Course Exercises

IBM Operations Analytics Predictive Insights 1.3.3: Implementation and Configuration

Course code TN612 ERC 1.0

IBM Training

March 2016 edition

NOTICES

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

TRADEMARKS

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

ITIL is a Registered Trade Mark of AXELOS Limited.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Contents

Unit 1 Predictive Insights overview exercises

This unit has no student exercises.

Unit 2 Architectural information exercises

This unit has no student exercises.

Unit 3 Installation exercises

In the following exercises, you install InfoSphere® Streams and IBM Operations Analytics Predictive Insights on the DB2® OMNIbus, and Web GUI software that is installed and running on your virtual machine. This installation is stand-alone with most software installed under one user.

All software installation files are hosted in the /**scapi_install_files** directory in the virtual machine. A temporary file to extract and run the installers is /**software-temp**. Both these directories are owned by the installation user **scadmin**.

Appendices A through D cover the installation of DB2, Jazz™ for Service Management, OMNIbus, and Web GUI.

Exercise 1 Installing InfoSphere Streams

In this exercise, you install InfoSphere Streams. Your virtual machine has the following software:

- DB2 10.5 FP3
- OMNIbus 8.1
- Web GUI 8.1

The user names and passwords that are associated with this software are listed in the following table.

User name	Password	Comments
root	object00	DB2 was installed with root
db2inst1	object00	The instance owner of the DB2 database
root	object00	The user name of the OMNIbus database
scadmin	object00	This user name installed OMNIbus and Web GUI

Complete the following tasks:

1. Add the scadmin user to the db2iadm1 group.
   a. Log in as **scadmin** with the password **object00**.
   b. In a terminal, log in as **root** with password **object00**.

```
su -
```

c. Add **scadmin** to the **db2iadm1** group.

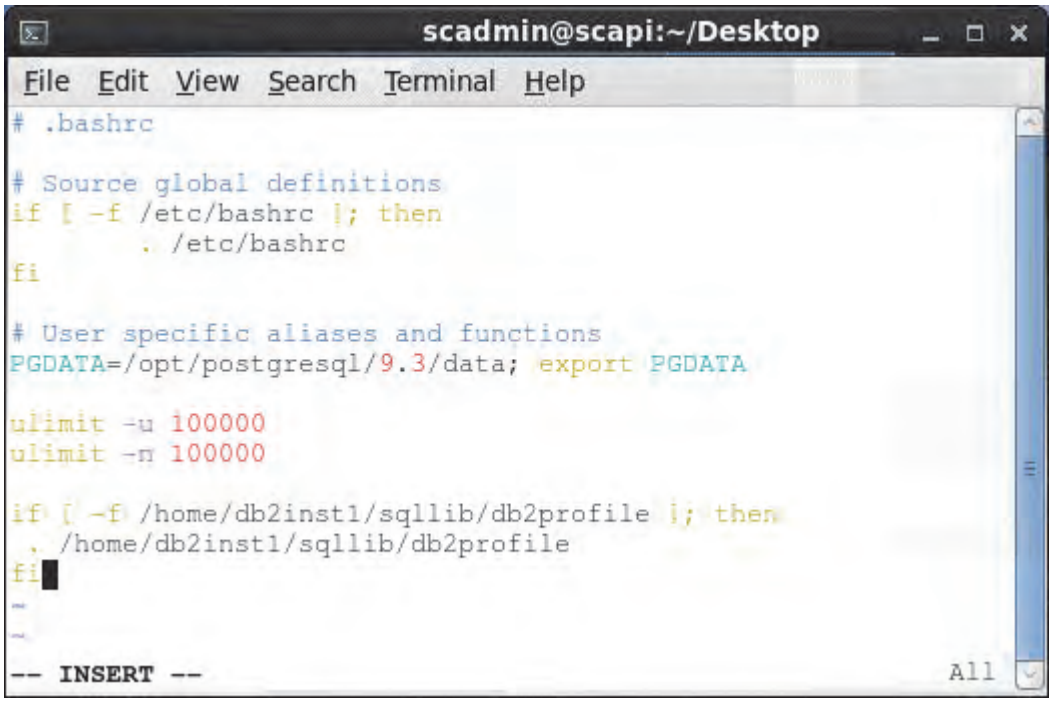
```
usermod  -a -G db2iadm1 scadmin
```

d. Exit the root user and return to the scadmin user.

2. Modify the bashrc for the scadmin user so that it can source the db2profile. Add these lines to the /home/scadmin/.bashrc file. Re-source the bashrc file to your open terminal.

a. Using your favorite editor (vi is used in the graphic), add the following lines to **/home/scadmin/.bashrc**:

```
if [ -f /home/db2inst1/sqllib/db2profile ];  then
.  /home/db2inst1/sqllib/db2profile
fi
```



b. Re-source the bashrc file.

```
.  ~/.bashrc
```

3. As the scadmin user, copy the **Streams-3.2.0.0-x86_64-el6.tar.gz** file and prepare it for installation.

a. Copy the installation file from **/scapi-install-files** to the **/software-temp** directory.

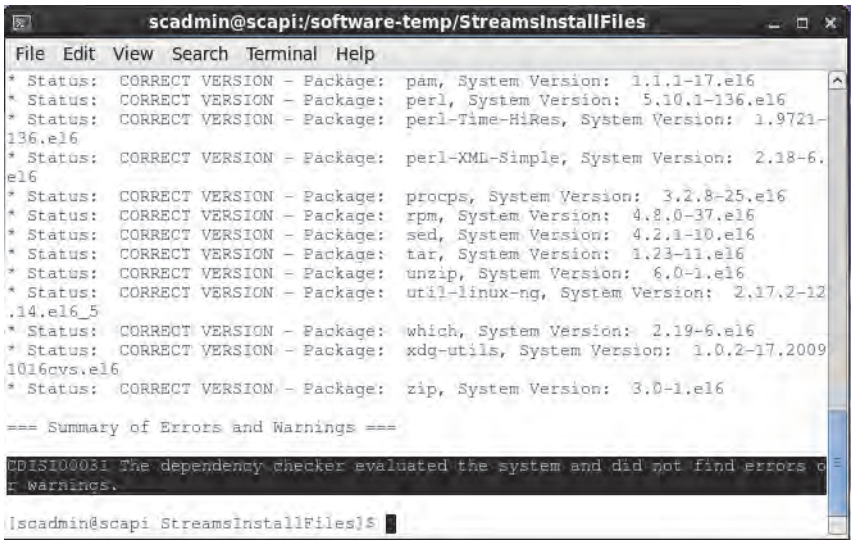
```
cp /scapi-install-files/Streams-3.2.0.0-x86_64-el6.tar.gz
/software-temp/.
```

b. Extract the installation file.

```
cd /software-temp
tar –xzf Streams-3.2.0.0-x86_64-el6.tar.gz
```

c. Run the **StreamsInstallFiles/dependency_checker.sh** script. Validate that all package requirements are met. Run dependency checker to find any missing packages.

```
cd StreamsInstallFiles
./dependency_checker.sh
```



d. Note if there are any packages that need to be installed.

Note: Since this virtual machine was prepared for you, all the necessary packages have been installed.

4. Start the installer with command **StreamsInstallFiles/InfoSphereStreamsSetup.bin.**

   a. Change directories to /software_temp/StreamsInstallfFiles.

   ```
   cd /software_temp/StreamsInstallFiles
   ```

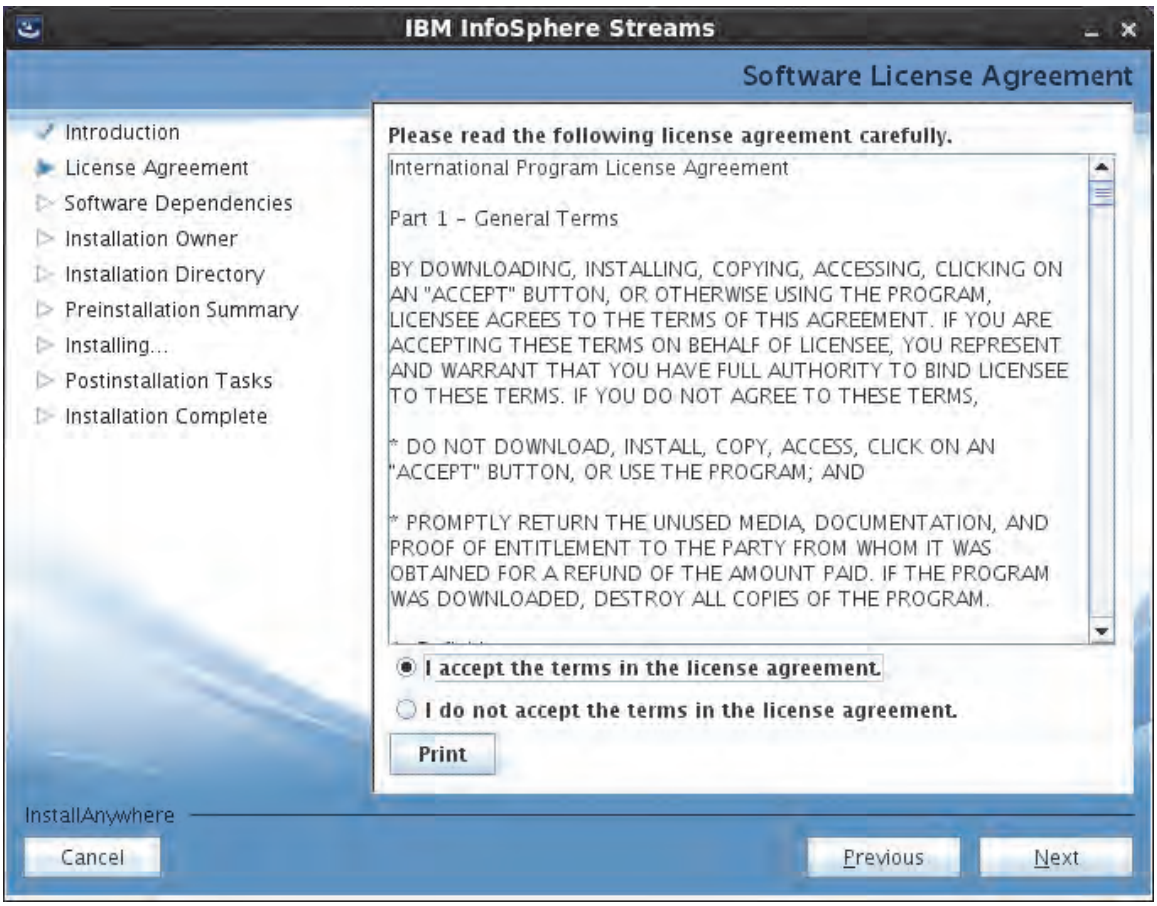
   b. Run the installer with the following command:

   ```
   ./InfoSphereStreamsSetup.bin
   ```

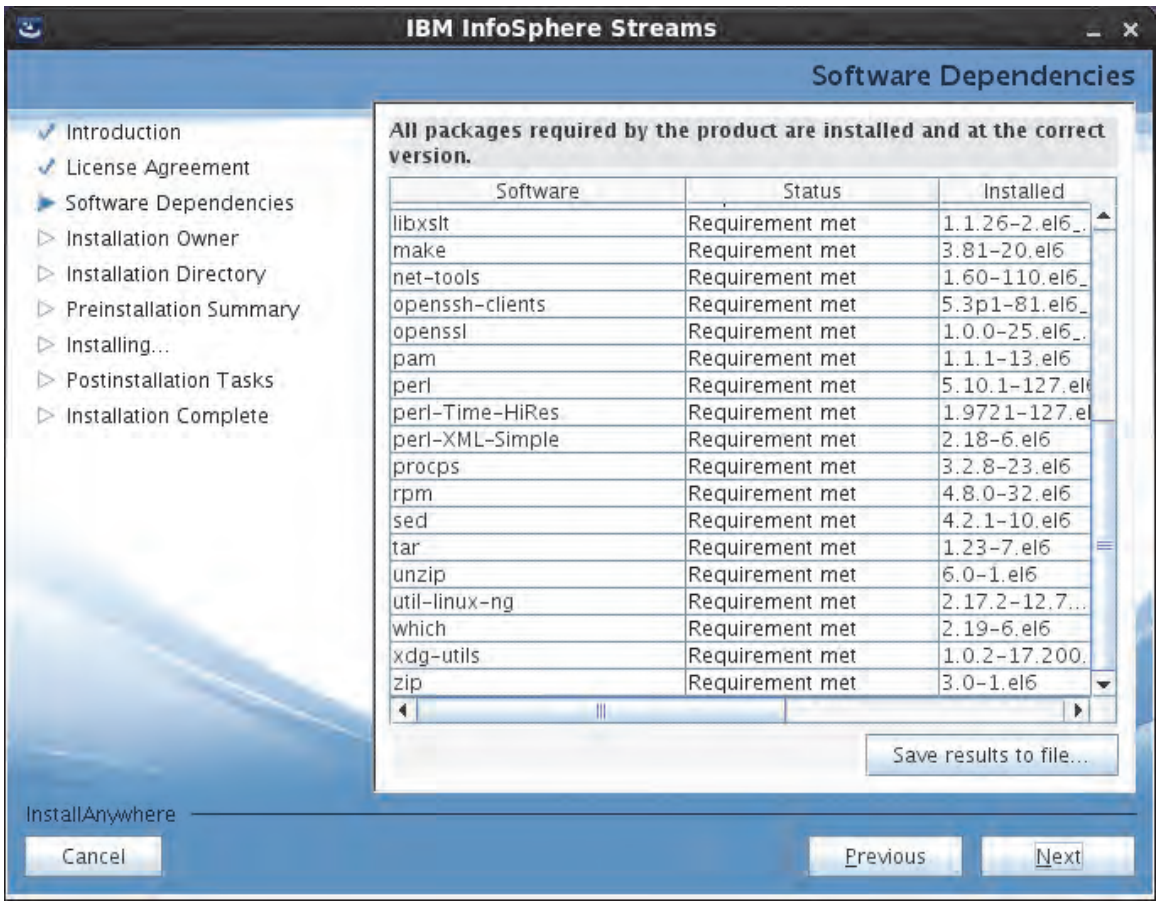
5. Select **English** and click OK.

6. On the Introduction page, click **Next**.

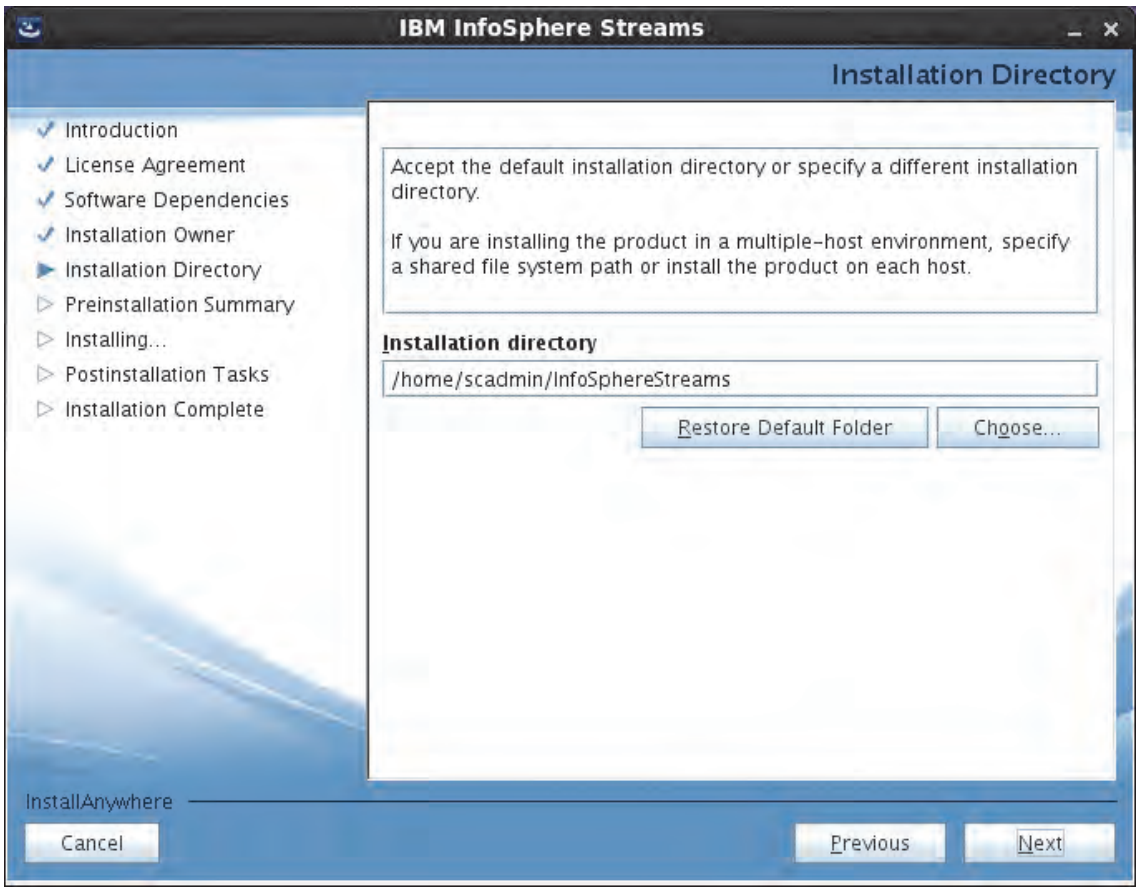
7. Accept the license agreement, and click **Next**.



8. Check that requirements are met. Click **Next**.

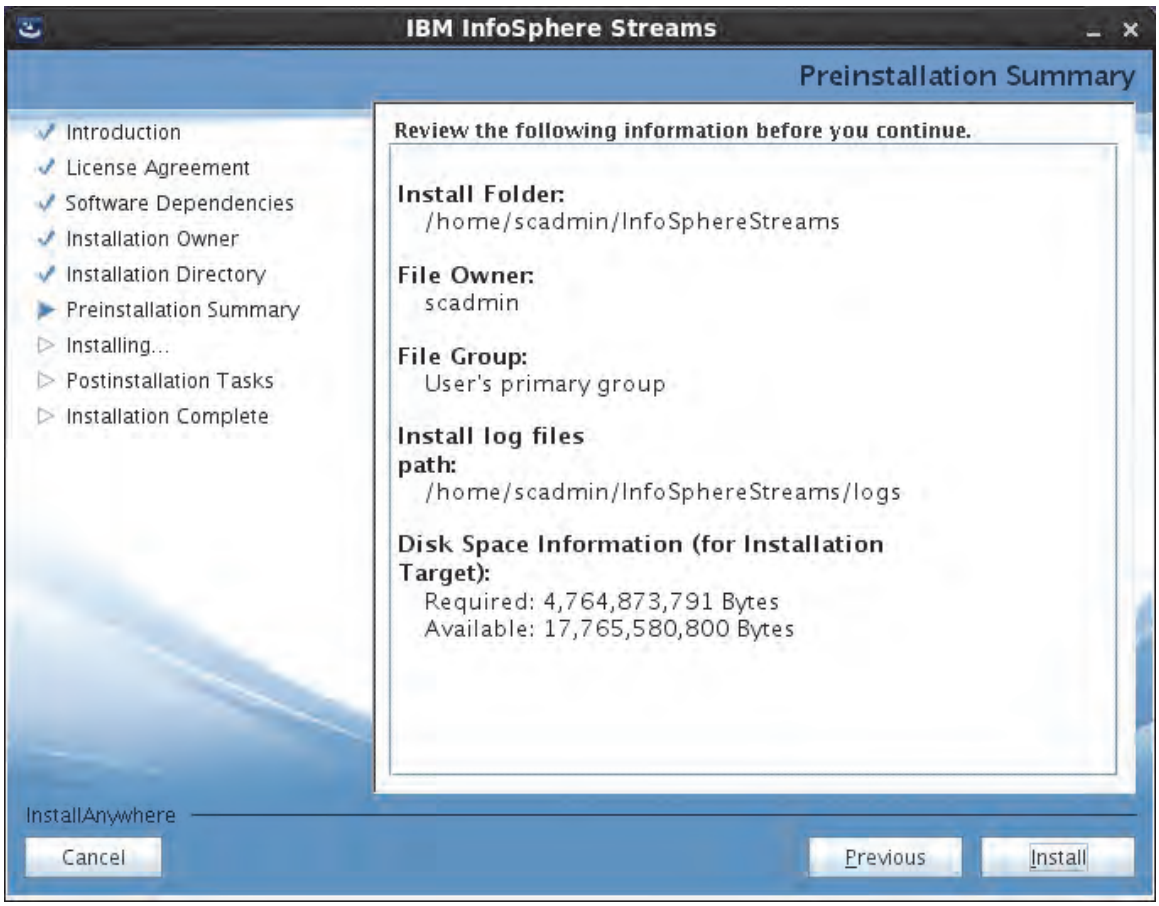


9. Accept the default installation directory. Click **Next**.

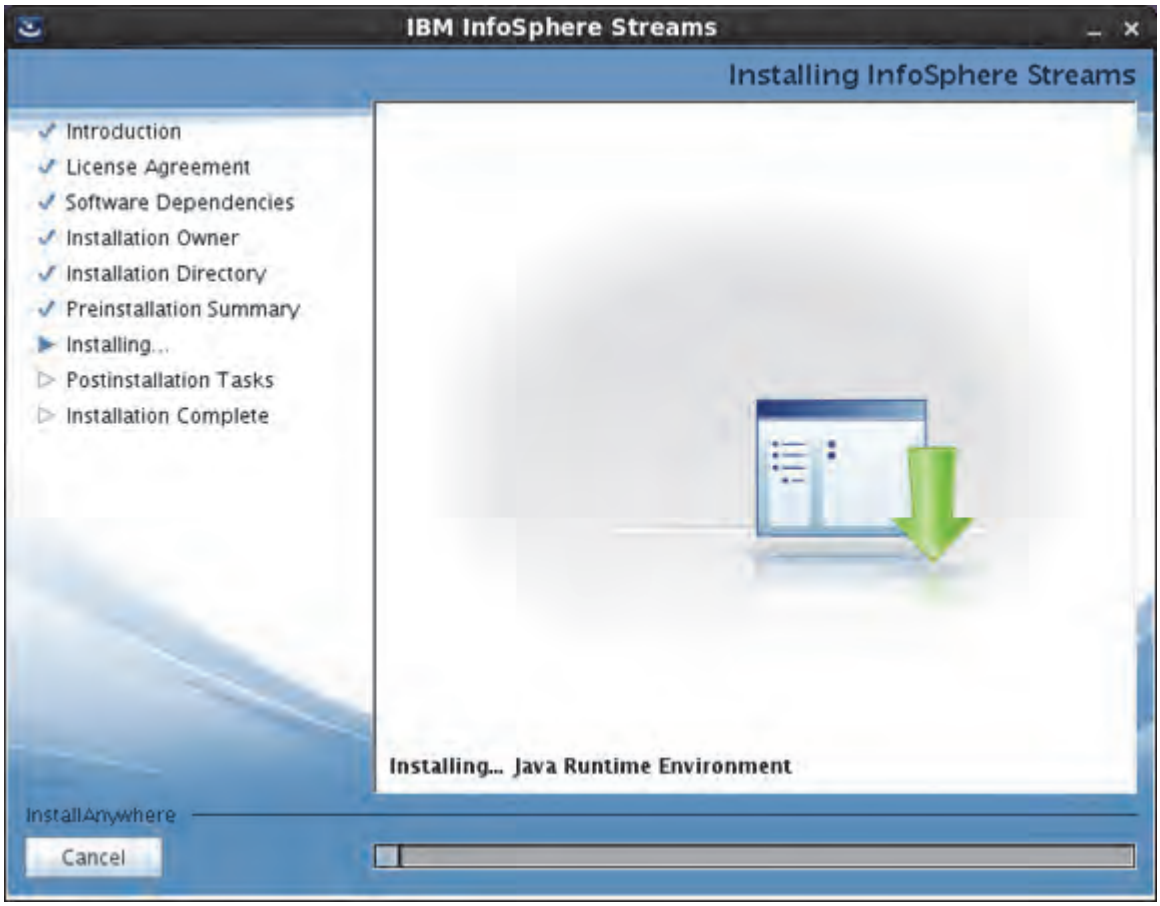


Note: In a production installation, it might be more proper to install the software into the /opt/IBM directory.

