calculated Z Score > 3, then its Z Score is set to 3; if calculated Z Score < -3, then its Z Score is set to -3.

Sample Z-Score Calculation Scripts

Gene Expression and Transcript Data

```
- *****
- Compute mean and stddev
- *****
create table microarray_mean_stddev AS
select trial_name, timepoint, gene_symbol, probeset,
avg(RAW_INTENSITY) as mean_value,
stddev(RAW_INTENSITY) as stddev_value
from deapp.de_subject_microarray_data
group by trial_name, timepoint, gene_symbol, probeset;

- *****
- Compute Z Score
- *****
update de_subject_microarray_data t1
set zscore=(t1.RAW_INTENSITY -
(select t2.mean_value from microarray_mean_stddev t2
where t2.trial_name=t1.trial_name and t2.gene_symbol=t1.gene_symbol and
t1.timepoint=t2.timepoint and t1.probeset=t2.probeset))/(select t3.stddev_value
from microarray_mean_stddev t3
where t3.trial_name=t1.trial_name and t3.gene_symbol=t1.gene_symbol and
t1.timepoint=t3.timepoint and t1.probeset=t3.probeset and
t3.stddev_value!=0);

- *****
- Data Post-processing
- *****
update de_subject_microarray_data set zscore_org=zscore;
update de_subject_microarray_data set zscore=3 where zscore>3;
update de_subject_microarray_data set zscore=-3 where zscore<-3;
```

RBM Data

```
- *****
- Identify min(value) for value is null or 0
- *****
create table rbm_min_tmp as
select trial_name, antigen_name, 0.5*min(value) as min_value,
abs(0.5*min(value)) as abs_value
from de_subject_rbm_data
where value >0
```

Z-Score Calculation

```
group by trial_name, antigen_name
union
select trial_name, antigen_name, 0.5*max(value) as min_value,
abs(0.5*max(value)) as abs_value
from de_subject_rbm_data
where value < 0
group by trial_name, antigen_name;

create table rbm_min as
select t1.trial_name, t1.antigen_name, t1.min_value
from rbm_min_tmp t1, rbm_min_tmp t2
where t1.trial_name=t2.trial_name and t1.antigen_name=t2.antigen_name and
t1.abs_value<t2.abs_value;

- *****
- Set n_value(value)= min(value)
- if value is null or 0
- *****
update de_subject_rbm_data t1
set n_value=(select min_value from rbm_min t2
where t2.trial_name=t1.trial_name and t2.antigen_name=t1.antigen_name)
where t1.n_value is null or t1.n_value=0;

- *****
- Compute mean and stddev
- *****
create table rbm_mean_stddev AS
select trial_name, antigen_name, timepoint, avg(n_value) as mean_value,
stddev(n_value) as stddev_value
from de_subject_rbm_data
group by trial_name, antigen_name, timepoint;

- *****
- Compute Z Score
- *****
update de_subject_rbm_data t1
set zscore=(t1.n_value-(select t2.mean_value from rbm_mean_stddev t2
where t2.trial_name=t1.trial_name and t2.antigen_name=t1.antigen_name and
t1.timepoint=t2.timepoint))/(select t3.stddev_value
from rbm_mean_stddev t3 where t3.trial_name=t1.trial_name and
t3.antigen_name=t1.antigen_name and
t1.timepoint=t3.timepoint and t3.stddev_value!=0);

- *****
- Data Post-processing
- *****
update de_subject_rbm_data set zscore_org=zscore;
update de_subject_rbm_data set zscore=3 where zscore>3;
update de_subject_rbm_data set zscore=-3 where zscore<-3;
```