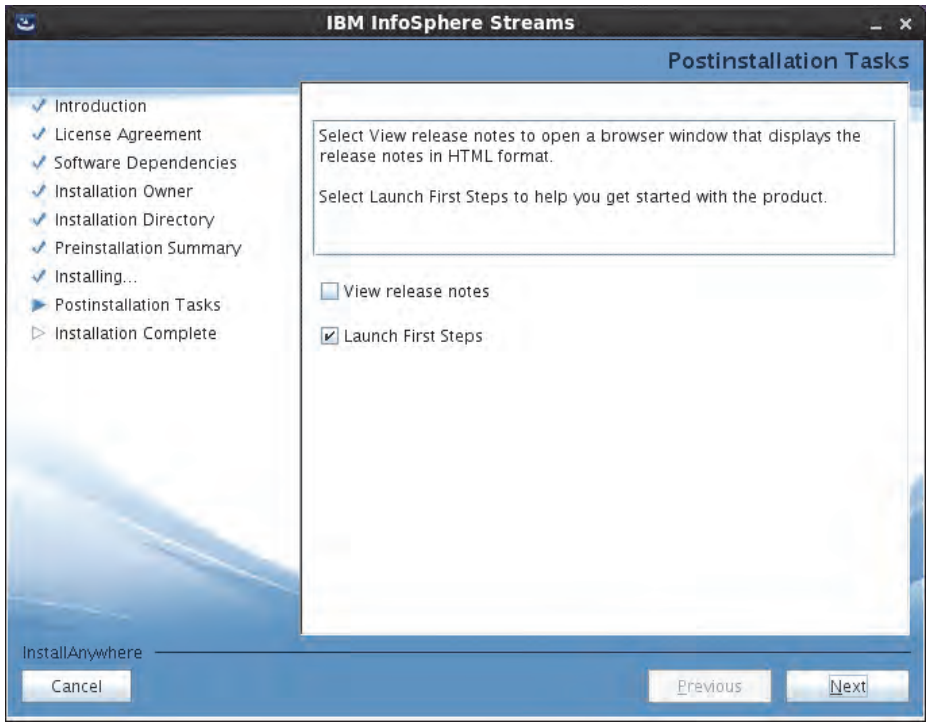
10. Review the preinstallation summary. Click **Install**.



The installation begins. It should take about 5 minutes to complete.



11. Complete the post-installation tasks.
    a. Ensure that **Launch First Steps** is selected, and click **Next**. In a few moments, another window appears.

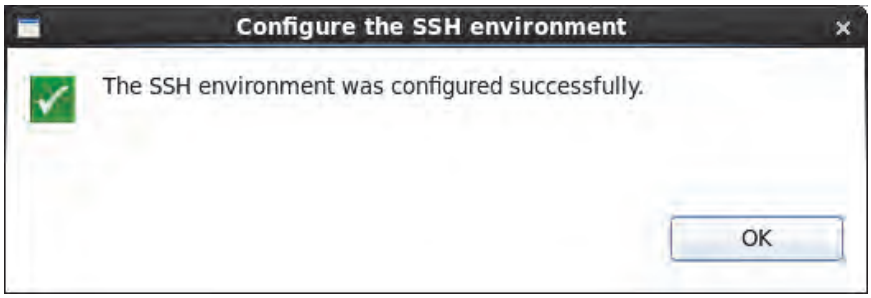


b. In the First Steps window, click **Configure the SSH Environment**.

c. This VM has been used for previous installations and the SSH environment has already been configured.

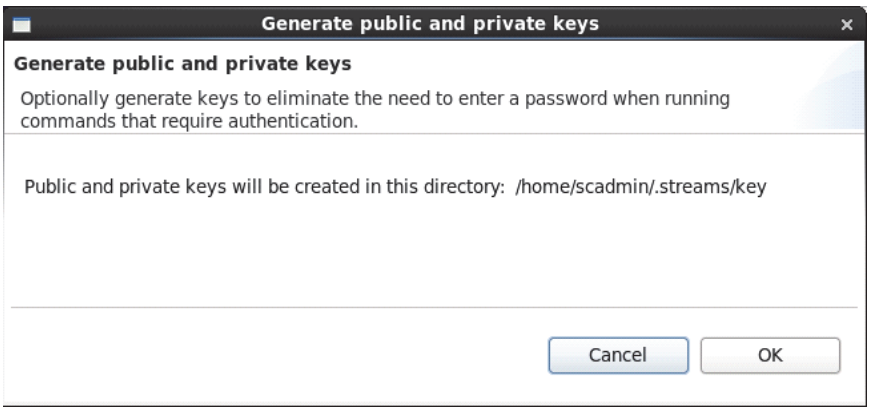
Note: When installing the first time on a Red Hat server, the following steps are required. Select the SSH key type, for example, DSA. Click OK.

Click OK after the SSH environment is configured.



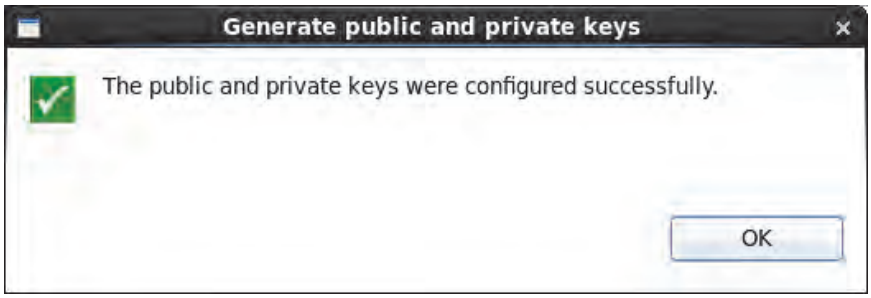
d. Click **Generate public and private keys**.

e. Click **OK** in the Generate public and private keys window.



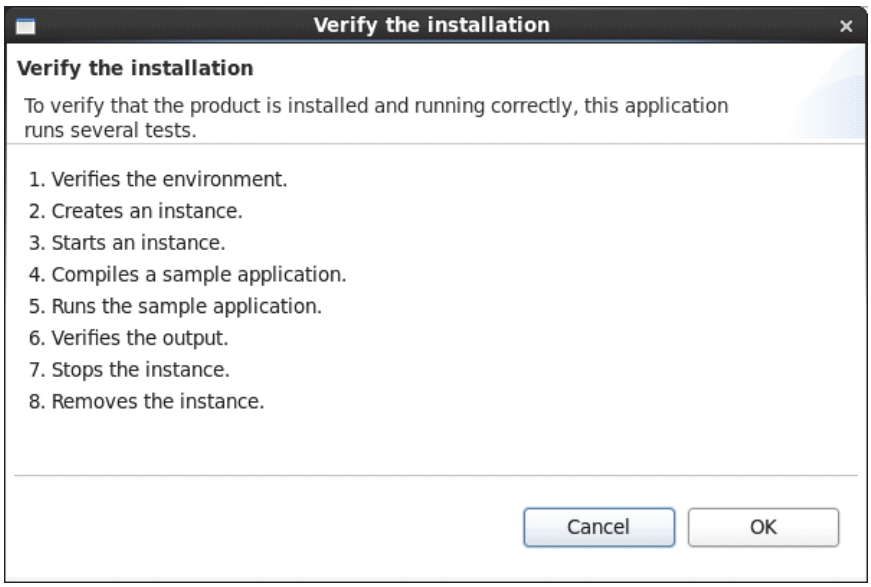
The keys are created and verified.

f. Click **OK** after the keys are created.

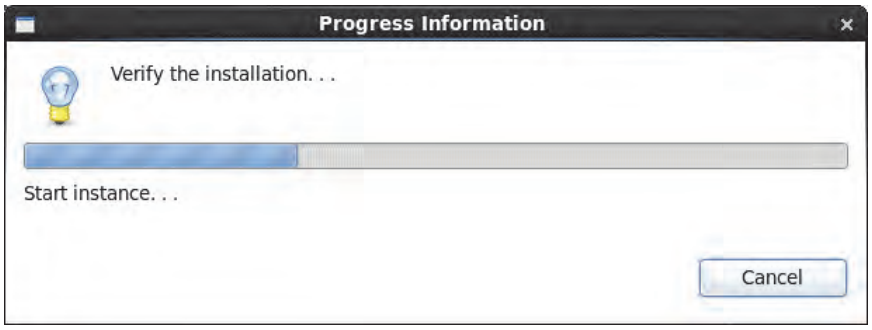


g. Click **Verify the installation**.

h. Click **OK** to verify the installation.

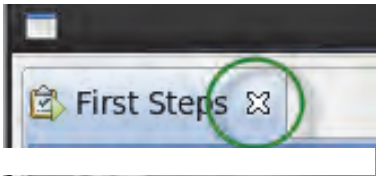


The verification process begins.



i. Click **OK** after a successful verification.

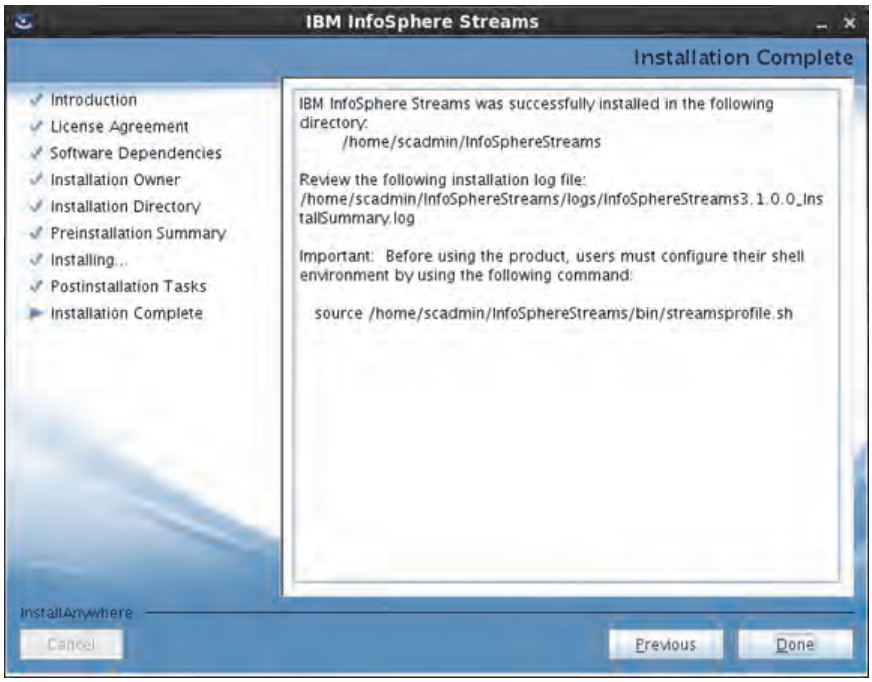
j. Click the **X** to close the First Steps window and exit the installer.



Note the command to start first steps in the future, although restarting it is rare.

k. Click **OK**.

l. Click **Done** to exit the installer.



12. Set the JAVA_HOME and PATH for the version of Java that is used by Streams. Ensure that scadmin uses the Streams profile. Add the following commands to the bashrc file for scadmin:

```
. /home/scadmin/InfoSphereStreams/bin/streamsprofile.sh
export JAVA_HOME=/home/scadmin/InfoSphereStreams/java
export PATH=/home/scadmin/InfoSphereStreams/java/bin:$PATH
```

Source the new bashrc file for the open terminal.

```
. ~/.bashrc
```

The streams installation is complete.

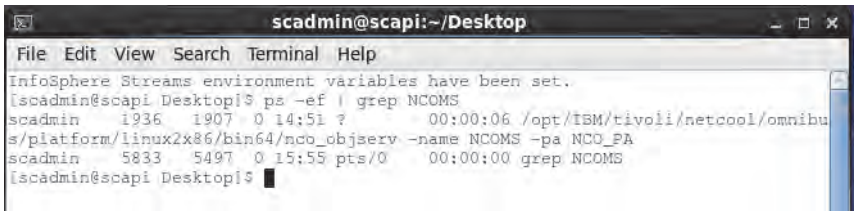
Exercise 2. Confirming important processes are running

It is important to ensure the following processes were started during the boot of the virtual machine and are up and running. If any of these processes are not functional, then you shall have problems with installation and later exercises in this lab.

1. Check that OMNIbus is running.
   a. In a terminal window, enter the following command:

      ```
      ps -ef | grep NCOMS
      ```

   b. If the following window opens, the process is running. Skip the following steps.



   c. If the process is not running, enter the following command:

      ```
      rm /opt/IBM/tivoli/netcool/omnibus/var/NCOMS.pid
      ```

   d. Wait a minute or two and check to see if OMNIbus has started by using the following command:

      ```
      ps -ef | grep NCOMS
      ```

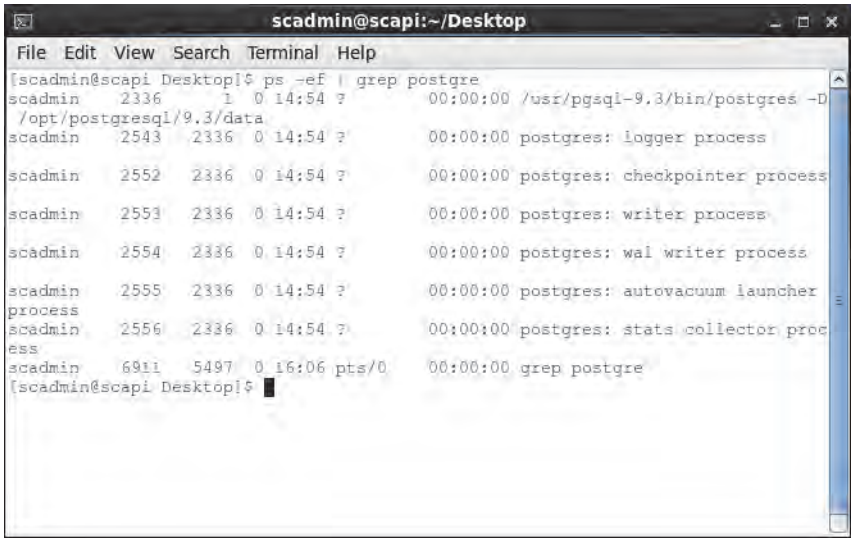
   e. Repeat process check until you get a response that the process is running.

Note: When OMNIbus does not start at boot, it usually means that a process identifier (PID) file is sitting around. This file can be left from either a previous shutdown that did not delete it or an unsuccessful start. The OMNIbus process automation daemon attempts to restart OMNIbus but cannot since it sees the existence of the PID file. Deleting the PID file when there the NCOMS process is gone allows the daemon to start OMNIbus. You will not be able to install Predictive Insights if OMNIbus is not running.

2. Using a similar process as with OMNIbus, check that PostgreSQL is running.
   a. In a terminal window, enter the following command

      ```
      ps -ef | grep postgre
      ```

b. If the following window opens, the process is running. Skip the following steps.



c. If the process is not running, enter the following command:

```
rm /opt/postgresql/9.3/data/postmaster.pid
```

d. Start the PostgreSQL database with the following command:

```
/usr/pgsql-9.3/bin/pg_ctl start -D /opt/postgresql/9.3/data
```

Like with OMNIbus, the PostgreSQL process checks for the existence of a PID file before it allows itself to start. If that file exists because of improper shutdown, PostgreSQL does not start.

Note: PostgreSQL is not needed for the installation process. It is used later in exercises that show you how to connect to a non-DB2 database that has data you wish to mediate.

Exercise 3 Installing Predictive Insights

In this exercise, you use the following steps to install the components for running Predictive Insights:

- Add the appropriate schema to the DB2 instance.
- Configure the Tivoli Integrated Portal that is used by Web GUI with the Predictive Insights applets.
- Install the data mediation client to create data models.
- Install the analytics software that works with InfoSphere Streams to analyze and monitor your system.

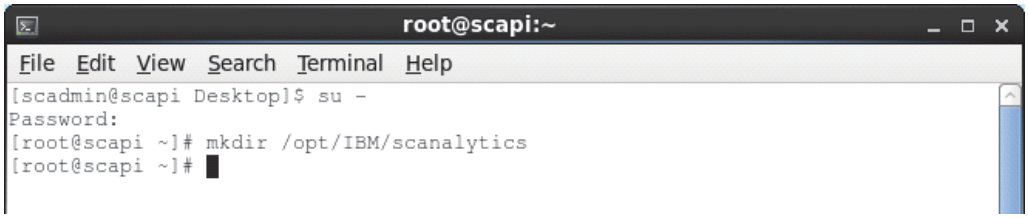
Complete the following steps:

1. Add the installation directory **scanalytics** that **scadmin** owns in the **/opt/IBM** directory. As the **root** user, create the **/opt/IBM/scanalytics** directory.
   a. If a terminal is not already open, then open one and become the **root** user with the password **object00.**

      ```
      su -
      ```

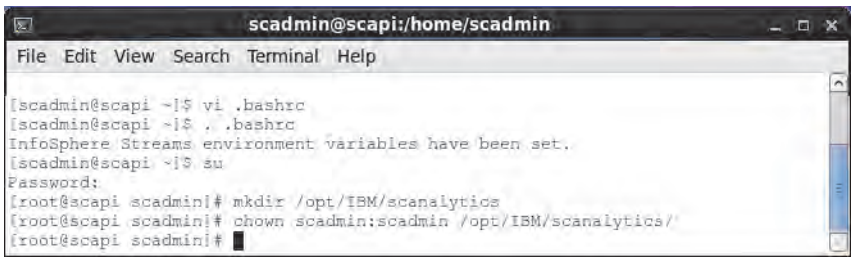
   b. Make a directory named **/opt/IBM/scanalytics**.

      ```
      mkdir /opt/IBM/scanalytics
      ```



2. Change the ownership of **/opt/IBM/scanalytics** to the user **scadmin** and the group **scadmin**.

   ```
   chown scadmin:scadmin /opt/IBM/scanalytics
   ```



3. Exit **root**.

Important: Be sure to do the remaining steps in the installation as the **scadmin** user and that the most recent changes to **bashrc** have been sourced to the terminal window you are in.

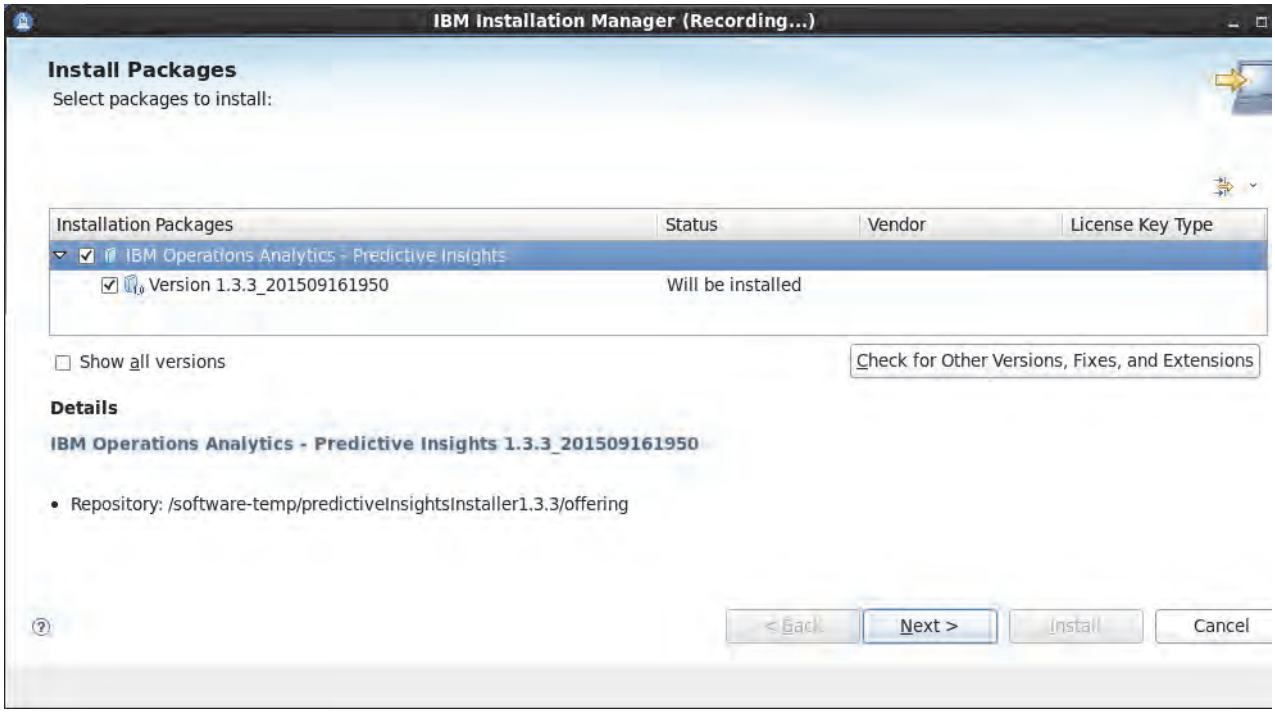
4. As the **scadmin** user, copy and extract the **IOA_PI_1.3.3.tgz** file from **/scapi-install-files** to **/software_temp**.

   ```
   cp /scapi-install-files/IOA_PI_1.3.3.tgz /software-temp/.
   cd /software-temp
   tar -xzf IOA_PI_1.3.3.tgz
   ```

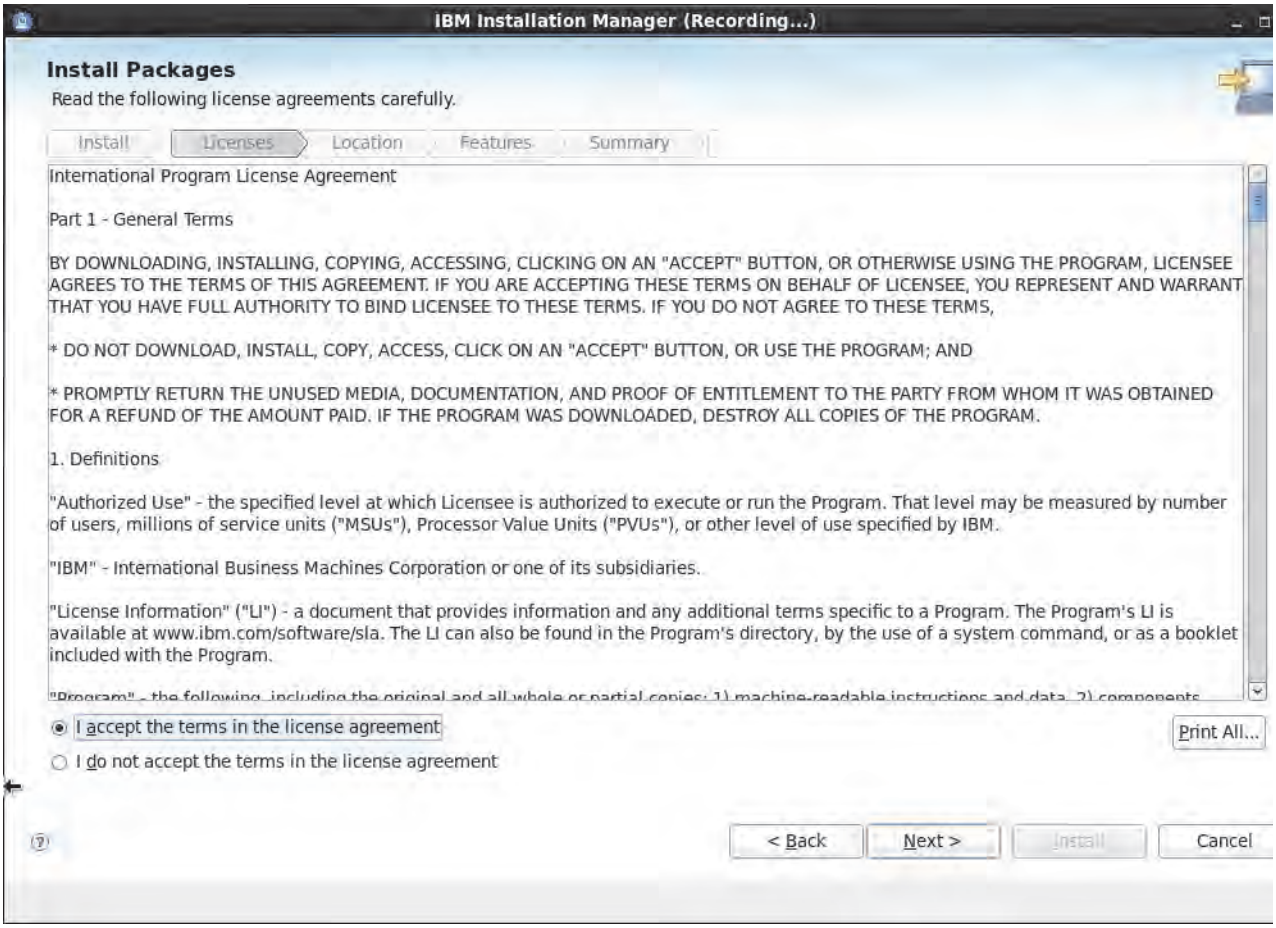
5. Start and configure the Predictive Insights installer.
   a. Start the **predictiveInsightsInstaller1.3/install.sh** file. Move into the installation directory and start the installer. Click the Install icon.

      ```
      cd /software-temp/predictiveInsightsInstaller1.3
      ./install.sh
      ```

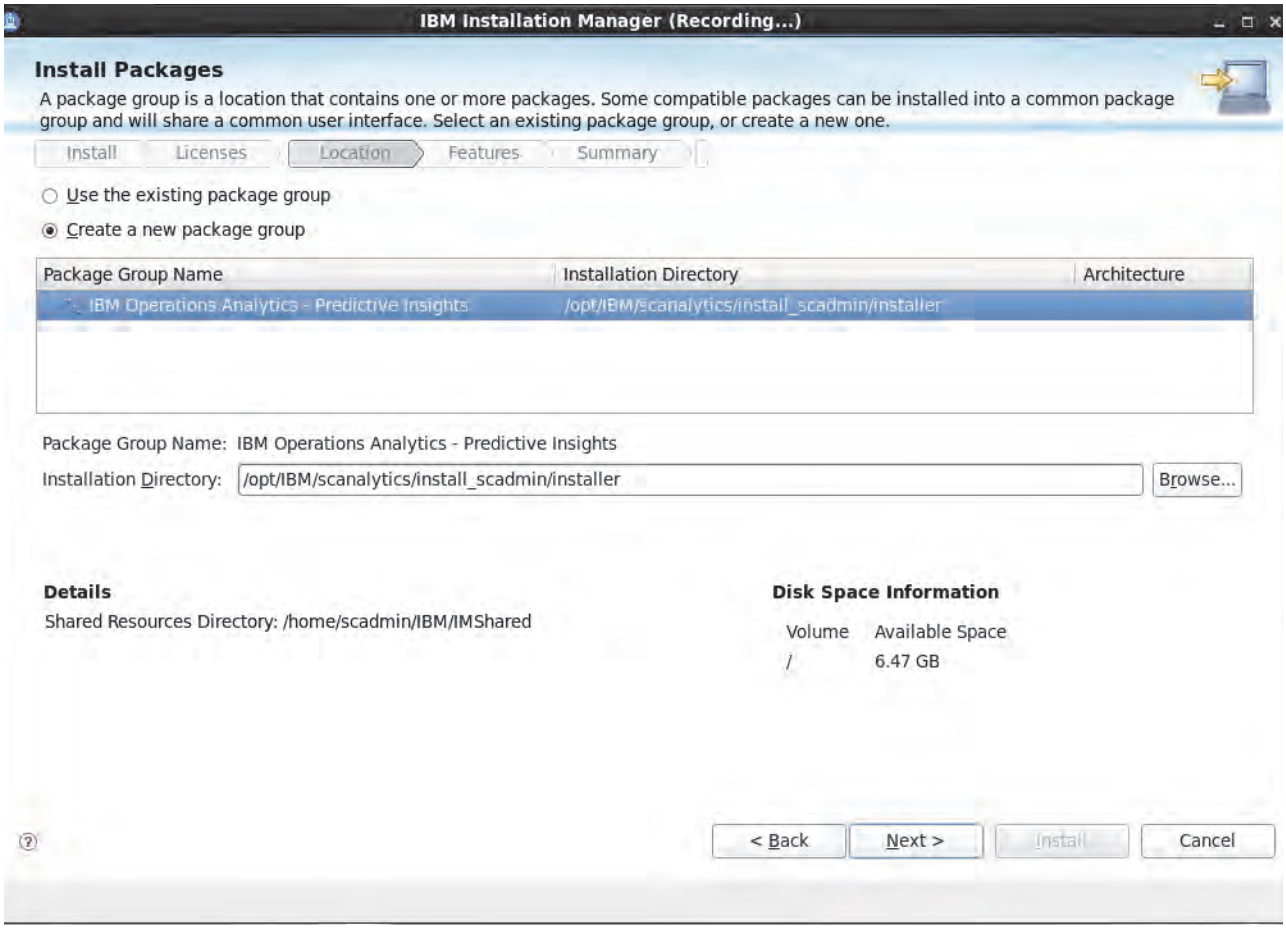
b. Select all the packages and click **Next** on the Install Packages page.



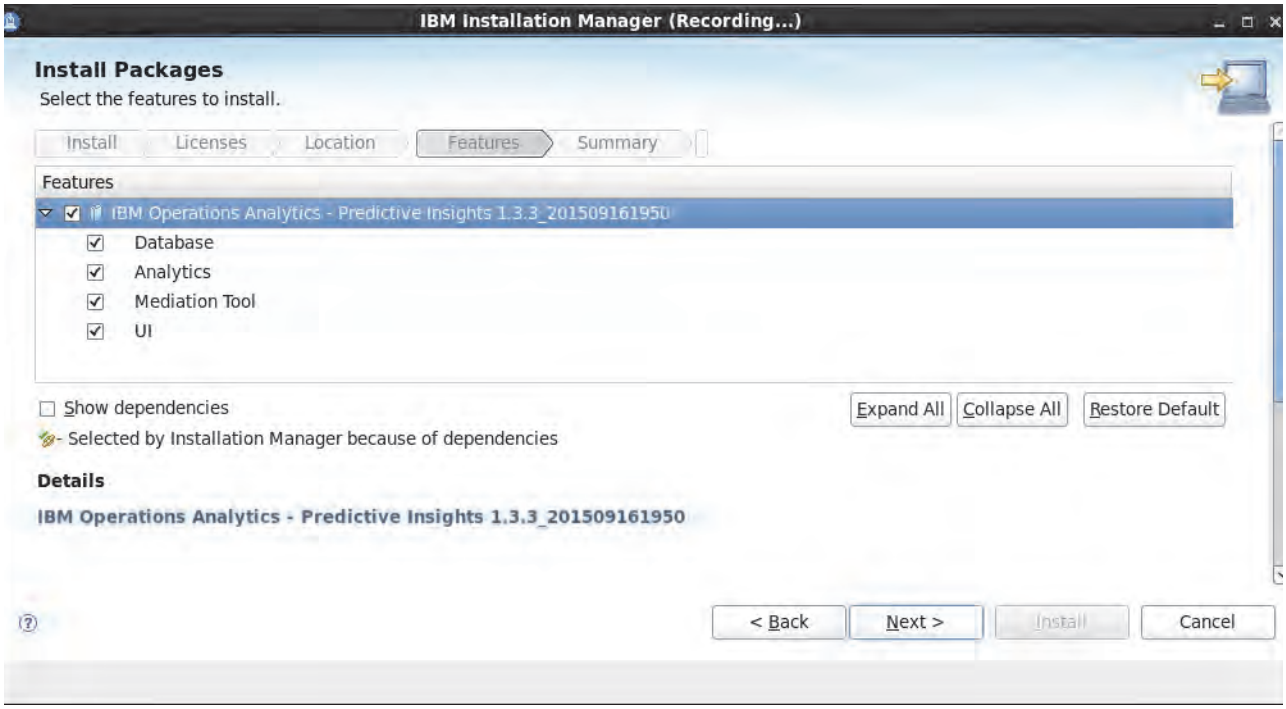
c. Accept the license agreement and click **Next**.



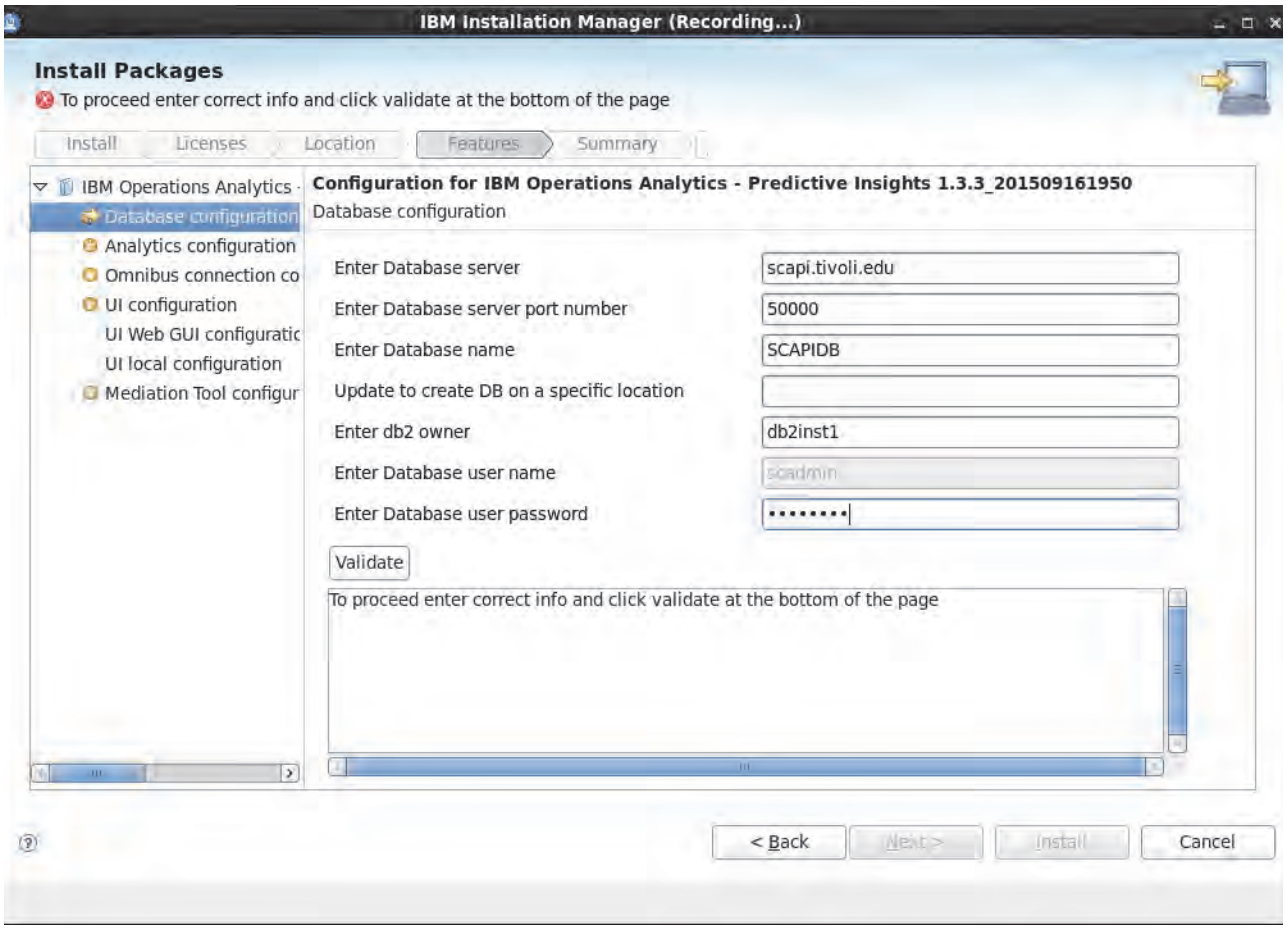
d. Accept the default installation directory and shared resources directory. Click **Next**.



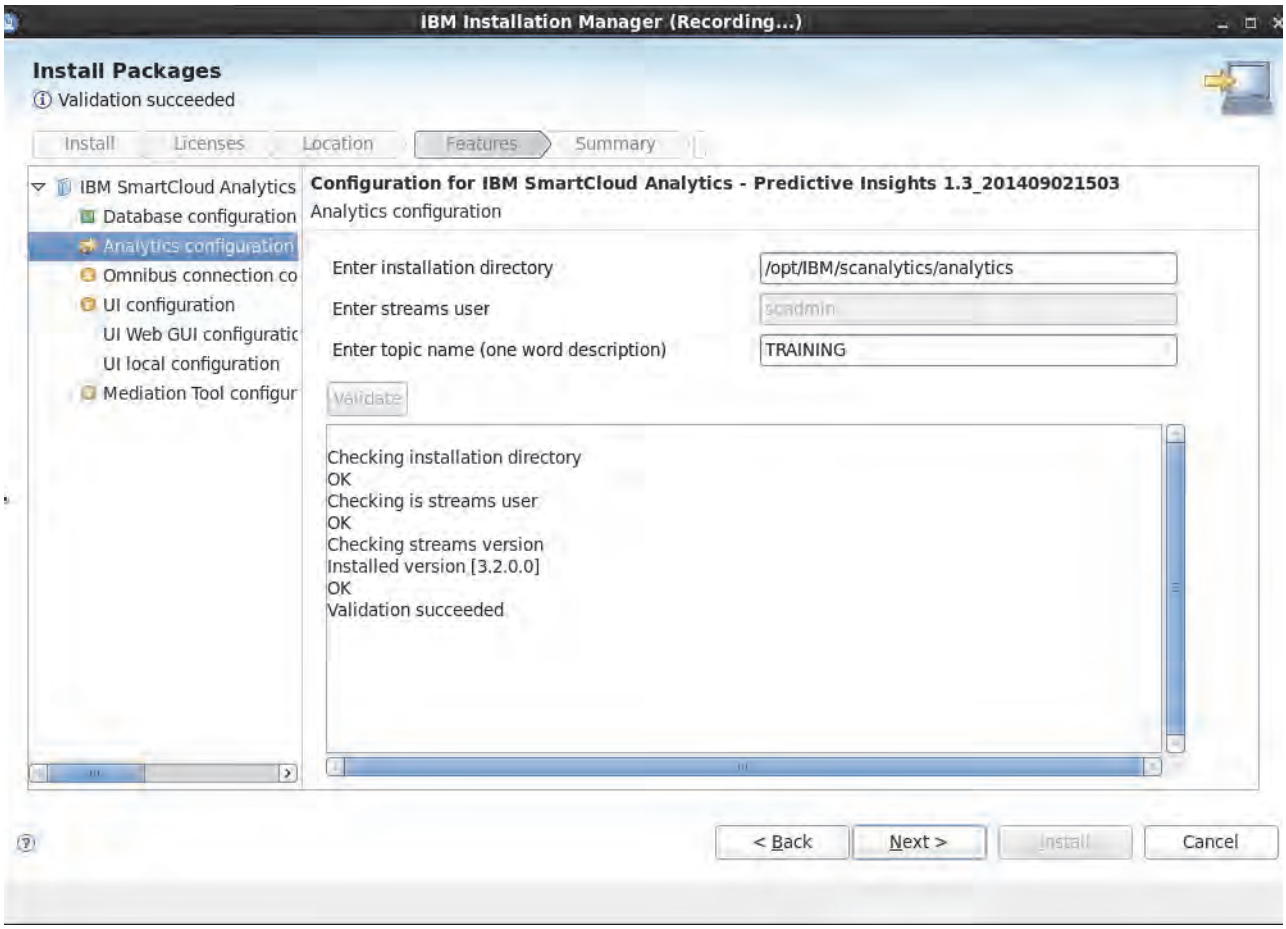
e. Install all the selected components, and click **Next**.



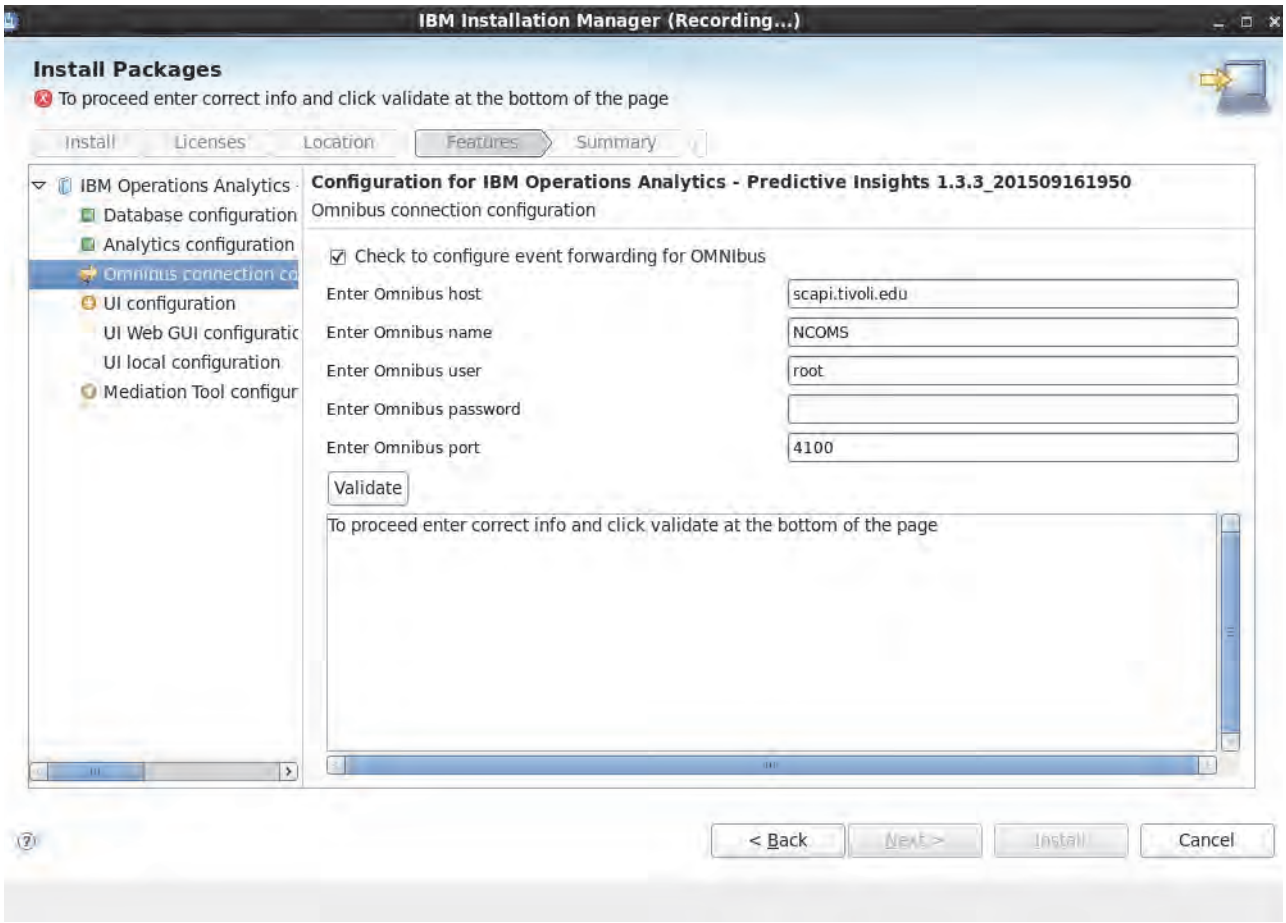
f. Ensure that the db2 owner is **db2inst1** and enter the password **object00**. Click **Validate** to validate the database details. Click **Next** after successful validation.



g. Use the default installation directory. Enter the topic name **TRAINING**. Click **Validate** to validate the details. Click **Next** after successful validation.

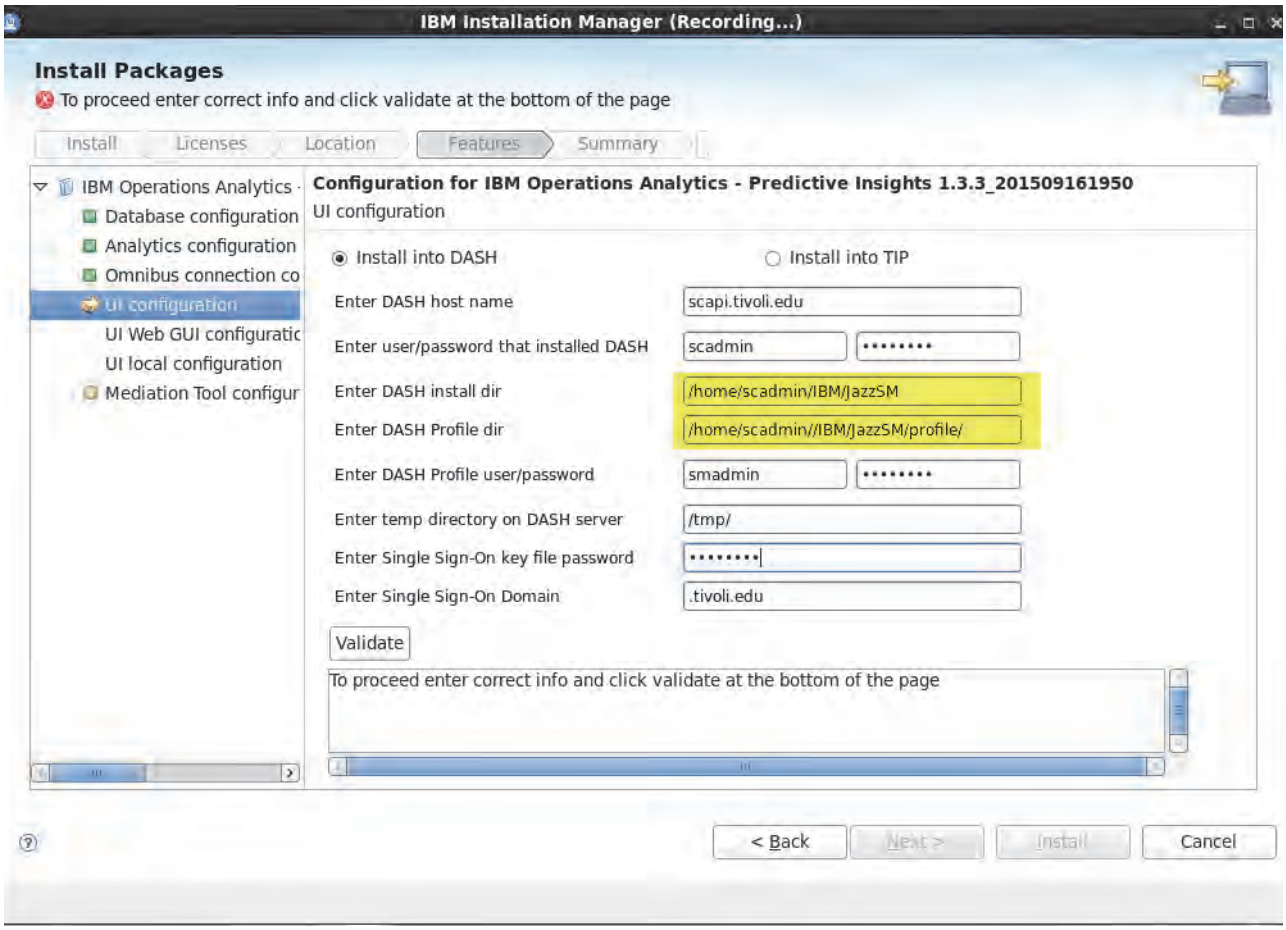


h. Confirm that the OMNIbus user name is **root**. Enter the password **object00.** Click **Validate** to validate the details. Click **Next** after successful validation.

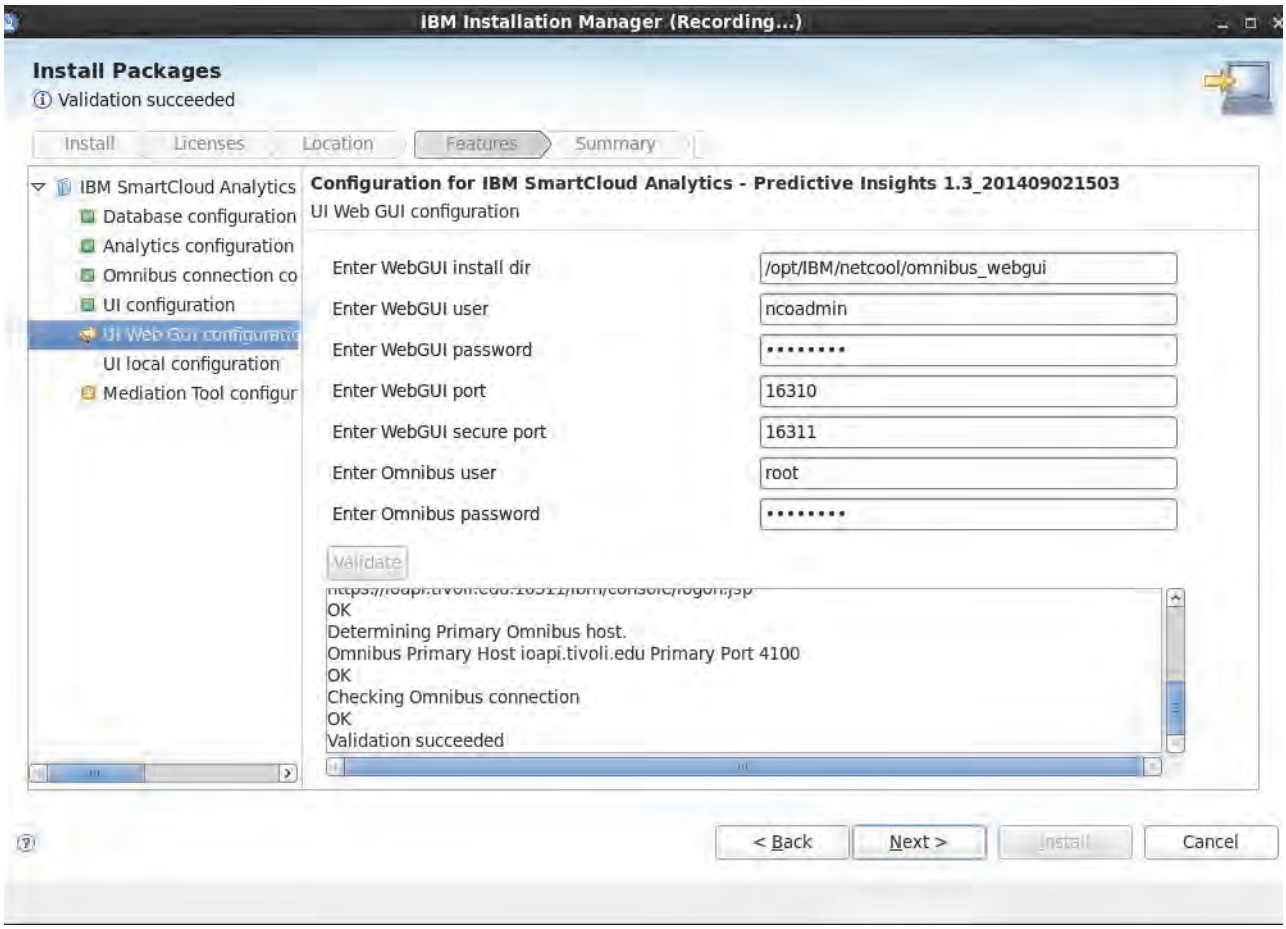


Note: This **root** user is the OMNIbus root user and not the operating system **root** user.