Protein Data

```
- *****
- Identify min(intensity) for intensity is null or 0
- intensity >= 0 or null
- *****
create table protein_min as
```

```

select trial_name, COMPONENT, 0.5*min(intensity) as min_value
from de_subject_protein_data
where intensity >0
group by trial_name, COMPONENT;

- *****
- Set n_value(intensity)= min(intensity)
- if intensity is null or 0
- *****
update de_subject_protein_data t1
set n_value=(select min_value from protein_min t2
where t2.trial_name=t1.trial_name and t2.component=t1.component)
where t1.n_value is null or t1.n_value=0;

- *****
- Compute mean and stddev
- *****
create table protein_mean_stddev AS
select trial_name, COMPONENT, timepoint, avg(n_value) as mean_value,
stddev(n_value) as stddev_value
from de_subject_protein_data
group by trial_name, COMPONENT, timepoint;

- *****
- ZScore(intensity)= [n_value(intensity)-mean(n_value)]/stddev(n_value)
- *****
update de_subject_protein_data t1
set zscore=(t1.n_value-(select t2.mean_value from protein_mean_stddev t2
where t2.trial_name=t1.trial_name and t2.component=t1.component and
t1.timepoint=t2.timepoint))/(select t3.stddev_value from protein_mean_stddev t3
where t3.trial_name=t1.trial_name and t3.component=t1.component and
t1.timepoint=t3.timepoint and t3.stddev_value!=0);

- *****
- Data Post-processing
- *****
update de_subject_protein_data set zscore_org=zscore;
update de_subject_protein_data set zscore=3 where zscore>3;
update de_subject_protein_data set zscore=-3 where zscore<-3;

```

Stored Procedures to Retrieve Z-Score for Heat Maps

Gene Expression and Transcript data

```

CREATE OR REPLACE PROCEDURE DEAPP."MICROARRAY_COMPARISON_QRY" (
patient_ids IN VARCHAR2, -- CSV list of patient IDs
sample_types IN VARCHAR2, -- CSV list of concept cds to use for filtering
-- pathway_name IN VARCHAR2, - Name of pathway to use for filtering
pathway_uid1 IN VARCHAR2, -- A Unique pathway ID from BIO_DADA_UID to use for
filtering
timepoints IN VARCHAR2, -- CSV list of timepoint concept codes
cv_1 IN OUT SYS_REFCURSOR --Resultset in Cursor for iteration by caller
)
AS

```

```

--Counter to check if samples exist.
sample_record_count INTEGER;
timepoint_count INTEGER;
BEGIN
-----
-- Returns PROBESET, GENE_SYMBOL, REFSEQ, LOG10_INTENSITY, PVALUE, PATIENT_ID,
ASSAY_ID
-- result set for specifed Pathways filtered
-- by Sample Types and Subject_id.
-- KCR@20090206 - First rev.
-- KCR@20090212 - Second rev. Changed logic so that if Sampel is not found it
returns Zero.
-- KCR@20090206 - Third rev. Using Collections to hold parsed values instead of
DB table.
-- HX@20090317 - Replace pathway_name with pathway_uid
-- HX@20090318 - Add pathway_uid column into DE_PATHWAY and populate data
-- from BIOMART.BIO_DATA_UID
-- 2009-05-04: replace refseq with probeset
-- 2009-05-26: remove GENE_SYMBOL's concatenation and change probeset to refseq
-- 2009-05-29: change LOG10_INTENSITY to LOG2_INTENSITY
-- 2009-06-18: add raw_intensity and change log2_intensity to zscore
-- 2009-06-23: Add timepoints as a parameter
-----
-- Check if sample Types Exist
SELECT COUNT(*)
INTO sample_record_count
FROM DE_SUBJECT_SAMPLE_MAPPING
WHERE concept_code IN
--Passing string to Text parser Function
(SELECT * from table(text_parser(sample_types)));
--Sample Record Count is invalid or non existent.
IF sample_record_count = 0
THEN
BEGIN
select count(*) into timepoint_count
from table(text_parser(timepoints));
if timepoint_count=0 then
OPEN cv_1 FOR
select distinct a.PROBESET, a.GENE_SYMBOL, a.refseq, a.zscore as LOG2_INTENSITY,
a.PVALUE, a.patient_ID, a.ASSAY_ID, a.raw_intensity
FROM DE_SUBJECT_MICROARRAY_DATA a, DE_pathway_gene c, de_pathway p
where p.pathway_uid= pathway_uid1 and c.pathway_id= p.id and
a.gene_symbol = c.gene_symbol and
a.patient_id IN (SELECT * from table(text_parser(patient_ids)))
order by a.GENE_SYMBOL, a.PROBESET ;
else
OPEN cv_1 FOR
select distinct a.PROBESET, a.GENE_SYMBOL, a.refseq, a.zscore as LOG2_INTENSITY,
a.PVALUE, a.patient_ID, a.ASSAY_ID, a.raw_intensity
FROM DE_SUBJECT_MICROARRAY_DATA a, DE_pathway_gene c, de_pathway p,
DE_subject_sample_mapping b
where p.pathway_uid= pathway_uid1 and c.pathway_id= p.id and
a.gene_symbol = c.gene_symbol and
a.patient_id IN (SELECT * from table(text_parser(patient_ids))) and
b.TIMEPOINT_CD IN (SELECT * from table(text_parser(timepoints))) and
a.PATIENT_ID=b.patient_id and a.timepoint=b.timepoint and
a.assay_id=b.assay_id