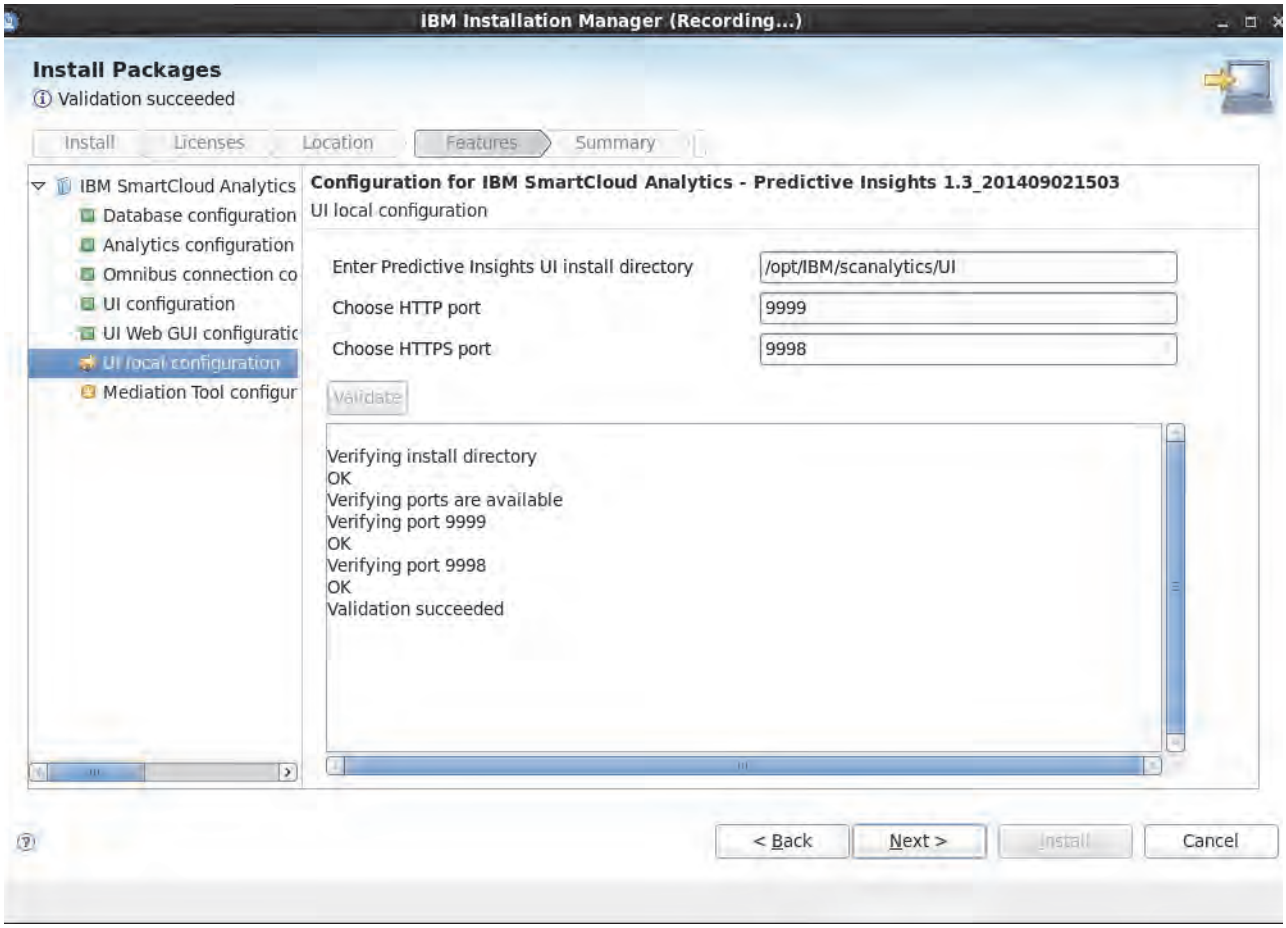
i. Select **Install into DASH**. Enter user name **scadmin** and the password of **object00** for the user that installed DASH. Change the DASH install directory to **/home/scadmin/IBM/JazzSM** and the DASH profile directory to **/home/scadmin/IBM/JazzSM/profile**. Enter the password **object00** for the DASH profile user **smadmin**. Enter the single sign-on password of **object00**. Click **Validate** to validate the details. Validation takes a couple minutes. Click **Next** after successful validation.



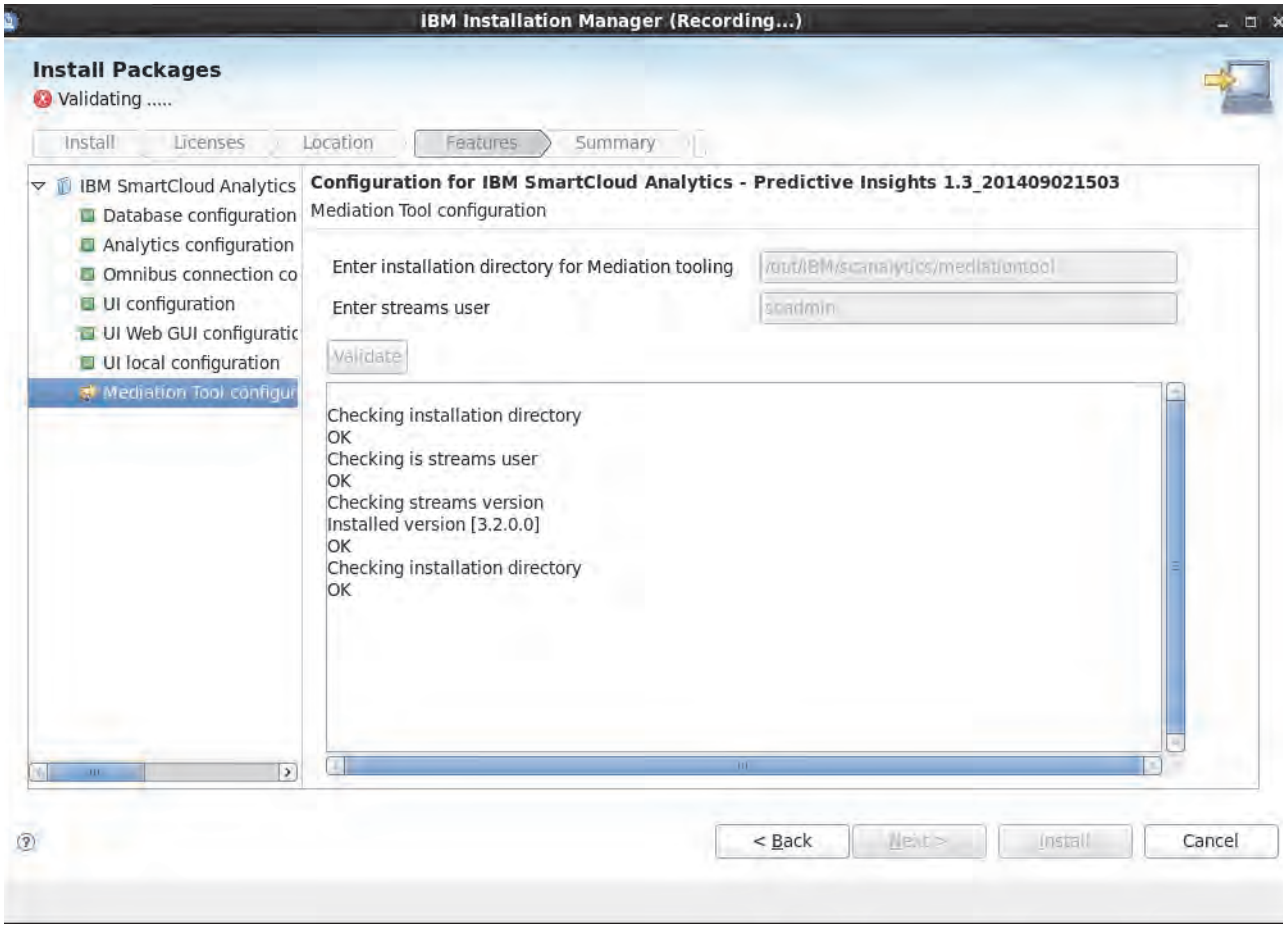
j. Enter the password of **object00** for the **ncoadmin** user. Enter the password of **object00** for the OMNIbus **root** user. Click **Validate** to validate the details. Click **Next** after successful validation.



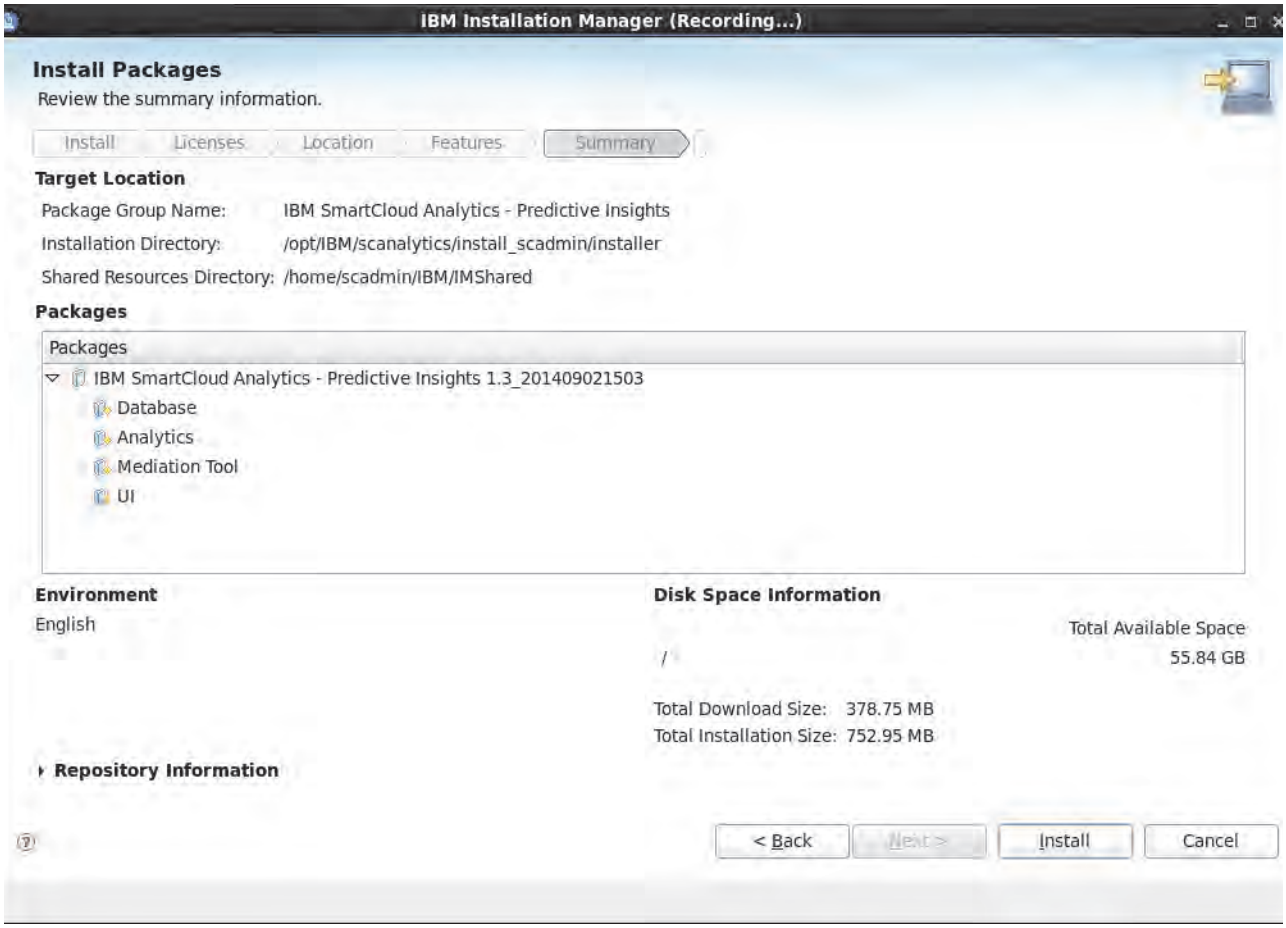
k. Validate that the default ports for the user interface are available. Click **Next**.



l. Validate the **streams** user **scadmin**. Click **Next**.

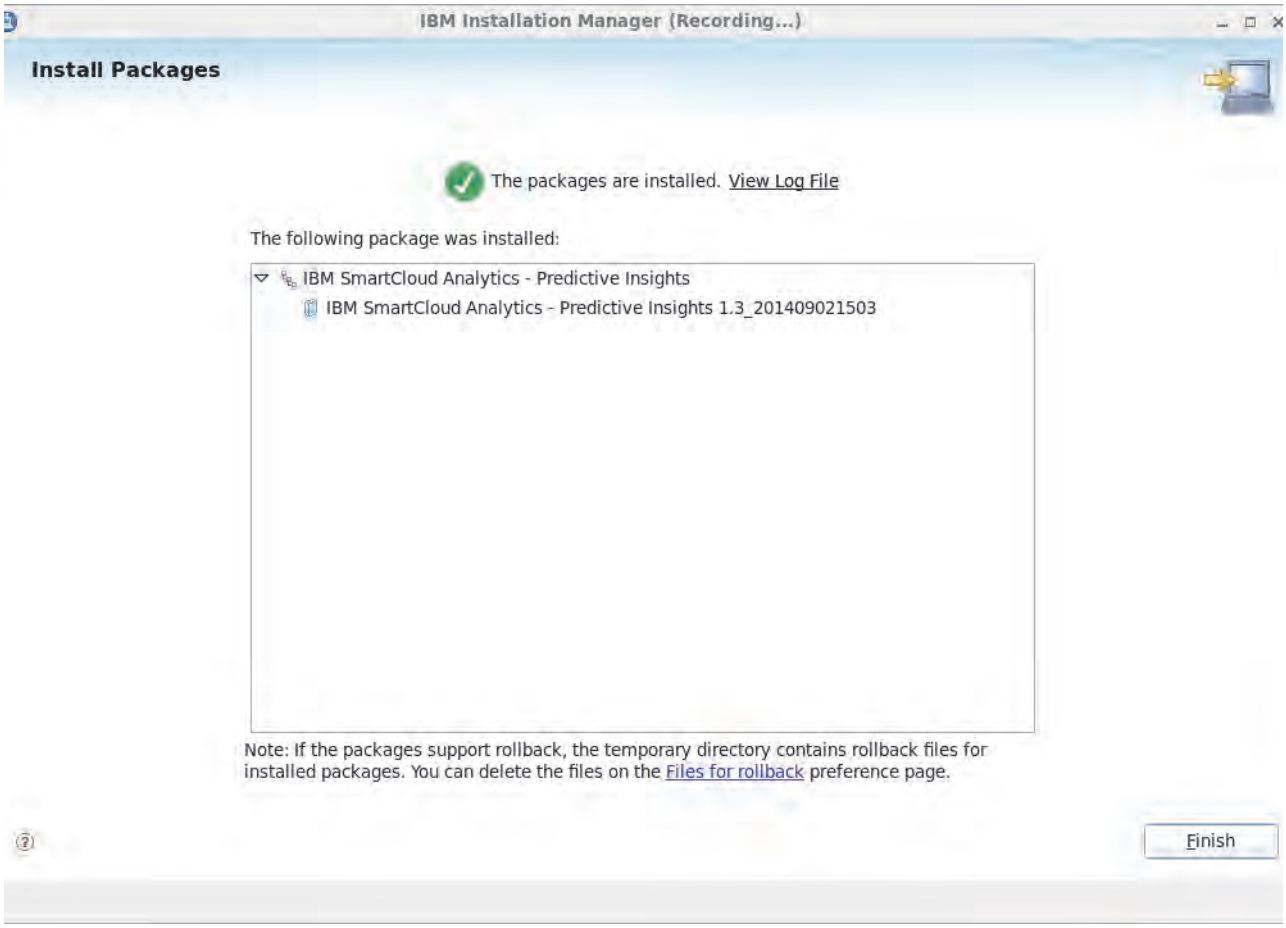


m. Click **Install**.



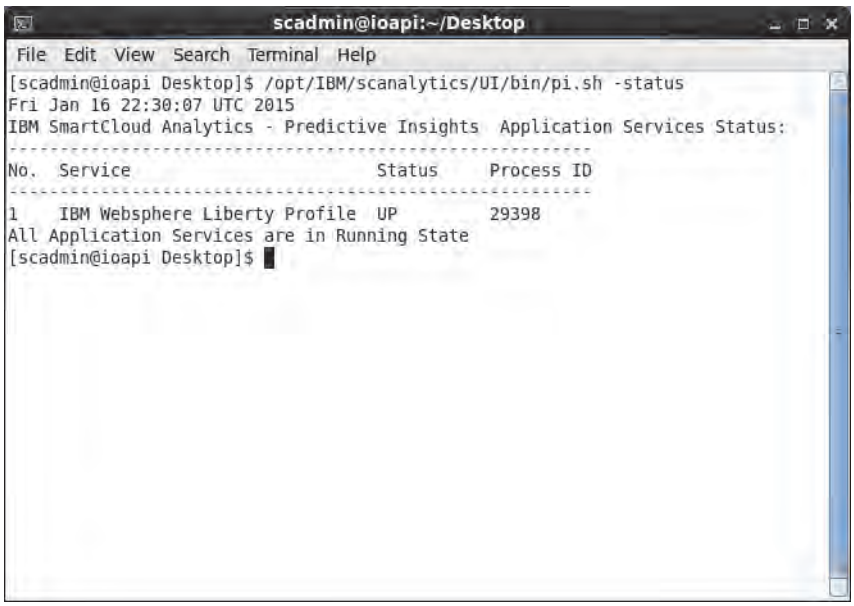
Installation takes approximately 30 - 40 minutes.

n. Click **Finish**. Close the Installer.

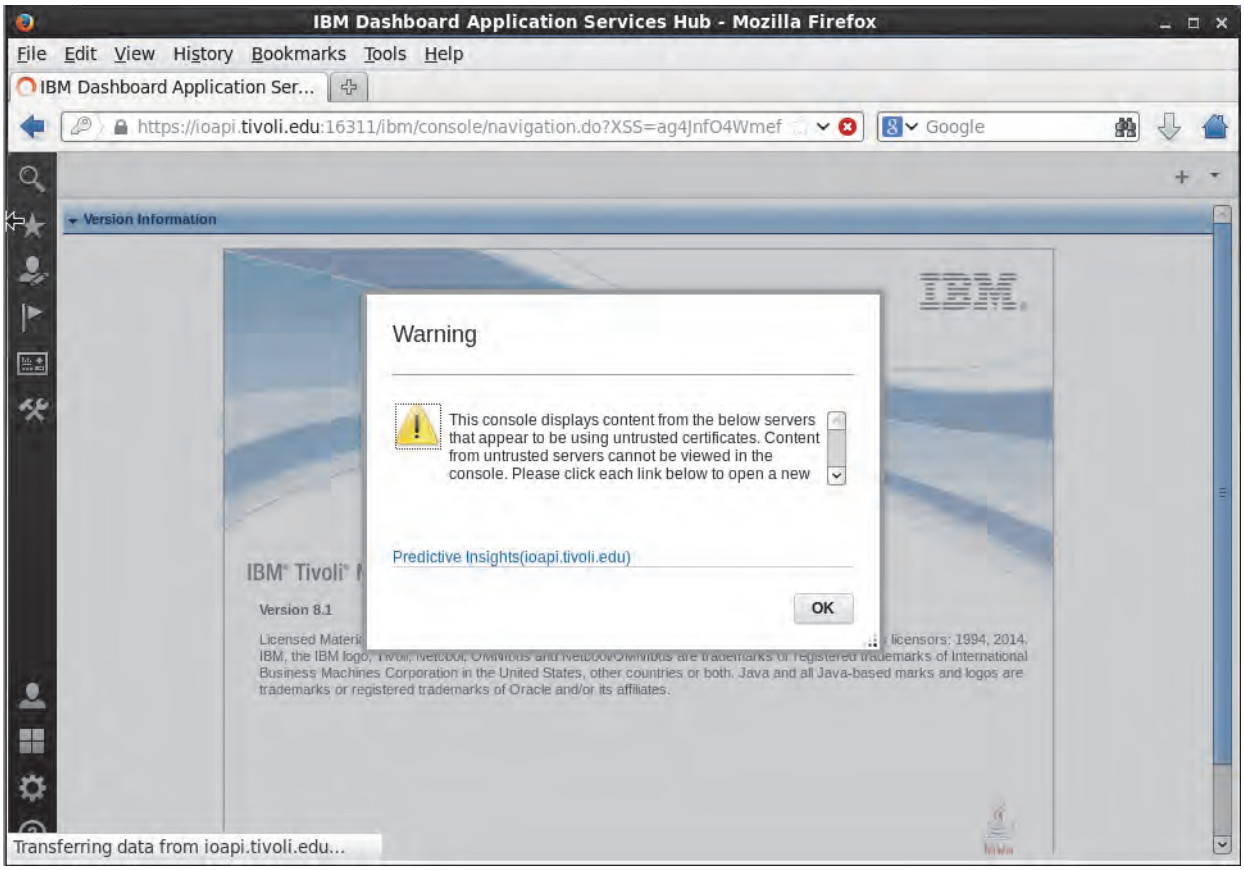


o. Verify that the Predictive Insights user interface web server is up and running. Enter the following command in a terminal window. Make sure that the status is *up*.

```
/opt/IBM/scanalytics/UI/bin/pi.sh -status
```



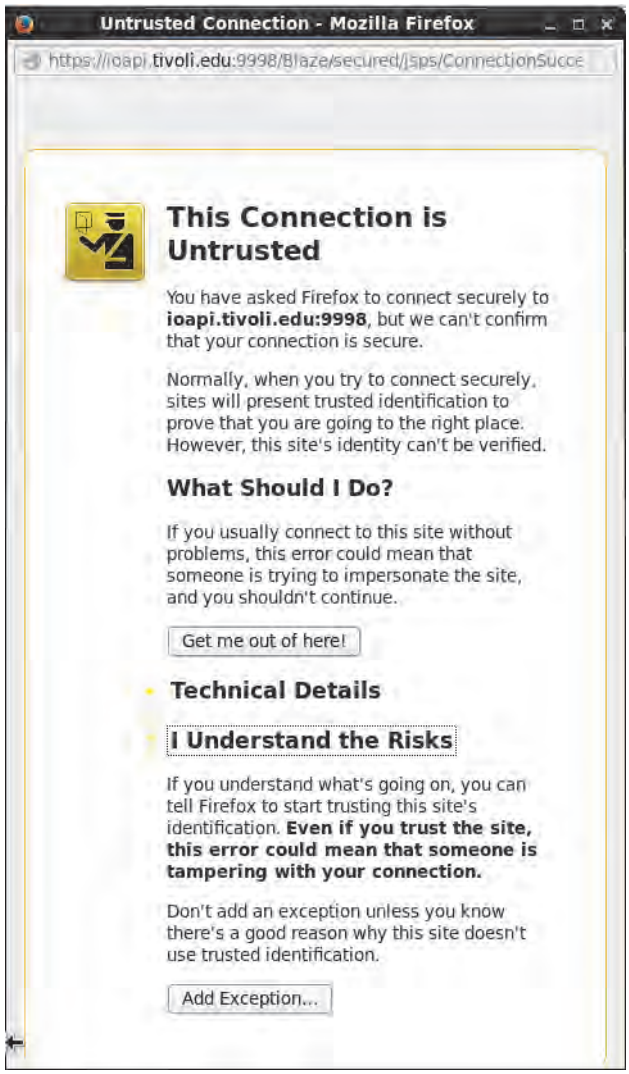
6. Configure the system to allow Predictive Insights into the DASH interface:
   a. If it is not already started, start the Firefox browser.
   b. Point the browser to the DASH URL:
      https://192.168.100.160:16311/ibm/console
   c. Enter the user name **ncoadmin** and password **object00** to log into DASH.
      Upon logging in to the server, you should receive a warning. This warning is generated because the Predictive Insights user interface does not have a signed certificate.
   d. Click the link that says **Predictive Insights (scapi.tivoli.edu)**.



You receive a notice that this is an untrusted connection.

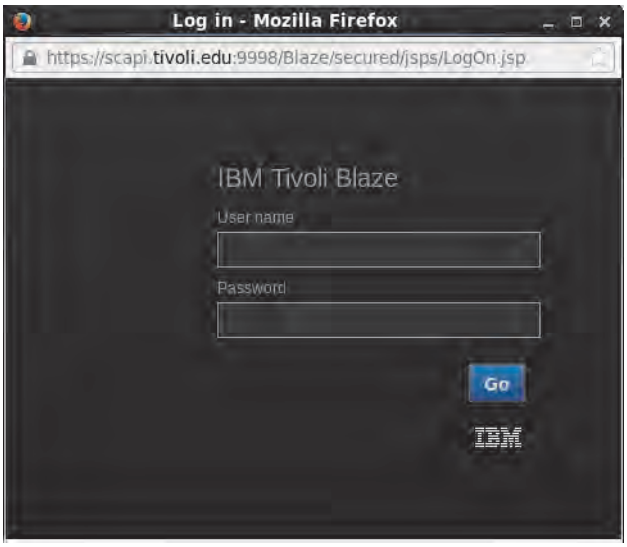
e. Click **I Understand the Risks**.

f. Click **Add Exception**.



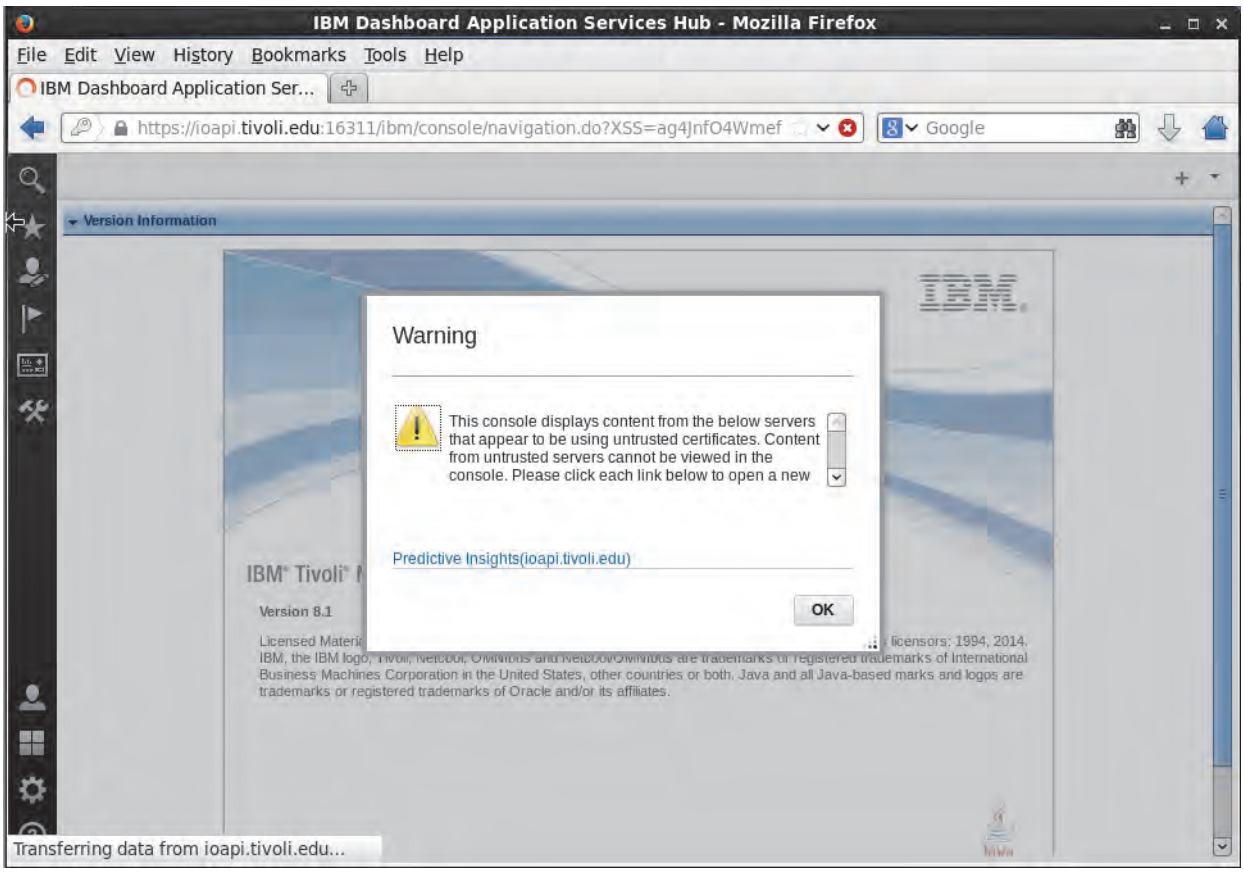
g. Click **Confirm Security Exception**.

h. You are prompted to provide credentials for the Blaze server. Enter the user name **ncoadmin** and the password **object00** and click **Go**.

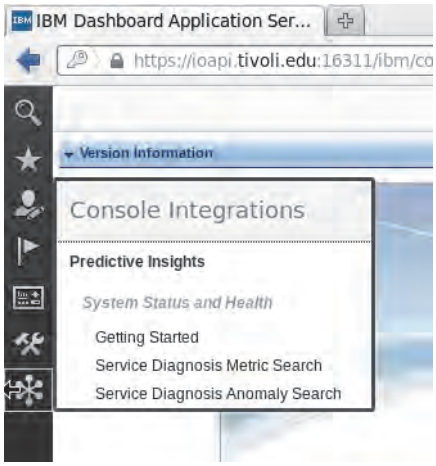


i. Close the browser window.

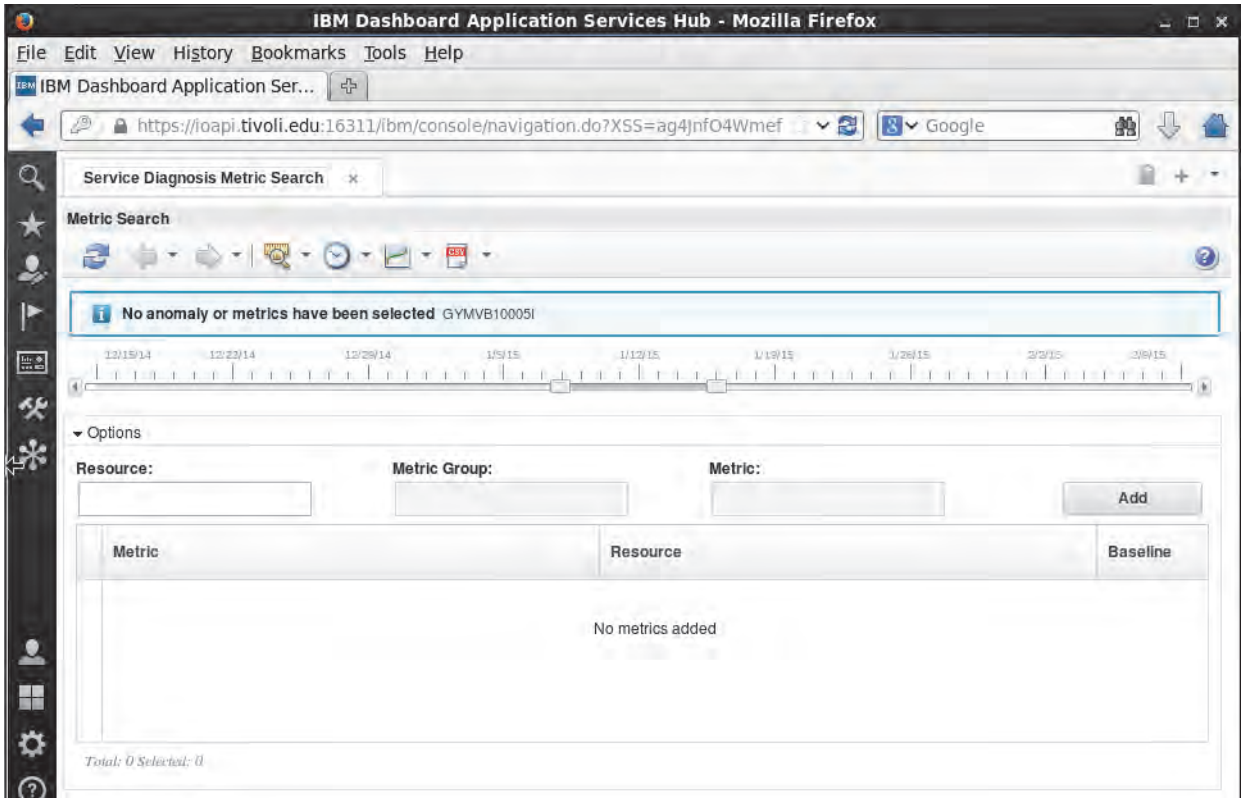
j. Click **OK**. Note the snowflake icon that is added the left menu.



k. Click the snowflake menu and select **Service Diagnosis Metric Search**.



The Predictive Insights Metric Search GUI opens.



The Predictive Insights software is installed.

Unit 4 Data sources and modeling exercises

In the following exercises, you learn about the three data sources that are available in this training. One source is a set of comma-separated value (CSV) files, and the other two are data that is in the DB2 and PostgreSQL databases. You connect to these data sources and model the data within them by using the data mediation tool that you installed earlier. You use the mediation tool to select appropriate KPIs and filter unnecessary resources from the data. You then deploy the model to the Predictive Insights server. As an optional exercise, you configure the mediation client and server to use the data that is in the PostgreSQL database.

These data sources are in one of two formats: wide and skinny. A wide format is where the data schema has individual columns for each KPI being measured, for example:

Time stamp	Resource	KPI-1	KPI-2	KPI-3
20151004_120000	HostA	12	104239	0.001
20151004_120000	HostB	3	203176	0.024
20151004_120000	HostC	22	231765	0.099
20151004_120000	HostD	8	99876	0.231

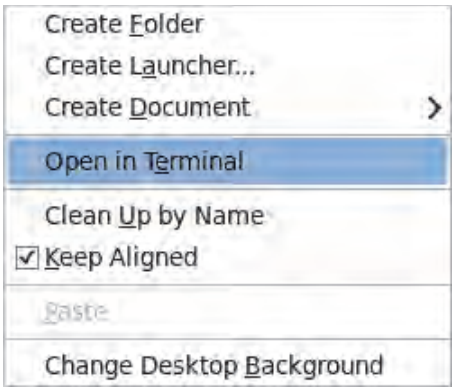
A skinny format is where each row in the table represents both a resource and a KPI value. This schema results in each row representing a different KPI, for example:

Time stamp	Resource	Metric	Value
20151004_120000	HostA	KPI-1	12
20151004_120000	HostA	KPI-2	104239
20151004_120000	HostA	KPI-3	0.001
20151004_120000	HostB	KPI-1	3

Exercise 1 Reviewing data sources

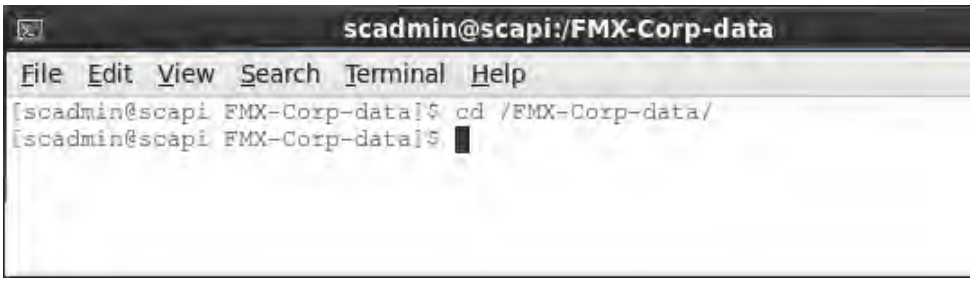
After accessing the desktop, review the historical data that you use in the upcoming exercises. Review the format of the historical files on the server.

1. Review the flat CSV files in the directory /FMX-Corp-data.
   a. Right-click the **scadmin** desktop and select **Open in Terminal.**

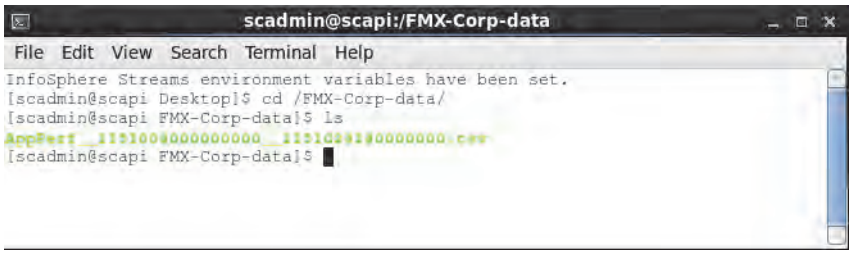


   b. Browse to the directory /FMX-Corp-data.

      `cd /FMX-Corp-data`



   c. List the files in this directory with the **ls** command.



      Note that the file format follows a *<table-name>__<start-timestamp>__<end-timestamp>.csv* naming convention. Scroll up in window and notice the following format for the files:

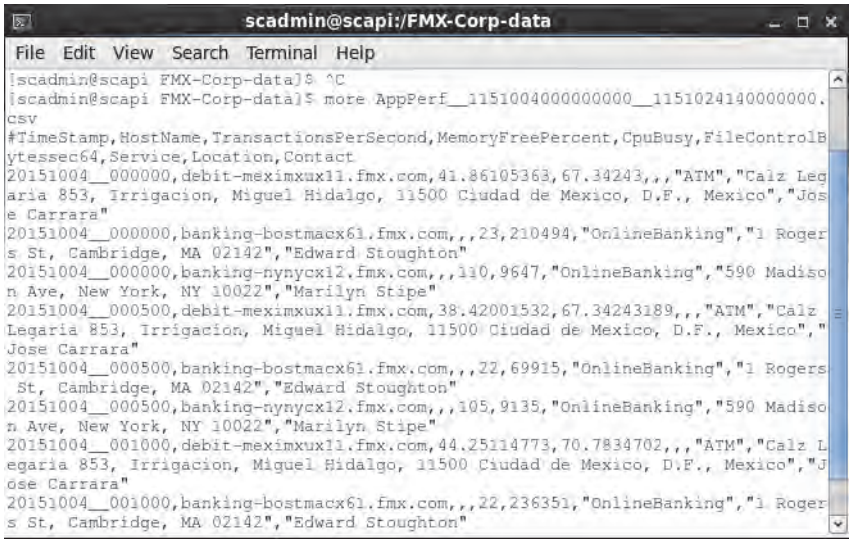
      AppPerf__*<timestamp>*__*<timestamp>*.csv

      The file spans from midnight 04 Oct 2015 to 2:10 pm of 24 Oct 2015 based on the format of the time stamps that align with 1YYmmDDhhMMssSSS.

AppPerf__1151004000000000__1151024140000000.csv

d. Review the data inside the **RespTime** file. Note the layout of the CSV file. Review the contents of RespTime__1150516000000000__1150601121000000.csv with the **more** command.

```
more AppPerf__1151004000000000__1151024140000000.csv
```

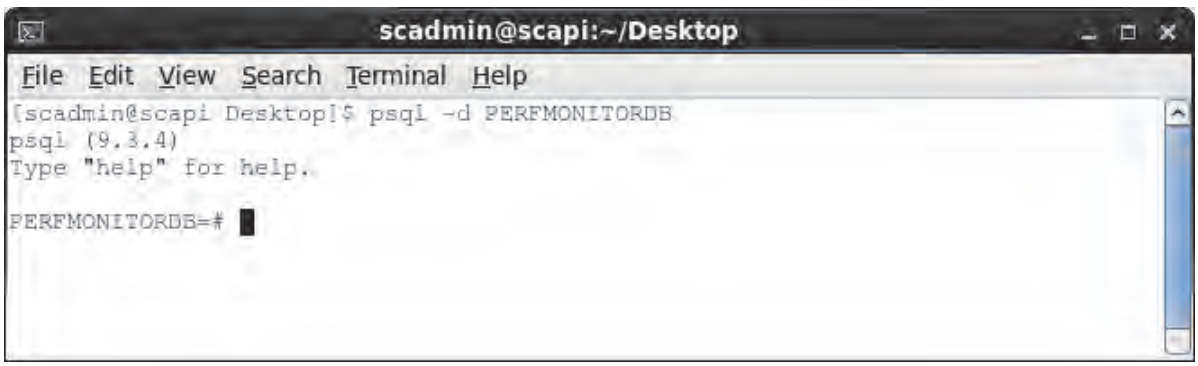


Note: The file has a header, which denotes the column names of the subsequent data. The data is comma-delimited. It may be difficult to see, but the file contains the following column: EndTime, HostName, TransactionsPerSecond, MemoryFreePercent, CpuBusy, FileControlBytessec64, Service, Location, Contact.

2. As the **scadmin** user, review the data in the PERFMONITORDB database and VM_HEALTH table that is in the PostgreSQL instance.

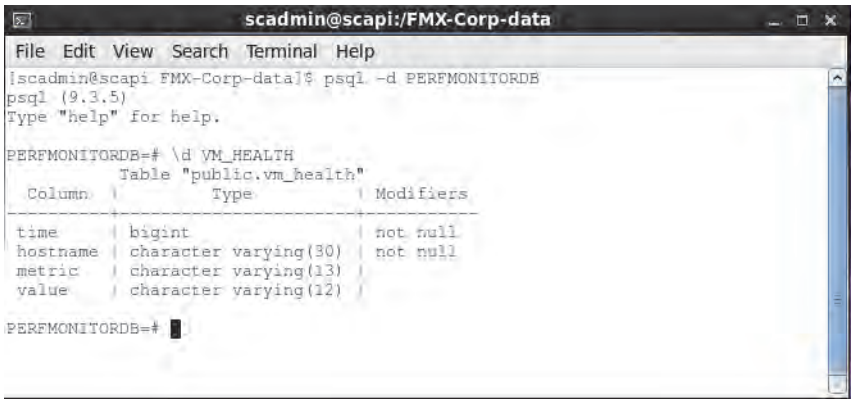
   a. Connect to the PERFMONITORDB database that is in the PostgreSQL instance. In a terminal, enter the following command to connect to PostgreSQL:

   ```
   psql -d PERFMONITORDB
   ```



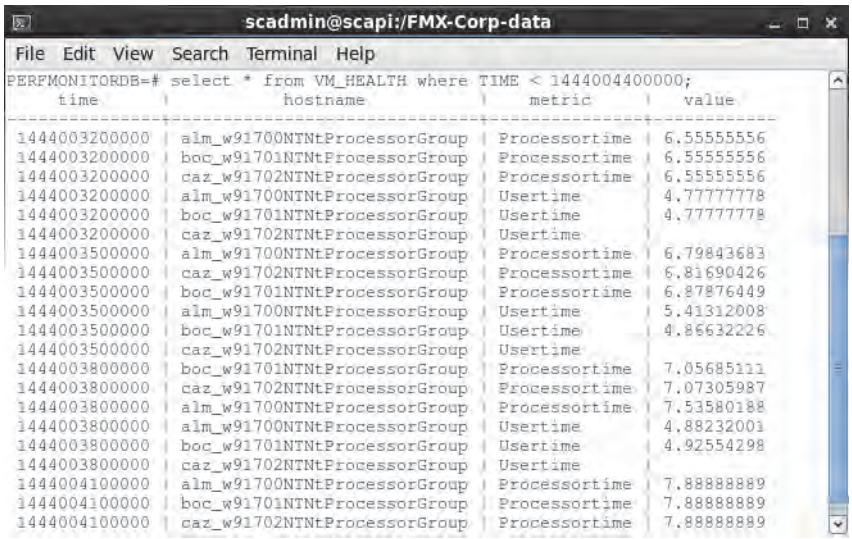
b. Display the schema and some data in the VM_HEALTH table. In the terminal window, enter the **\d VM_HEALTH** command.

```
\d VM_HEALTH
```



c. To display data in the VM_HEALTH table, use the SELECT statement. The table has 34,000 rows of data. Use a limiter to reduce the output. The epoch time stamp in this example limits the amount of data.

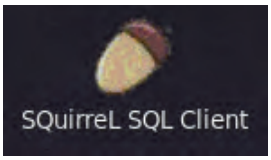
```
select * from VM_HEALTH where TIME < 1444004400000;
```



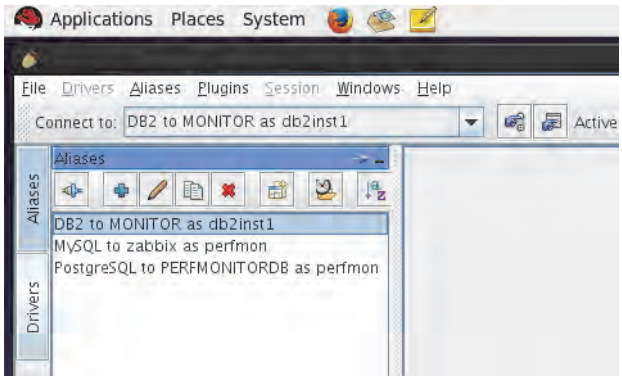
Important: Note the schema and data of the PostgreSQL VM_HEALTH table. This table is skinny because each row represents a different KPI. The metric switches between **Processortime** and **Usertime**.

d. Exit the psql command line with the **\q** command.

3. As the **db2inst1** user, review the data in the MONITOR database that is in a DB2 database. You use SQuirreL as a client tool to look at the data tables.
   a. Connect to the MONITOR tables in DB2 with the SQuirreL SQL client utility. On the Linux desktop, double-click the acorn icon to start SQuirreL SQL client.



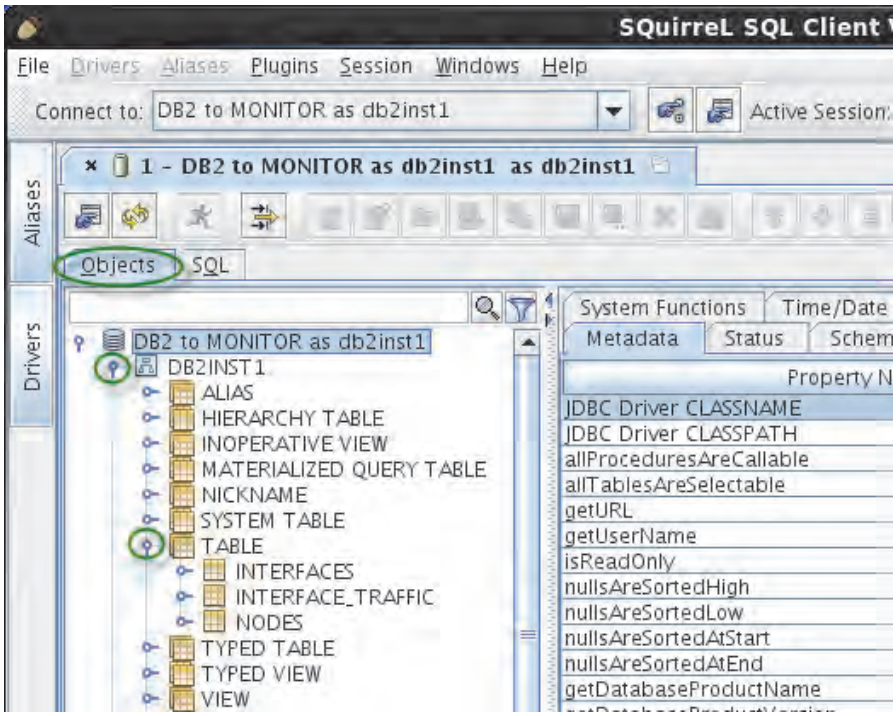
   b. In the left window, double-click the **DB2 to MONITOR as db2inst1** alias.



   c. You are auto-connected to the database.