```

```

order by a.GENE_SYMBOL, a.PROBESET ;
end if;
END;
--else use all filters (If Subject is non existent or invalid, then return
ELSE
BEGIN
if timepoint_count=0 then
OPEN cv_1 FOR
select distinct a.PROBESET, a.GENE_SYMBOL, a.refseq, a.zscore as LOG2_INTENSITY,
a.PVALUE, a.patient_ID, a.ASSAY_ID, a.raw_intensity
FROM DE_SUBJECT_MICROARRAY_DATA a, DE_pathway_gene c, de_pathway p,
DE_subject_sample_mapping b
where p.pathway_uid= pathway_uid1 and c.pathway_id= p.id and
a.gene_symbol = c.gene_symbol and
a.PATIENT_ID = b.PATIENT_ID and a.assay_id = b.assay_id and
b.concept_code IN (SELECT * from table(text_parser(sample_types))) and
a.patient_id IN (SELECT * from table(text_parser(patient_ids)))
order by a.GENE_SYMBOL, a.PROBESET;
else
OPEN cv_1 FOR
select distinct a.PROBESET, a.GENE_SYMBOL, a.refseq, a.zscore as LOG2_INTENSITY,
a.PVALUE, a.patient_ID, a.ASSAY_ID, a.raw_intensity
FROM DE_SUBJECT_MICROARRAY_DATA a, DE_pathway_gene c, de_pathway p,
DE_subject_sample_mapping b
where p.pathway_uid= pathway_uid1 and c.pathway_id= p.id and
a.gene_symbol = c.gene_symbol and
a.PATIENT_ID = b.PATIENT_ID and a.assay_id = b.assay_id and
b.concept_code IN (SELECT * from table(text_parser(sample_types))) and
a.patient_id IN (SELECT * from table(text_parser(patient_ids))) and
b.TIMEPOINT_CD IN (SELECT * from table(text_parser(timepoints))) and
a.PATIENT_ID=b.patient_id and a.timepoint=b.timepoint
order by a.GENE_SYMBOL, a.PROBESET;
end if;
END;
END IF;
END microarray_comparison_qry;
/

```

RBM data

```

CREATE OR REPLACE PROCEDURE DEAPP."RBM_COMPARISON_QRY" (
patient_ids IN VARCHAR2, -- list of patient IDs in CSV
concept_cds IN VARCHAR2, -- list of concept_cds in CSV
timepoints IN VARCHAR2, -- list of timepoint concept_cds in CSV
cv_1 IN OUT SYS_REFCURSOR --Resultset in Cursor for iteration by caller
)
AS
tp_cnt INTEGER;
concept_cnt INTEGER;
BEGIN
-----
-- Returns ANTIGENE, GENE_SYMBOL, VALUE, PATIENT_ID, ASSAY_ID
-- result set for specifed Pathways filtered
-- by Sample Types and Subject_id.
-- HX@20090317 - First rev.
-- HX@20090319 - Rename DE_RBM_DATA to DE_SUBJECT_RBM_DATA and move CONCEPT_CD
-- from DE_RBM_DATA to DE_SUBJECT_SAMPLE_MAPPING, and make the query