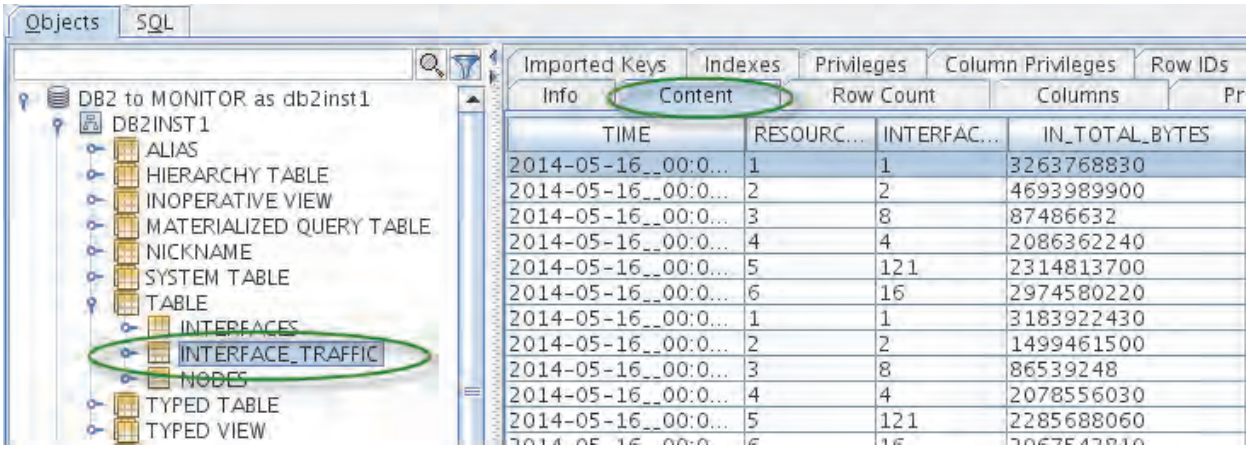
Note: Occasionally, you might receive a time-out error with your first connection to the database. Try connecting again. The problem usually resolves itself.

d. On the **Objects** tab, expand **DB2INST1**, and expand **TABLE.**

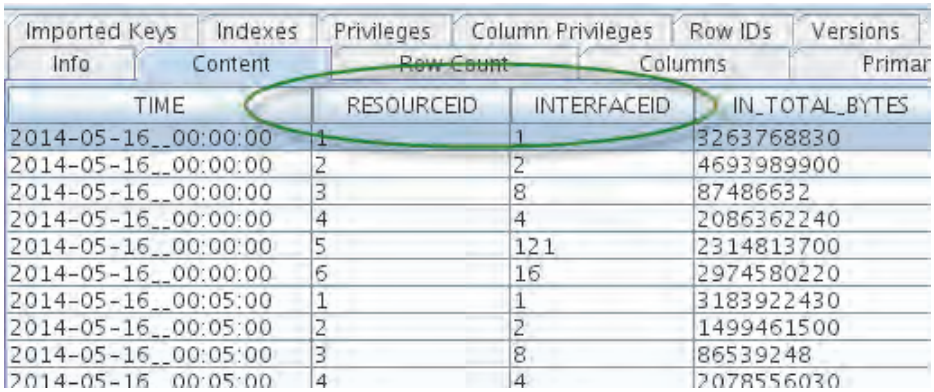


Note: On rare occasions, there are no objects displayed under the base database node in the object tree. Use the yellow refresh button in the upper left corner of the SQuirreL user interface to refresh the screen. This step usually resolves the problem and the sub-nodes appear.

e. Display the INTERFACE_TRAFFIC table and note the foreign keys. Select the **INTERFACE_TRAFFIC** table, and click the **Content** tab.

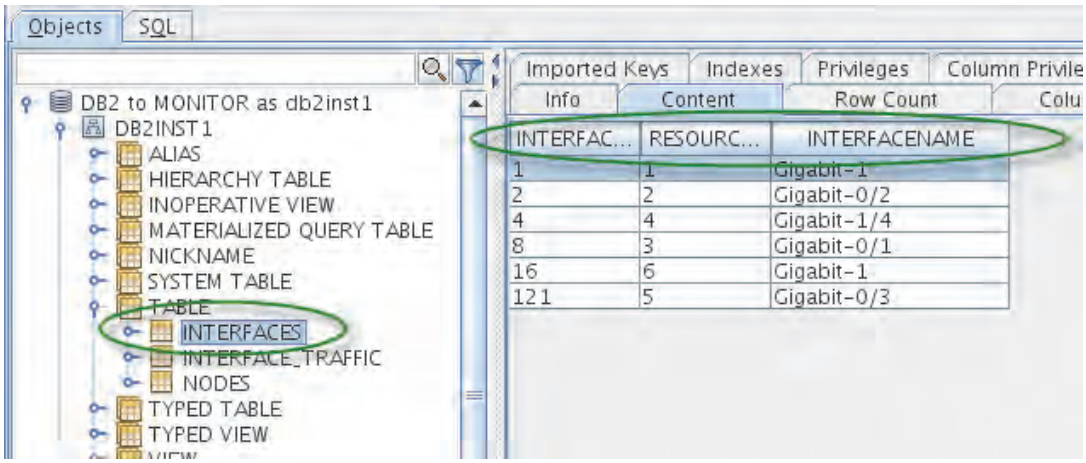


Note that the **RESOURCEID** and **INTERFACEID** uses foreign keys to reference the host name and interface of a device.

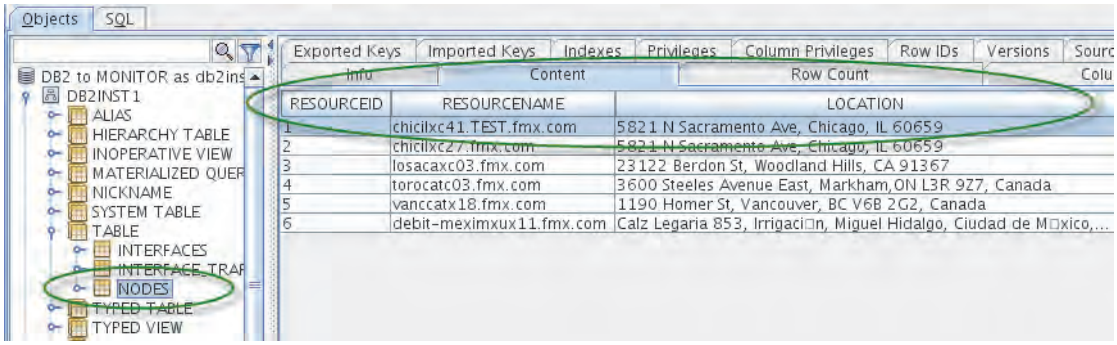


TIME	RESOURCEID	INTERFACEID	IN_TOTAL_BYTES
2014-05-16_00:00:00	1	1	3263768830
2014-05-16_00:00:00	2	2	4693989900
2014-05-16_00:00:00	3	8	87486632
2014-05-16_00:00:00	4	4	2086362240
2014-05-16_00:00:00	5	121	2314813700
2014-05-16_00:00:00	6	16	2974580220
2014-05-16_00:05:00	1	1	3183922430
2014-05-16_00:05:00	2	2	1499461500
2014-05-16_00:05:00	3	8	86539248
2014-05-16_00:05:00	4	4	2078556030

f. Select the **INTERFACES** tables and note the data that is on the **Content** tab. See how the **INTERFACEID** resolves to an **INTERFACENAME**. Note the **RESOURCEID** that is associated with this table.

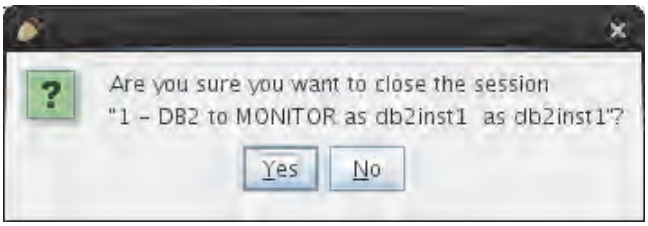


g. Select the **NODES** table and note the data that is on the **Content** tab. You can associate **RESOURCEID** with a **RESOURCENAME**. Note the additional columns associating Location and Contact with the resource.



Note: As you should see, there are foreign keys being used to reference the resources and nodes. These foreign keys need to be resolved before you mediate the data or you end up with meaningless numbers in the final Predictive Insights alarms.

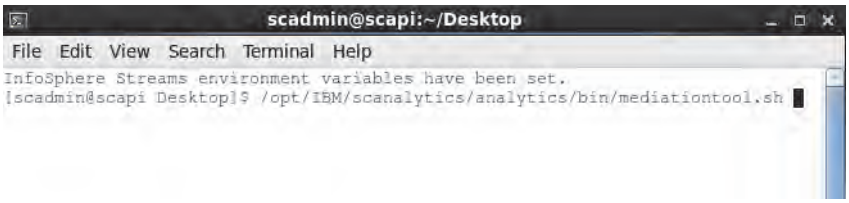
h. Close the SQuirreL SQL client window. Click **Yes** when prompted.



Exercise 2 Starting the data mediation tool

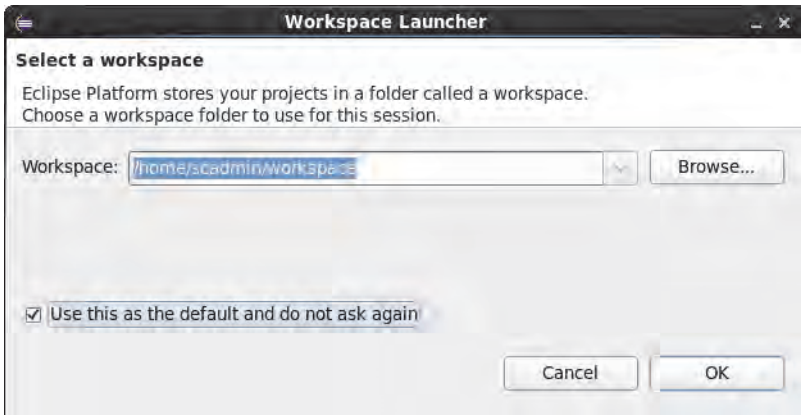
Having reviewed the format of the three data sources, you use the Predictive Insights mediation tool to model the data, which is then extracted and analyzed by the server. This exercise starts the data mediation tool and sets up your initial project.

1. Start the data mediation tool.
   a. Open a terminal window on the virtual machine desktop.
   b. Enter the command **/opt/IBM/scanalytics/analytics/bin/mediationtool.sh**.

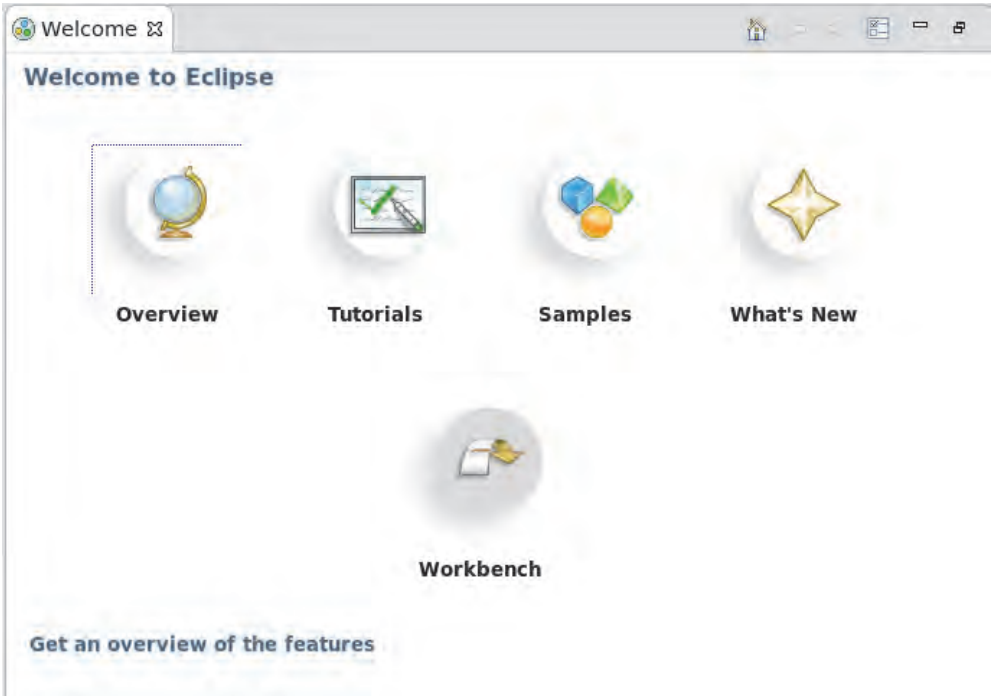


Important: If you receive an error when starting the mediation client, then your environment has not been refreshed with the new items that have added to the bashrc file after Predictive Insights was installed. Opening a new terminal window solves this problem.

c. Use the default workspace recommended by the launcher. Select **Use this as the default and do not ask again**. Click **OK**.

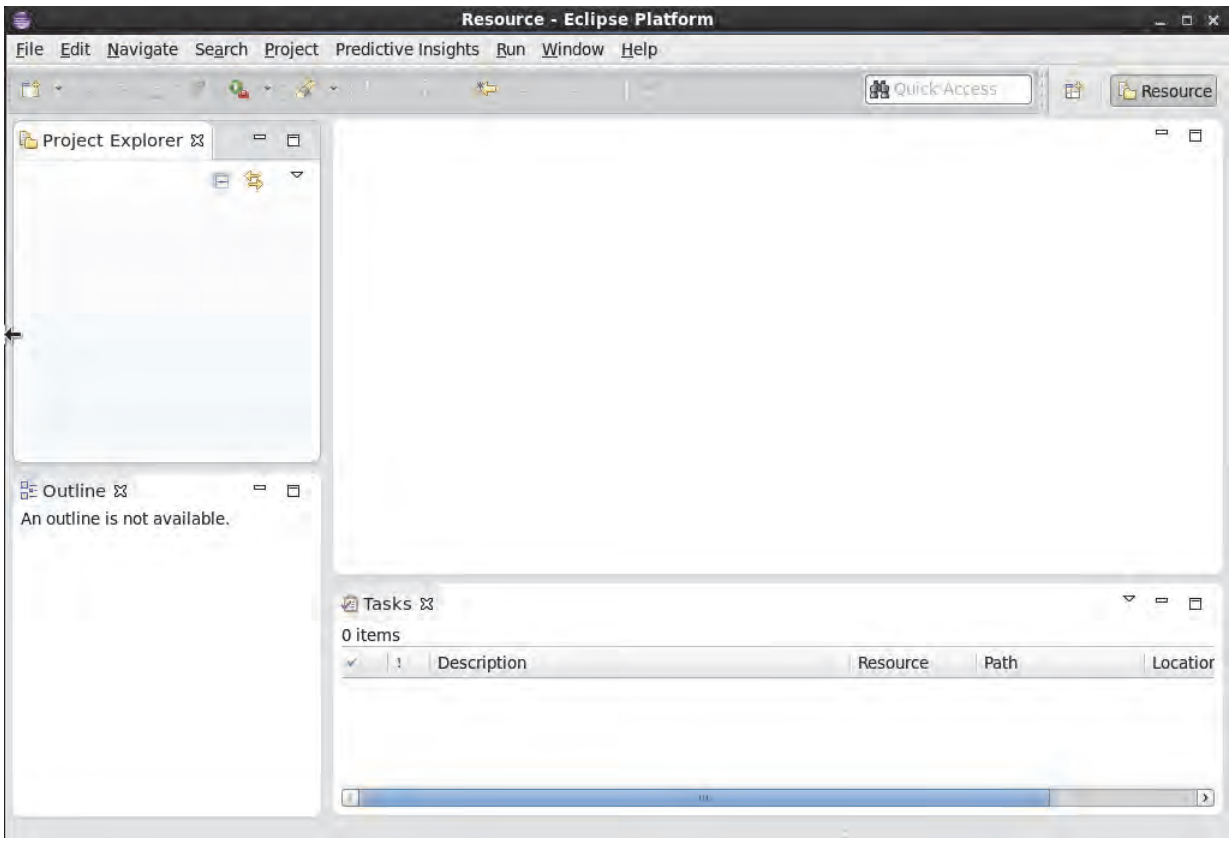


d. The data mediation tool opens.

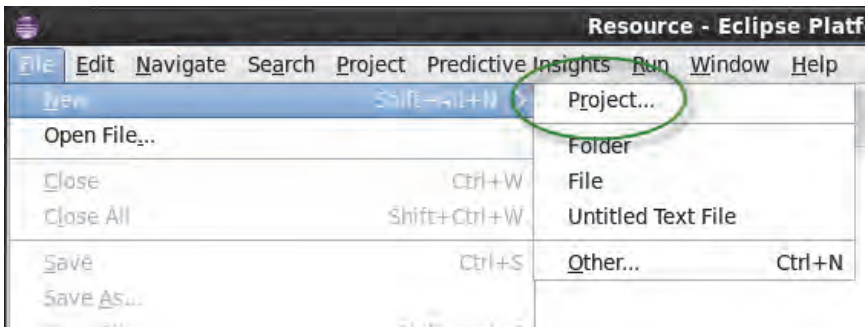


e. Select **Workbench**.

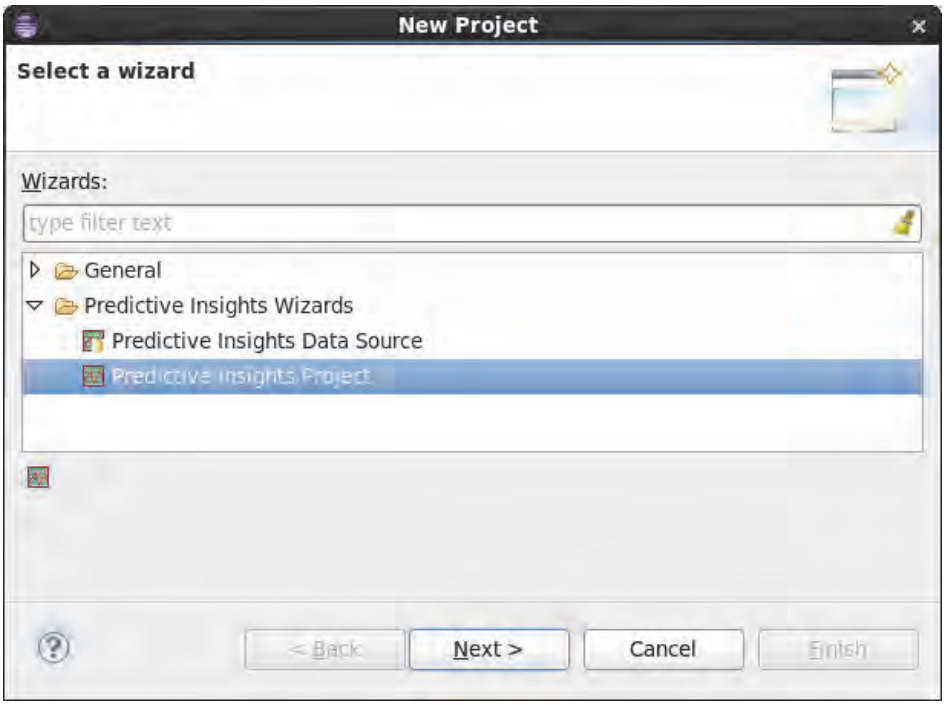
The data mediation tool workbench opens.



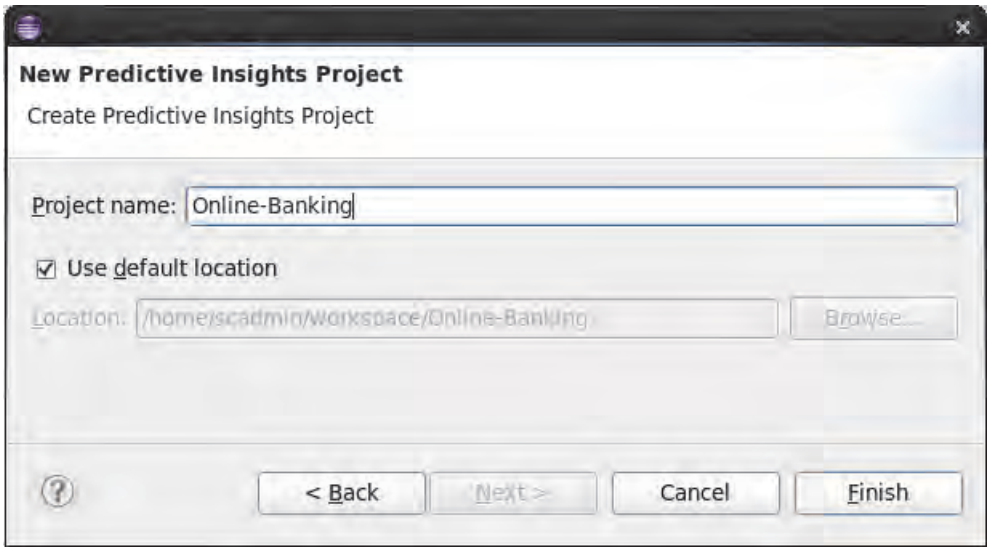
2. Create a project.
   a. In the workbench, select **File** > **New** > **Project**.



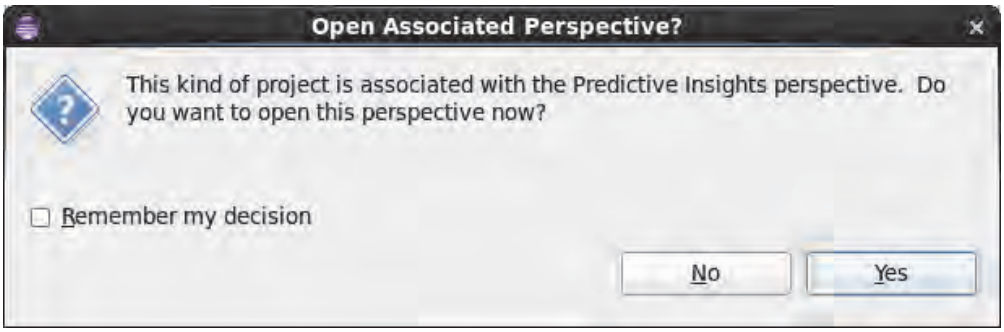
b. In the New Project window, open the **Predictive Insights Wizards** folder and select **Predictive Insights Project**. Click **Next**.



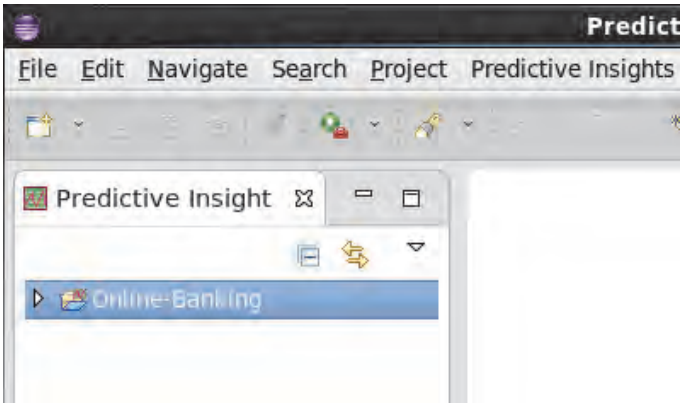
c. In the New Predictive Insights Project window, enter the project name of **Online-Banking**. Click **Finish**.



d. In the Open Associated Perspective window, click **Yes**.



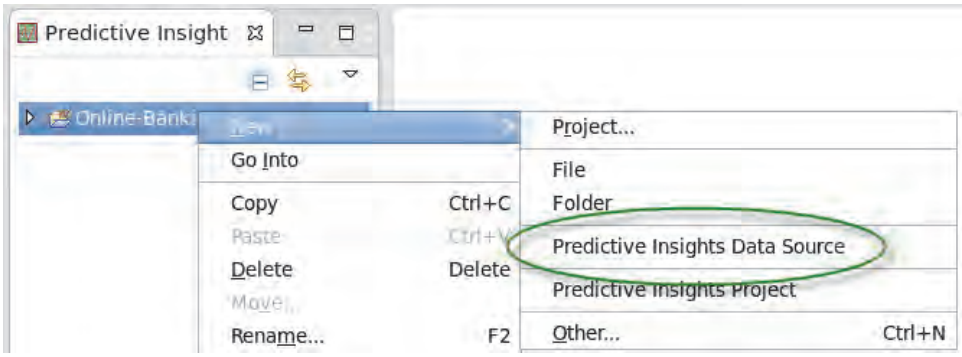
The project is created.



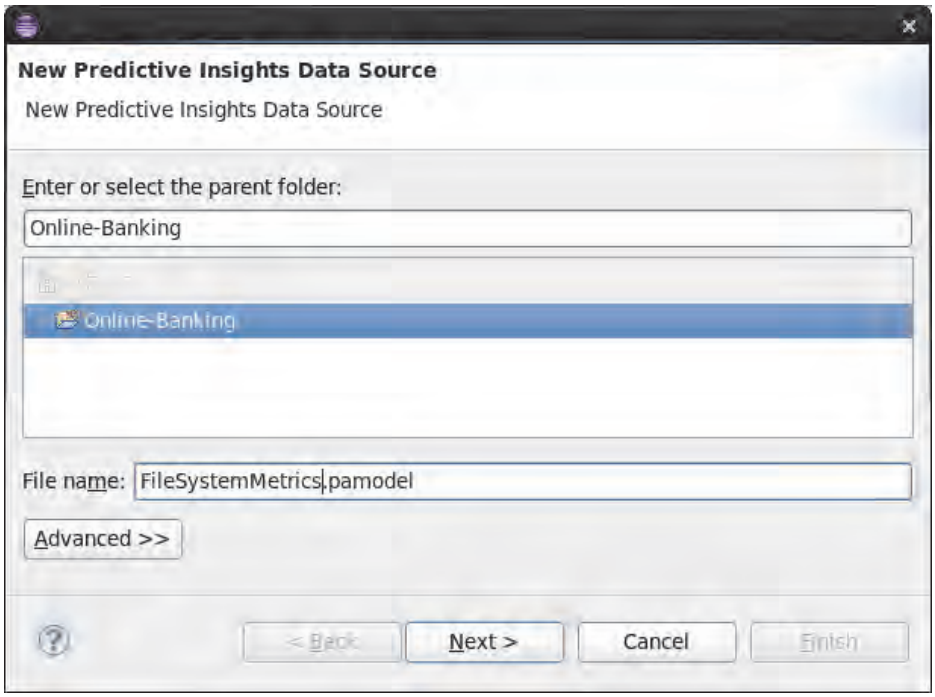
Exercise 3 Adding a CSV data source to the model

In this exercise, you add a data source that uses flat files in a directory structure to extract data about the systems being monitored. Named files are added to this directory from an external process. The naming of the files in this directory is important because the modeling tool uses the format of the files to determine a schema and which one to select.

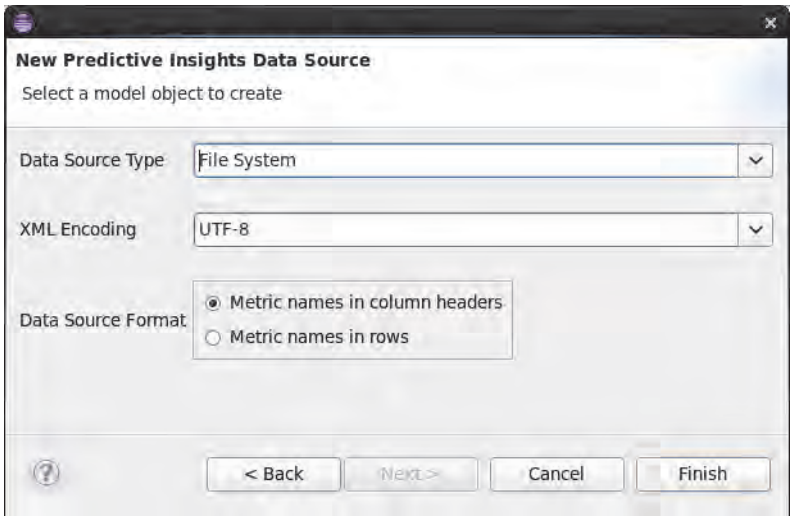
1. Add a file system data source to the **Online-Banking** project.
   a. Create a data source for the project by right-clicking **Online-Banking project** and selecting **New** > **Predictive Insights Data Source**.



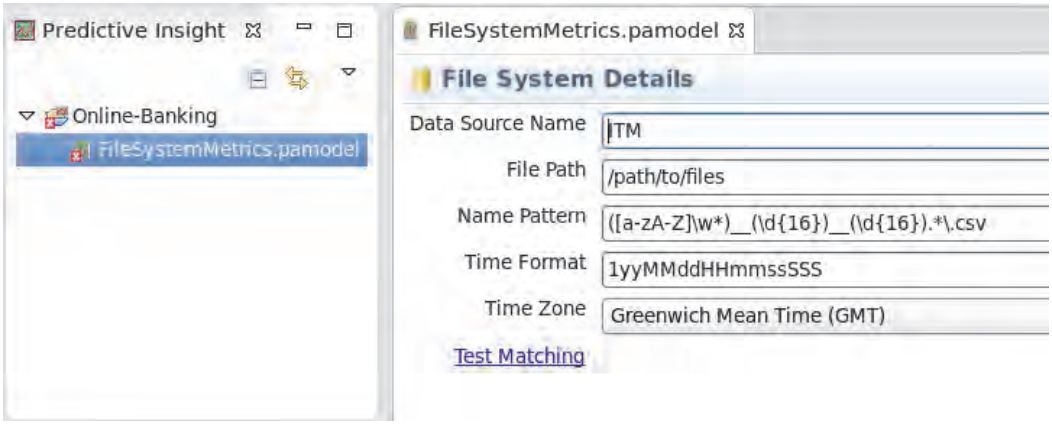
   b. Change the file name to **FileSystemMetrics.pamodel**. Click **Next**.



c. Select the **File System** data source from the **Data Source Type** menu. Leave the Data Source Format with the default. Click **Finish.**

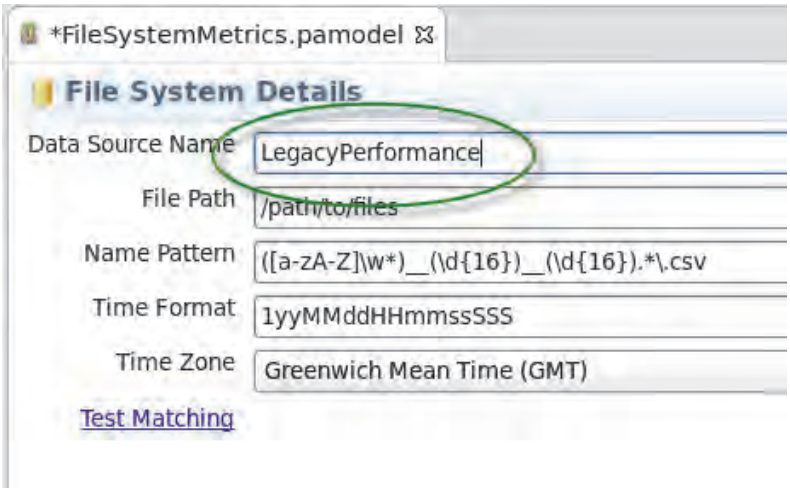


Note the change in the mediation tool GUI with the addition of the data source.

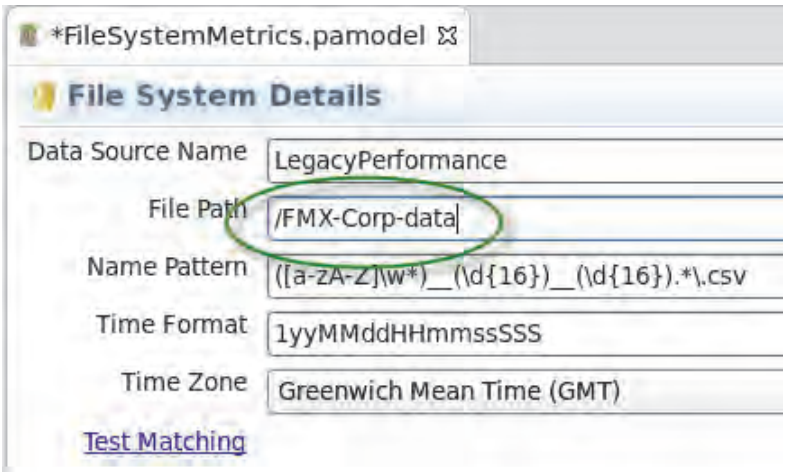


Note: In the previous exercise, you reviewed the contents of the flat file and confirmed that the metric names were in the column headers (also known as wide format).

2. Change the data source name to **LegacyPerformance** and point the data source to the **/FMX-Corp-data** file system.
   a. In the File System Details window, change the **Data Source Name** from **ITM** to **LegacyPerformance**.

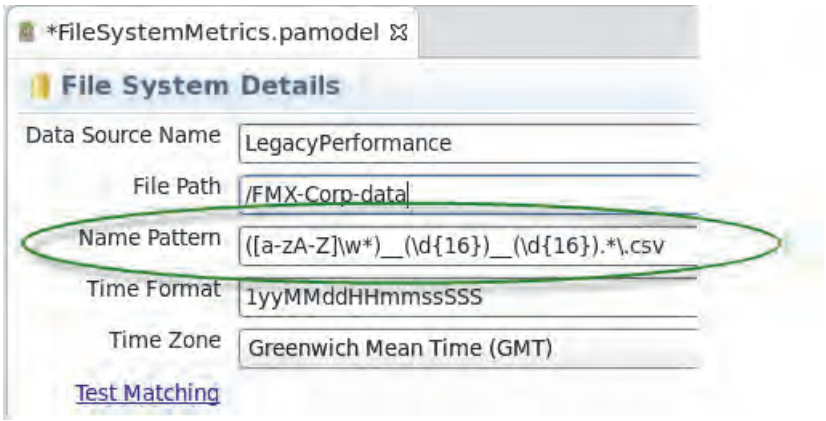


   b. Change the **File Path** from **/path/to/files** to **/FMX-Corp-data**. You change the path name in the field by placing the cursor on the path, left-clicking it, and typing a directory name into the field. The autocompletion feature helps with selecting the directory.

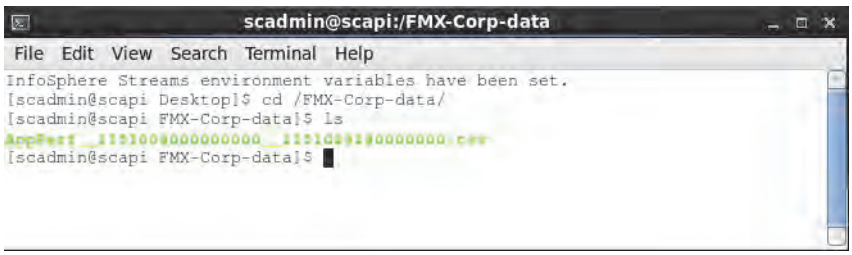


3. Review the name pattern and time formats for the file names.
   a. Check the **Name Pattern** regular expression and how it extracts data from the file name. See the format of the default regular expression that is used to parse the file names that are

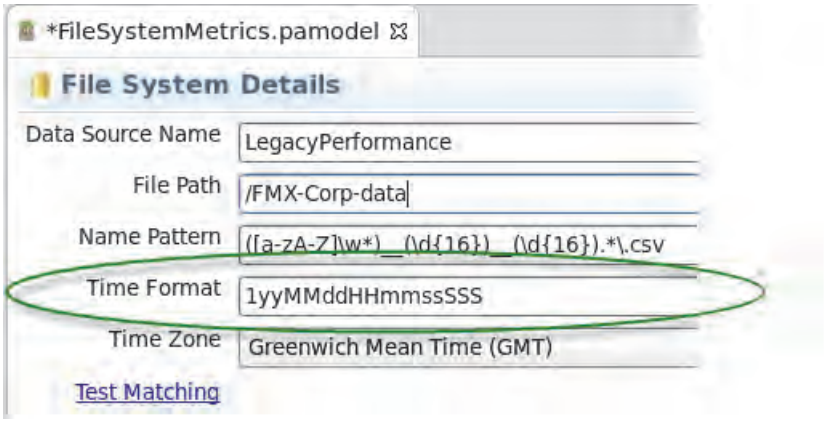
in /FMX-Corp-data. The regular expression requires that it extract the base file name (like a table name), the start time, and end time for the data that is in the file.

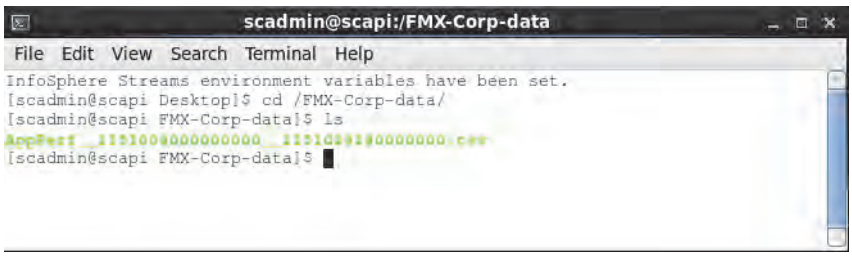


See how that regular expression aligns with actual files in /FMX-Corp-data.



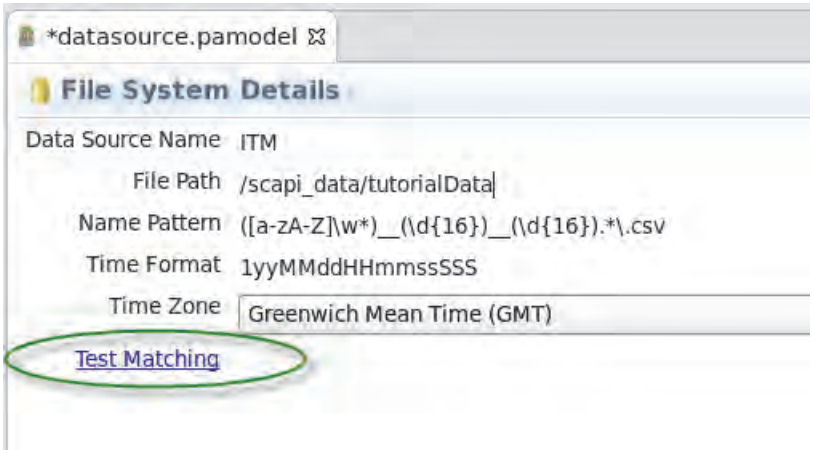
b. Check the **Time Format** and how it converts the extracted time stamp. Review the time format and compare it with the time stamp that is in the file names.

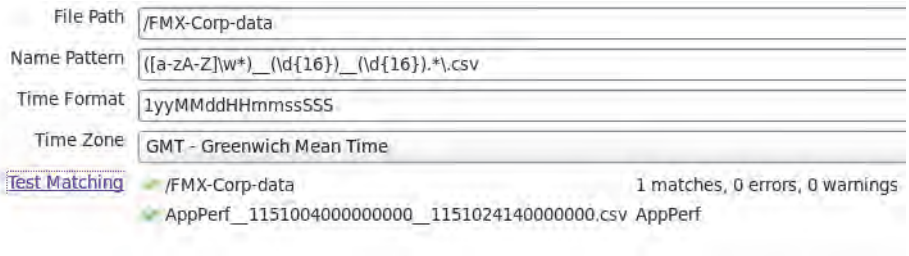




c. Note that the time stamps in the file names are in milliseconds.

4. Test the file naming convention against the files in the /FMX-Corp-data. Click the **Test Matching** link to test the files in the directory. Review the results. There should be no errors.

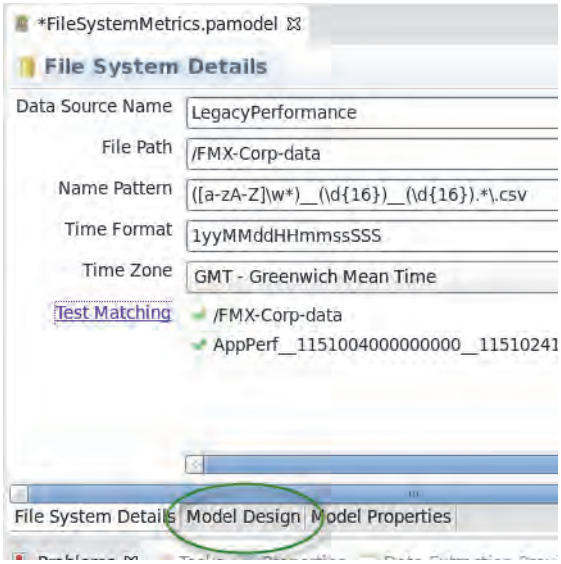




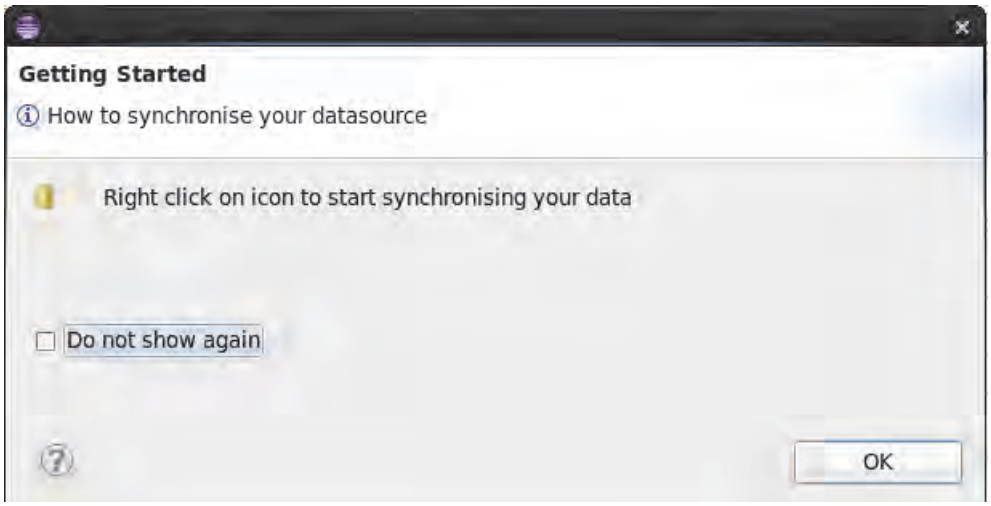
Exercise 4 Adding tables and metrics to the model

Now that the mediation tool can parse through the files in the file system, you define the tables for the data source. With file system data, the name of the file in Predictive Insights is like a table. An example might be **RespTime**. You also must ensure that the type and use of the data in the CSV files are aligned with the data model.

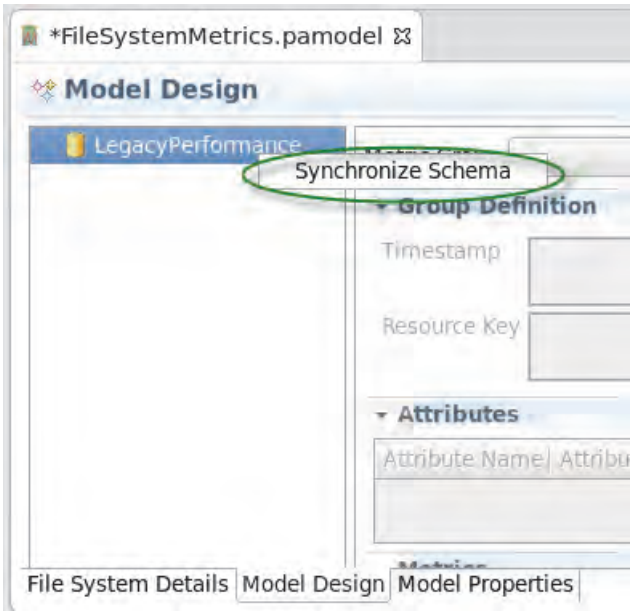
1. Synchronize the files in the file system into a table-like schema and add those tables to the model.
   a. Click the **Model Design** subtab on your data source to synchronize the **LegacyPerformance**.



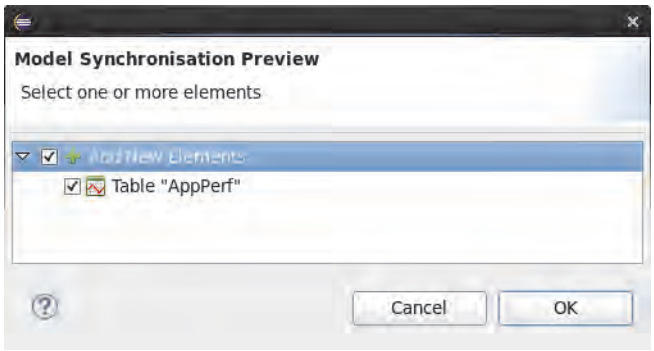
   b. When you see a tool tip that describes how to synchronize your data source, click **OK**.



c. Right-click the **LegacyPerforemance** icon in the **Model Design** area and select **Synchronize Schema**. The mediation tool analyzes the CSV Files and returns the data groups (or tables) that it finds.



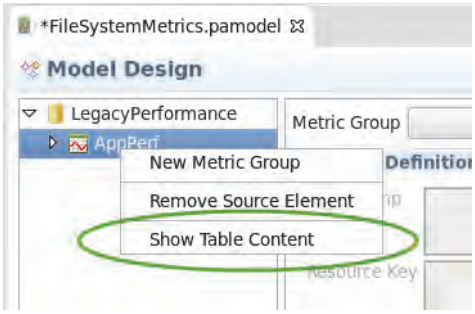
d. Add the new data elements that are discovered in /**FMX-Corp-data**. Use the wizard to select the data groups that you want to include in your model. Click the **Add New Elements** twistie, and review the elements in the file system. Note the sample table, **AppPerf**. Select **Add New Elements** and **Table AppPerf**. Click **OK**.



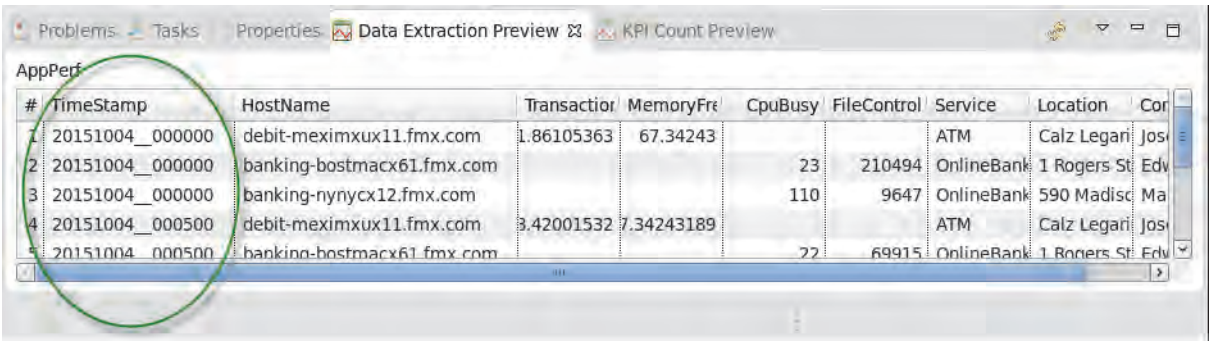
e. Check that data is retrievable from the RespTime table.

i. Click the **LegacyPerformance** twistie to see the LegacyPerformance data tables.

ii. Right-click the **AppPerf** table and select **Show Table Content.** This step ensures that raw data is retrievable from your data source.

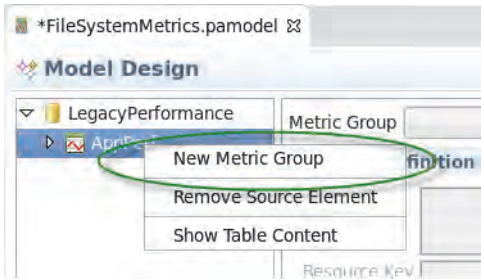


Note the Data Extraction Preview in the lower part of the window. Take special note of the **EndTime** column and the time stamp format. You account for this format in an upcoming task.