```

```

-- and table names are consistent with MicroArray's one
-- 2009-06-18: change normalized_value to zscore
-- 2009-06-23: Add timepoints as parameter
-- 2009-06-30: Remove t1.zscore is not null condition
-----

select count(*) into tp_cnt
from de_subject_sample_mapping
where timepoint_cd in
(select * from table(text_parser(timepoints)));

select count(*) into concept_cnt
from de_subject_sample_mapping
where concept_code in
(select * from table(text_parser(concept_cds)));

if (tp_cnt=0) then
if(concept_cnt>0) then
OPEN cv_1 FOR
select distinct t1.ANTIGEN_NAME, t1.GENE_SYMBOL, t1.zscore as value,
t1.patient_id, t1.assay_id
FROM DE_SUBJECT_RBM_DATA t1, de_subject_sample_mapping t2
where -- t1.zscore is not null and
t2.patient_id IN (SELECT * from table(text_parser(patient_ids))) and
t2.concept_code in (select * from TABLE(text_parser(concept_cds))) and
t1.data_uid=t2.data_uid and t1.assay_id=t2.assay_id;
else
OPEN cv_1 FOR
select distinct t1.ANTIGEN_NAME, t1.GENE_SYMBOL, t1.zscore as value,
t1.patient_id, t1.assay_id
FROM DE_SUBJECT_RBM_DATA t1
where -- t1.zscore is not null and
t1.patient_id IN (SELECT * from table(text_parser(patient_ids)));
end if;
else
if(concept_cnt=0) then
OPEN cv_1 FOR
select distinct t1.ANTIGEN_NAME, t1.GENE_SYMBOL, t1.zscore as value,
t1.patient_id, t1.assay_id
FROM DE_SUBJECT_RBM_DATA t1, de_subject_sample_mapping t2
where --t1.zscore is not null and
t2.patient_id IN (SELECT * from table(text_parser(patient_ids))) and
t2.timepoint_cd in (select * from TABLE(text_parser(timepoints))) and
t1.data_uid = t2.data_uid and t1.assay_id=t2.assay_id;
else
OPEN cv_1 FOR
select distinct t1.ANTIGEN_NAME, t1.GENE_SYMBOL, t1.zscore as value,
t1.patient_id, t1.assay_id
FROM DE_SUBJECT_RBM_DATA t1, de_subject_sample_mapping t2
where --t1.zscore is not null and
t2.patient_id IN (SELECT * from table(text_parser(patient_ids))) and
t2.concept_code in (select * from TABLE(text_parser(concept_cds))) and
t2.timepoint_cd in (select * from TABLE(text_parser(timepoints))) and
t1.data_uid = t2.data_uid and t1.assay_id=t2.assay_id;
end if;
end if;
END RBM_comparison_gry;
/

```

Protein data

```

CREATE OR REPLACE PROCEDURE DEAPP."PROTEIN_COMPARISON_QRY" (
patient_ids IN VARCHAR2, -- CSV list of patient IDs
sample_types IN VARCHAR2, -- CSV list of concept cds to use for filtering
pathway_uid1 IN VARCHAR2, -- A Unique pathway ID from BIO_DADA_UID to use for
filtering
timepoints IN VARCHAR2, -- CSV list of timepoint concept codes
cv_1 IN OUT SYS_REFCURSOR --Resultset in Cursor for iteration by caller
)
AS
--Counter to check if samples exist.
sample_record_count INTEGER;
timepoint_count INTEGER;
BEGIN
-----
-- Returns PROBESET, GENE_SYMBOL, REFSEQ, LOG10_INTENSITY, PVALUE, PATIENT_ID,
ASSAY_ID
-- result set for specifed Pathways filtered
-- by Sample Types and Subject_id.
-- KCR@20090206 - First rev.
-- KCR@20090212 - Second rev. Changed logic so that if Sampel is not found it
returns Zero.
-- KCR@20090206 - Third rev. Using Collections to hold parsed values instead of
DB table.
-- HX@20090317 - Replace pathway_name with pathway_uid
-- HX@20090318 - Add pathway_uid column into DE_PATHWAY and populate data
-- from BIOMART.BIO_DATA_UID
-- 2009-05-04: replace refseq with probeset
-- 2009-05-26: remove GENE_SYMBOL's concatenation and change probeset to refseq
-- 2009-05-29: change LOG10_INTENSITY to LOG2_INTENSITY
-- 2009-06-18: add raw_intensity and change log2_intensity to zscore
-- 2009-06-23: Add timepoints as a parameter
-----
-- Check if sample Types Exist
SELECT COUNT(*)
INTO sample_record_count
FROM DE_SUBJECT_SAMPLE_MAPPING
WHERE concept_code IN
--Passing string to Text parser Function
(SELECT * from table(text_parser(sample_types)));
--Sample Record Count is invalid or non existent.
IF sample_record_count = 0
THEN
BEGIN
select count(*) into timepoint_count
from table(text_parser(timepoints));
if timepoint_count=0 then
OPEN cv_1 FOR
select distinct a.component, a.GENE_SYMBOL, a.zscore,
a.patient_ID, a.ASSAY_ID, a.intensity
FROM DE_SUBJECT_PROTEIN_DATA a, DE_pathway_gene c, de_pathway p
where p.pathway_uid= pathway_uid1 and c.pathway_id= p.id and
a.gene_symbol = c.gene_symbol and
a.patient_id IN (SELECT * from table(text_parser(patient_ids)))
order by a.GENE_SYMBOL, a.COMPONENT ;

```

```

else
OPEN cv_1 FOR
select distinct a.COMPONENT, a.GENE_SYMBOL, a.zscore,
a.patient_ID, a.ASSAY_ID, a.intensity
FROM DE_SUBJECT_PROTEIN_DATA a, DE_pathway_gene c, de_pathway p,
DE_subject_sample_mapping b
where p.pathway_uid= pathway_uid1 and c.pathway_id= p.id and
a.gene_symbol = c.gene_symbol and
a.patient_id IN (SELECT * from table(text_parser(patient_ids))) and
b.TIMEPOINT_CD IN (SELECT * from table(text_parser(timepoints))) and
a.PATIENT_ID=b.patient_id and a.timepoint=b.timepoint and
a.assay_id=b.assay_id
order by a.GENE_SYMBOL, a.COMPONENT;
end if;
END;
--else use all filters (If Subject is non existent or invalid, then return
ELSE
BEGIN
if timepoint_count=0 then
OPEN cv_1 FOR
select distinct a.COMPONENT, a.GENE_SYMBOL, a.zscore,
a.patient_ID, a.ASSAY_ID, a.intensity
FROM DE_SUBJECT_PROTEIN_DATA a, DE_pathway_gene c, de_pathway p,
DE_subject_sample_mapping b
where p.pathway_uid= pathway_uid1 and c.pathway_id= p.id and
a.gene_symbol = c.gene_symbol and
a.PATIENT_ID = b.PATIENT_ID and a.assay_id = b.assay_id and
b.concept_code IN (SELECT * from table(text_parser(sample_types))) and
a.patient_id IN (SELECT * from table(text_parser(patient_ids)))
order by a.GENE_SYMBOL, a.COMPONENT;
else
OPEN cv_1 FOR
select distinct a.COMPONENT, a.GENE_SYMBOL, a.zscore,
a.patient_ID, a.ASSAY_ID, a.intensity
FROM DE_SUBJECT_PROTEIN_DATA a, DE_pathway_gene c, de_pathway p,
DE_subject_sample_mapping b
where p.pathway_uid= pathway_uid1 and c.pathway_id= p.id and
a.gene_symbol = c.gene_symbol and
a.PATIENT_ID = b.PATIENT_ID and a.assay_id = b.assay_id and
b.concept_code IN (SELECT * from table(text_parser(sample_types))) and
a.patient_id IN (SELECT * from table(text_parser(patient_ids))) and
b.TIMEPOINT_CD IN (SELECT * from table(text_parser(timepoints))) and
a.PATIENT_ID=b.patient_id and a.timepoint=b.timepoint
order by a.GENE_SYMBOL, a.COMPONENT;
end if;
END;
END IF;
END protein_comparison_qry;
/

```


Pathway Studio Enterprise Server

Users/Info

- pse/8iDM110J: Linux user for installation and can run PSE and the FLEX License server
- VNC for pse user: 174.129.105.99:6/rec0m321
- Oracle user: pse/pse209@pw1; pse62/pse62@DW2
- JAVA_HOME and CATALINA_HOME changed in .bashrc for user pse to run Java 1.6 JRE and Tomcat 6 deployed with PSE
- Default PSE users
 - Admin, Manager, User Roles: admin/1234
 - Manager, User Roles: manager1/12345
 - User Roles: demo1/12345, demo2/12345
- <http://localhost:8073/PathwayStudio> to log in

Start/Stop License Server

- Places and open PSE License Server, File->Open in terminal
- Start/Stop: agilm start/stop

Start/Stop Tomcat Server

- Places and open PSE Tomcat Server, File->Open in terminal
- Start/Stop: catalina.sh start/stop

Creating Schema Objects

```
pecli dbmanager -cO
```

Populating the Database

```
./pecli netimport resnet6.part1.rnef 2>&1 1>p1.log&
```

```
./pecli netimport resnet6.part2.rnef 2>&1 1>p1.log&
```

```
./pecli netimport resnet6_update_2008_Q2.rnef 2>&1 1>Q2.log&
```

```
./pecli netimport resnet6_update_2008_Q3.rnef 2>&1 1>Q3.log&
```

```
./pecli netimport resnet6_update_2008_Q4.rnef 2>&1 1>Q4.log&
```

Post-load Database Optimization

```
./pecli dbmanager -p
```

GenePattern Server Installation

GenePattern is used to generate Dataset Explorer heat maps and other visualizations.

Note: The GenePattern source code can be downloaded from:
`ftp://ftp.broadinstitute.org/pub/genepattern/src/`

On Unix

To install GenePattern on a Unix server:

1. Create a folder for GenePattern-related files outside the tomcat directory, such as `/home/appuser/gp`.
2. Install a very old version of R; GenePattern requires R 2.5.0.
3. Install the g77 Fortran compiler, the readline development libraries, and the X11 development libraries.

As root, you can do these tasks with the following commands:

```
yum install gcc-gfortran
yum install readline-devel
yum update libX11
yum install libX11-devel
yum install libXt-devel
```

4. Create the directory `/home/appuser/gp`.
5. Change to the directory `/home/appuser/gp`.
6. Download R from <http://cran.stat.ucla.edu/src/base/R-2/R-2.5.0.tar.gz>.
7. Unpack R with the following command:

```
tar xvfz R-2.5.0.tar.gz
```

8. Build the executable with the following commands:

```
./configure
make
make check
```

9. Create the folder `/home/appuser/patches`.

10. Follow the installation directions at the following site:

http://www.broadinstitute.org/cancer/software/genepattern/download/gp_server_install_wafile.html

- a. Specify the hostname within J&J, such as `transmartdev`.
- b. Specify the domain name `jj.com`.
- c. Specify port `8443` to run the server.

11. Make sure you modify the files `genepattern.properties` and `genepattern.properties.default`.

Here are a set of parameters to set (in this case, for `transmartdev.jj.com`):

- ☐ `GENEPATTERN_PORT=8443`
- ☐ `GenePatternURL=https\://transmartdev.jj.com\:8443/gp/`
- ☐ `redirect.to.fq.host=false`
- ☐ `fqHostName=transmartdev.jj.com`
- ☐ `java=/usr/local/jre1.6.0_16/bin/java`
- ☐ `run_r_path=/usr/local/tomcat-5.5.27/webapps/gp/WEB-INF/classes/`
- ☐ `R_HOME=/home/appuser/R-2.5.0/`
- ☐ `R2.5_HOME=/home/appuser/R-2.5.0/`
- ☐ `tomcat=/usr/local/tomcat-5.5.27/`
- ☐ `patches=/home/appuser/gp/patches`
- ☐ `tasklib=/home/appuser/gp/taskLib`
- ☐ `jobs=/usr/local/tomcat-5.5.27/webapps/gp/jobResults`
- ☐ `tomcatCommonLib=/usr/local/tomcat-5.5.27/gp/WEB-INF/lib`
- ☐ `webappDir=/usr/local/tomcat-5.5.27/webapps/gp/`
- ☐ `java.net.ssl.trustStore=/usr/local/tomcat-5.5.27/.keystore`

You may need to modify other parameters.

12. Configure tomcat to use SSL.

For J&J production, you will need to configure tomcat to use the Network Solutions certificates. To do so:

- a. Get the Network Solutions root certificate bundle. Download links are [here](#). Specifically, download [Network Solutions Certificate Bundles \(non EV\)](#).
- b. Convert the bundle from DER to PEM format using a command like:

```
openssl pkcs7 -in NetSol_OV.p7b -inform der -outform pem
-print_certs -out NetSol_OV.p7
```

- c. Bundle the existing certificate (being used by apache), private key (also being used by apache) and certificate chain (downloaded above) together into a PKCS12 document. The easiest way to do this is with a command like the following (substituting the actual passwords for the words "password"):

```
openssl pkcs12 -export -in /etc/httpd/conf.d/TRANSMARTDEV.JNJ.COM.crt
-inkey /etc/httpd/conf.d
/server.key -passin pass:password -passout pass:password
-out bundled.p12 -CAfile NetSol_OV.p7
-chain -name tomcat
```

13. Copy the generated file (bundled.p12) to a location like:

```
/usr/local/tomcat-5.5.27/conf
```

14. Modify the file /usr/local/tomcat-5.5.27/conf/server.xml to allow SSL connections to tomcat through port 8443. Substitute the actual password for the word "password" in the XML code below. Also, note the location of the keyfile.

```
<Connector port="8443" maxHttpHeaderSize="8192"
proxyName="transmartdev.jnj.com" proxyPort="8443"
keystoreFile="/usr/local/tomcat-5.5.27/conf/bundled.p12"
keystorePass="password" keypass="password" keystoreType="PKCS12"
keyAlias="tomcat"
maxThreads="150" minSpareThreads="25"
    MaxSpareThreads="75"
    enableLookups="false"
    disableUploadTimeout="true"
    acceptCount="100" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS" />
```

15. Restart the tomcat server (using the scripts /usr/local/tomcat-5.5.27/bin/shutdown.sh and /usr/local/tomcat-5.5.27/bin/startup.sh).

16. Be sure to modify AWS security group rules to allow connections to port 8443.

17. Log into the biomart server at a URL like:

<https://transmartdev.jnj.com:8443/gp>

18. Register the server.

19. Create a user called biomart.

20. Install all the available modules.

On Windows

To install GenePattern on a Windows server:

1. Download the Windows version from the following location:
<http://www.broadinstitute.org/cancer/software/genepattern/installer/latest/install.htm>
2. Run the downloaded executable.
3. Follow the screen prompts to install the GenePattern server and the associated R and Perl software packages.
4. Validate the server installation with the following:

`StartGenePattern.exe`

`GenePatternHome.html`

Note: Because there are so many sections of the GenePattern code that relate to host name and the installation directory, it is not practical to start from source code and manually set up libraries and classes under an existing Tomcat directory structure.

5. For debugging and software integration, it is important to run GenePattern from its Tomcat `webapps\gp` directory (rather than from the immutable `StartGenePattern.exe`).

Refer to the following document on how to install GenePattern from the war file:

http://www.broadinstitute.org/cancer/software/genepattern/download/gp_server_install_warfile.html

Note: This document does not describe manually moving libraries and classes into an existing Tomcat directory. It is still necessary to run `InstallAnywhere` included in the war file.)

To run GenePattern from the Windows command line:

1. In the `genepattern.properties` file (located in `GenePatternInstallDir\Tomcat\resources\`), make sure the property `redirect.to.fq.host` is set to `false`.

(This may not be necessary.)

2. On the command line, change to the Tomcat directory.
3. Run the following command

`bin\startup.bat`

Useful Links

Connection details for websites and tools.

Semantic Browser

<http://knoesis.wright.edu/library/tools/semanticbrowser/SemanticBrowser.htm>

Java Framework for Semantic Queries

<http://jena.sourceforge.net/>

Project Management tool at J&J

<http://cntusraweb3.na.jnj.com/tp/Default.aspx>

Login: No Login Required

Password: 1234

Rembrandt

<http://rembrandt-db.nci.nih.gov>

Login: housmand

Password: Dh0802@@@

ResNet Server Database

IP Connection String: <http://bwebs1.janbe.eu.jnj.com:10023/PathwayExpert/services/SputnikServiceSoap>

Login: demo

Password: 12345

Pictor

<http://pictor.janbe.eu.jnj.com>

Chip (MicroArray Analysis Pipeline)

<http://chip.janbe.eu.jnj.com>

OmniViz Version 6.0.1

Download Link: \\rndusljfps02\rndusdfsroot\LaJolla\All\LJPublic\OmniViz\OmniViz6.0.1\

Ingenuity

<https://analysis.ingenuity.com/ipawst/public/ipatest.jsp>

Login: dhousman@recomdata.com

Password: Madeline1

SAS Software

Base SAS 9.13 software <\\cntusmasmss01\smspkge\$\GR103195>

License key <\\cntusmasmss01\smspkge\$\GR103755>

Note: only one key is available

General J&J Software

To download and install new software applications on JnJ computers use the following URL for the GAD - <http://cntusraweb3.na.jnj.com/portaldocs/home/BusinessTools/GAD.htm>