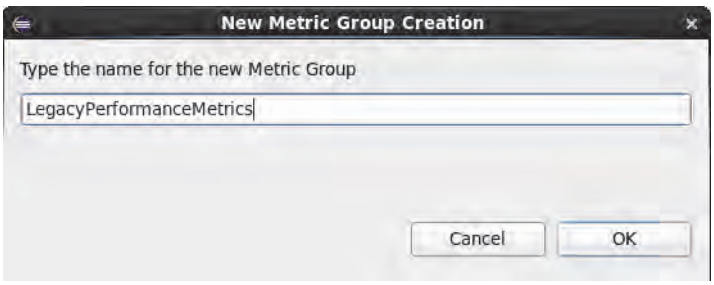


2. Add the LegacyServerResponse metric group from the RespTime table.

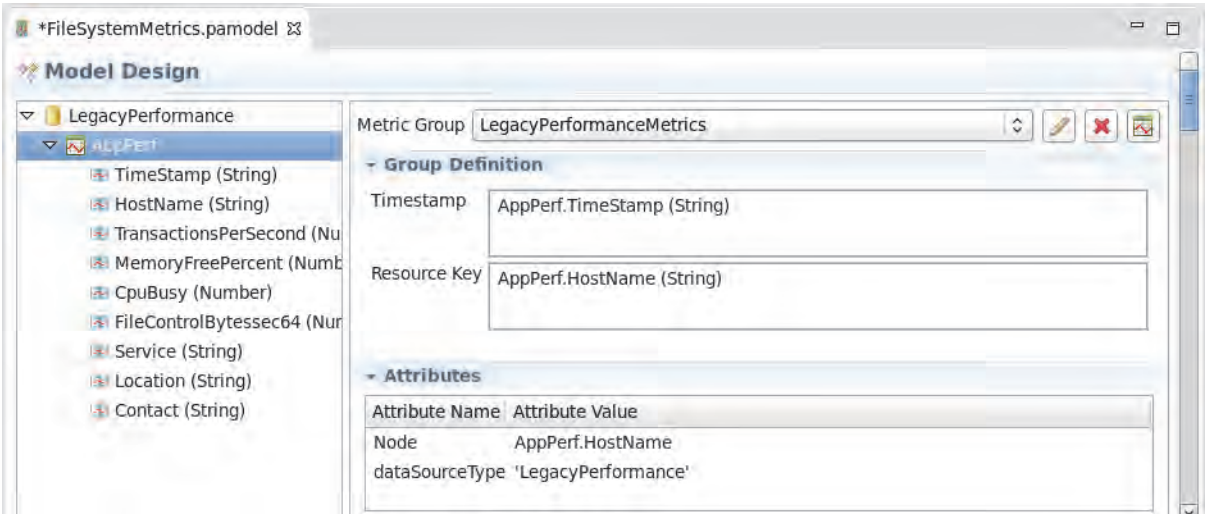
a. Select **AppPerf** > **New Metric Group**.



b. Change the default name of **AppperfGroup** to **LegacyPerformanceMetrics.** Click **OK**.



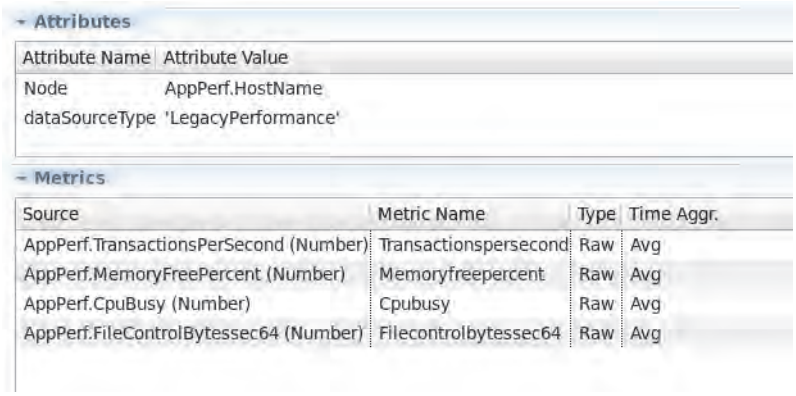
c. Check **Timestamp** and **Resource Key** for accuracy. Note the change in the GUI as the metric group is added.



Note: The time stamp and resource key that was selected automatically by the tool. Sometimes, the tool does not make the correct selections. You must correct this situation by deleting the key and selecting the correct one.

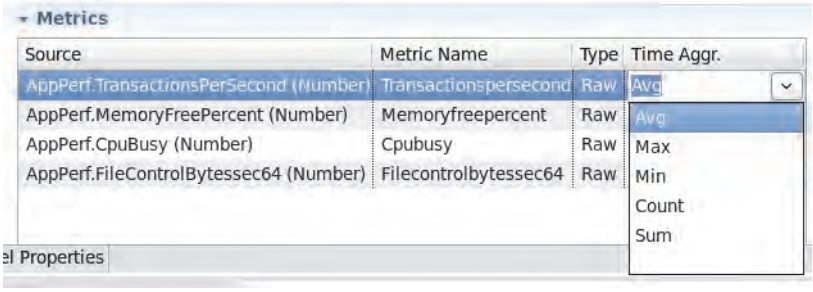
Important: If you must change the Resource Key, you must also add the new resource as an Attribute as well.

d. Check that the appropriate metrics are selected. Scroll down to the **Metrics** section. Note that there are four metrics are in your data table.

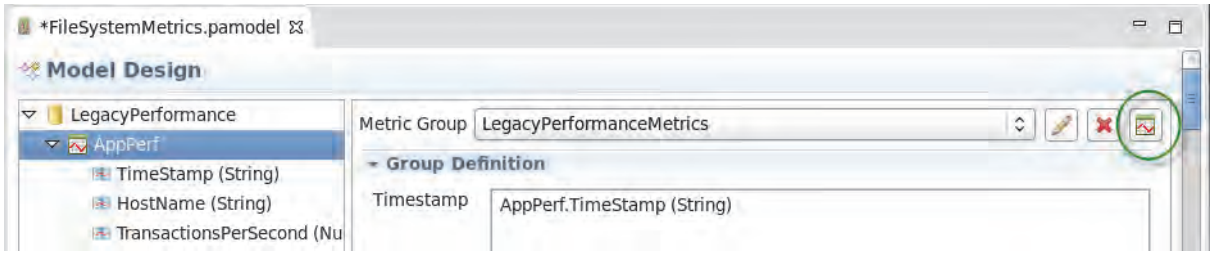


e. Click **Avg** for the **AppPerf.TransactionsPerSecond**. Note the list of options for Time Aggregation.

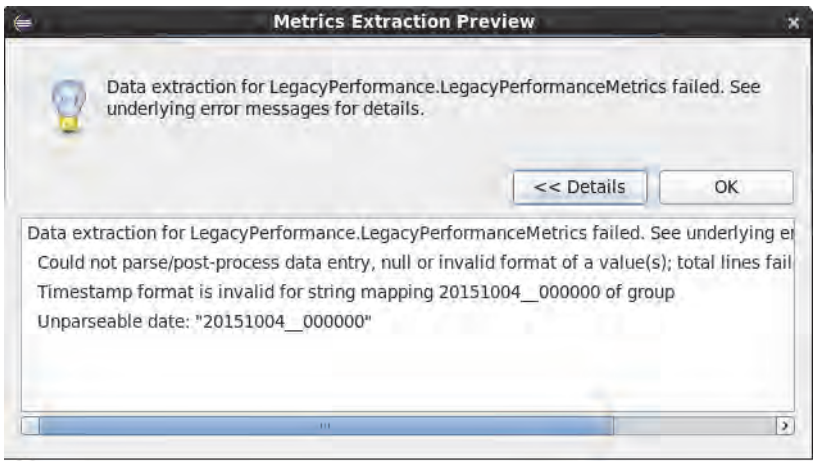
This selection determines what to do when multiple values are returned for this metric for any resource. For example, you aggregate every five minutes on a 1-minute data source. This selection determines what to do with the five values that are returned for a specific resource. Leave the Time Aggregation on **Avg**.



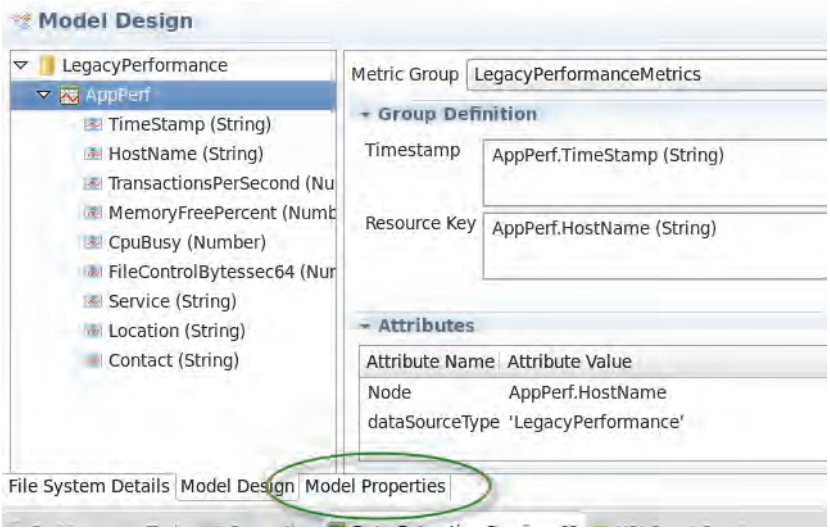
f. Preview data that is extracted by the **LegacyPerformanceMetrics** metric group. Scroll to the top of the **Model Design** tab, and select the **Preview the extract** button.



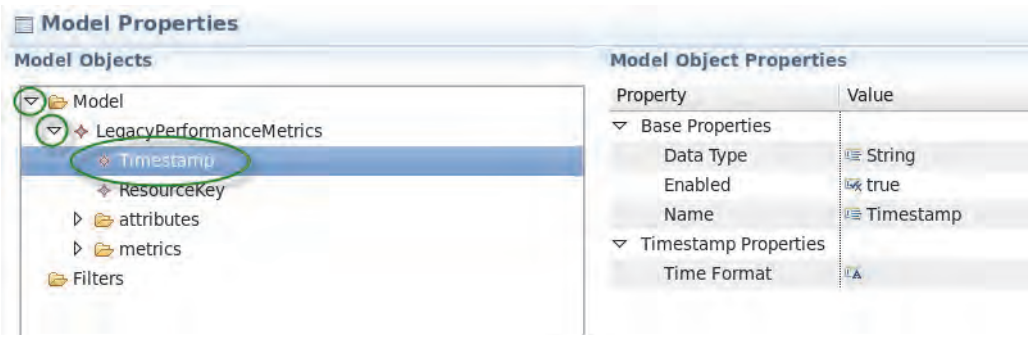
g. Review the error message about no mapping function for the string-based time stamp.



h. Change the time stamp format so that data can be extracted. Click the **Model Properties** tab.

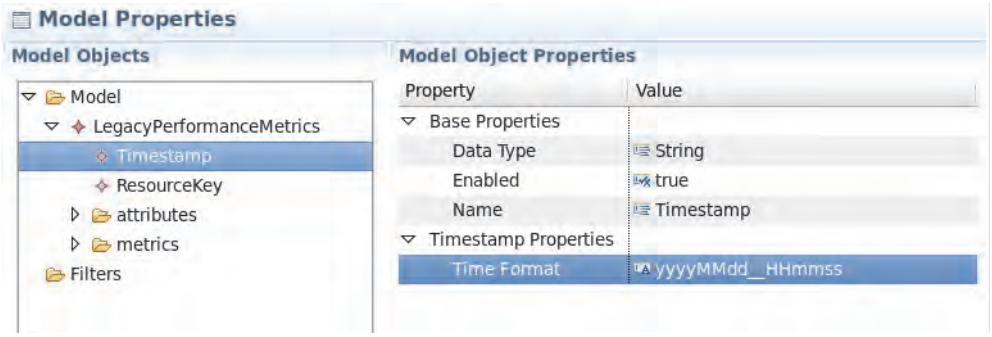


i. Expand **Model** > **LegacyPerformanceMetrics**. Select **Timestamp.**

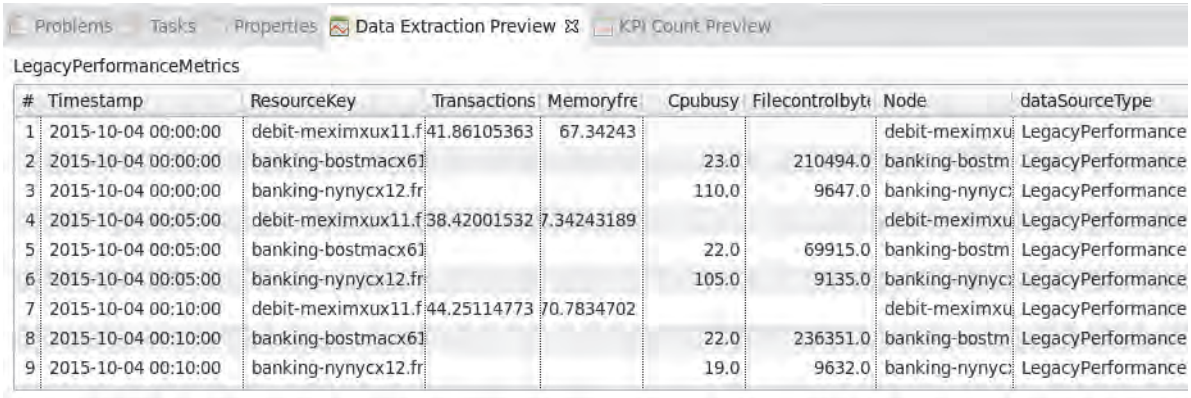


j. Enter the **Time Format** that aligns with the time stamp string that is used in the data, the time stamp format that you noted in.

`yyyyMMdd__HHmmss (note there are two "_" between dd and HH)`



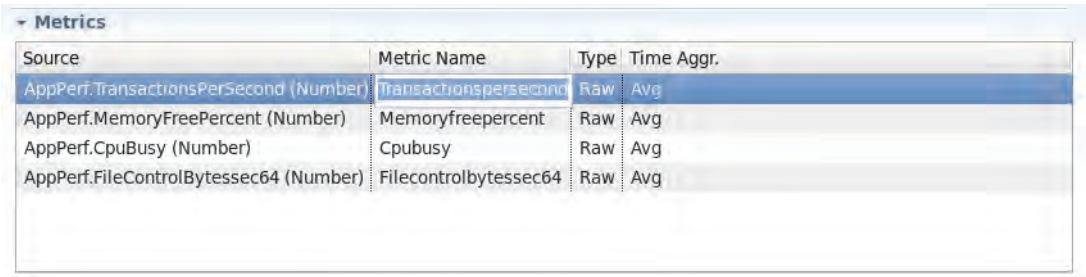
k. Return to **Model Design** tab, and preview the data.



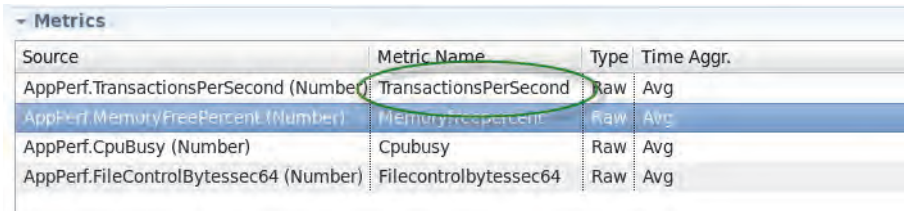
#	Timestamp	ResourceKey	Transactions	Memoryfre	Cpubusy	Filecontrolbyt	Node	dataSourceType
1	2015-10-04 00:00:00	debit-meximxux11.f	41.86105363	67.34243			debit-meximxu	LegacyPerformance
2	2015-10-04 00:00:00	banking-bostmacx61			23.0	210494.0	banking-bostm	LegacyPerformance
3	2015-10-04 00:00:00	banking-nynycx12.fr			110.0	9647.0	banking-nynyc	LegacyPerformance
4	2015-10-04 00:05:00	debit-meximxux11.f	38.42001532	7.34243189			debit-meximxu	LegacyPerformance
5	2015-10-04 00:05:00	banking-bostmacx61			22.0	69915.0	banking-bostm	LegacyPerformance
6	2015-10-04 00:05:00	banking-nynycx12.fr			105.0	9135.0	banking-nynyc	LegacyPerformance
7	2015-10-04 00:10:00	debit-meximxux11.f	44.25114773	70.7834702			debit-meximxu	LegacyPerformance
8	2015-10-04 00:10:00	banking-bostmacx61			22.0	236351.0	banking-bostm	LegacyPerformance
9	2015-10-04 00:10:00	banking-nynycx12.fr			19.0	9632.0	banking-nynyc	LegacyPerformance

3. Rename the metrics to be more meaningful for end users. Change the metrics names to use a capital letter for each word in the name.

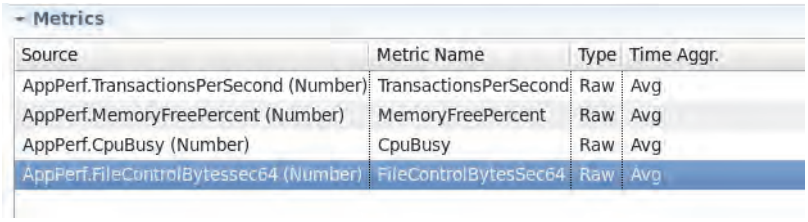
a. In the Metrics window on the Model Design page, select the **Transactionspersecond** Metric Name.



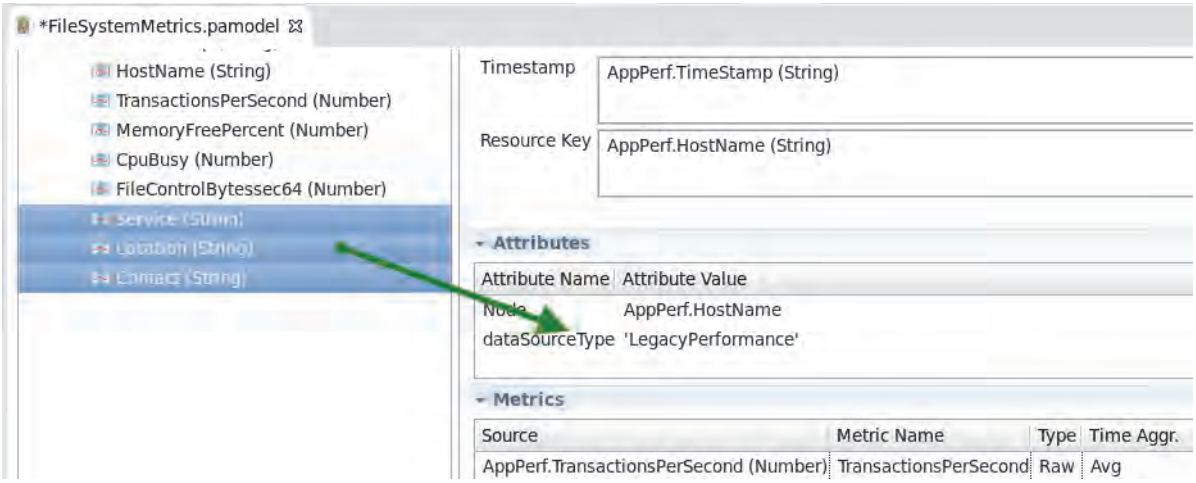
b. Modify the name to be TransactionsPerSecond



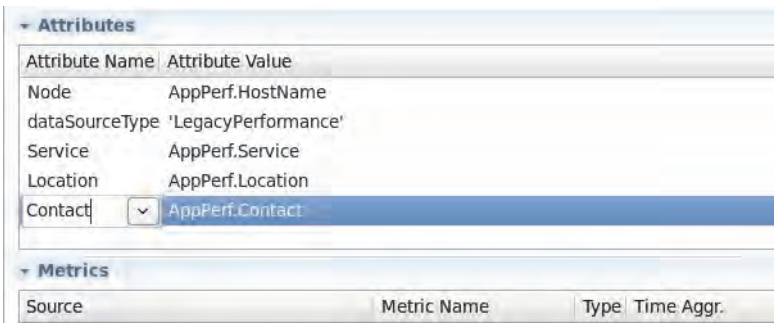
c. Repeat this process for the other metrics.



4. Add the **Service**, **Location**, and **Contact** information as attributes.
   a. Multi-select the **Service**, **Location**, and **Contact** fields in the left pane and drag them to the attributes area.

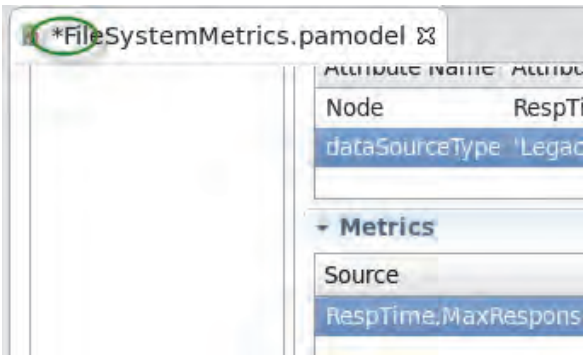


   b. Modify the attribute names either by selecting one from the pull down list or by typing it in. Be sure to use **Location**, **Service**, and **Contact** as the Attribute Name.



Note: The attributes you are supplying here must be data items that are in the data source itself. To add additional attributes from other sources would require a tool like Impact to enrich the alarm in the OMNIbus database. These attributes become part of the alarm that is generated by Predictive Insights.

5. Save the changes to the data source. Note the asterisk on the tab name. It denotes unsaved changes.



6. Click **File** > **Save**.

Exercise 5 Creating a database view

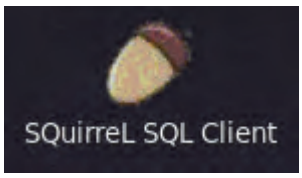
The Predictive Insights data mediation client cannot join data tables in databases or CSV files. You must ensure that a data table has the following items in it for Predictive Insights to extract the data:

- **Time stamp**: It must have a time stamp either in a string-readable form or in UNIX epoch time.
- **Host name**: It must have a host name that is common across all the data sources where the host is used.
- **Additional information**: It can have any other useful information to uniquely identify the resource (or subresource attribute) that you want to measure; for example, interface name, location, contact information, etc.
- **Metrics**: It contains values that measure specific performance attributes of your infrastructure.

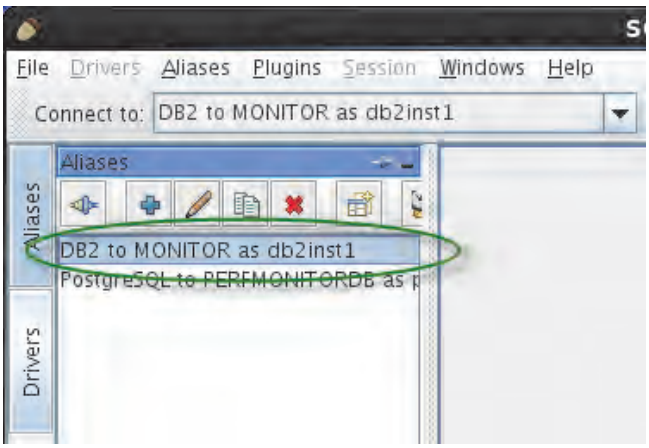
In Exercise , “,” on page 62, you noted that the tables in the DB2 database used foreign keys. If you access the INTERFACE_TRAFFIC table to retrieve useful metrics, you get only resource names and interfaces that are integers. The values are useless in Predictive Insights. To solve this problem, you must create a database view in DB2 that resolves these foreign keys into actual names.

Complete the following steps:

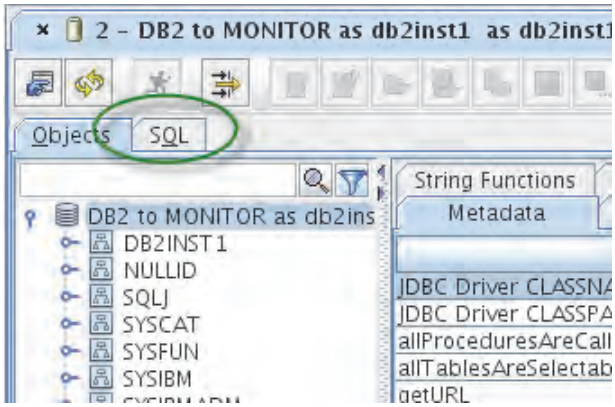
1. Start the SQuirrreL JDBC database client software and connect to the PostgreSQL database. On the Linux desktop, double-click the acorn icon to start **SQuirreL SQL Client**.



2. In the left window, double-click the **DB2 to MONITOR as db2inst1** alias. The client connects to the database and displays information about the schema.

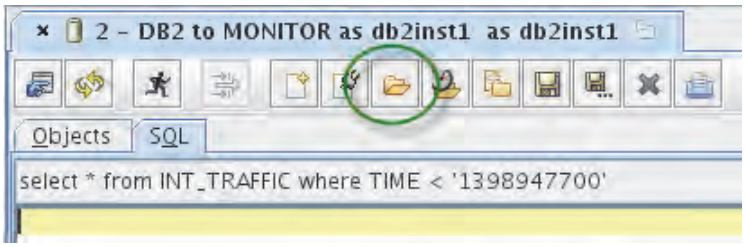


3. Create a view that resolves the interface and resource foreign keys into actual host names and interfaces.
   a. Click the **SQL** tab.

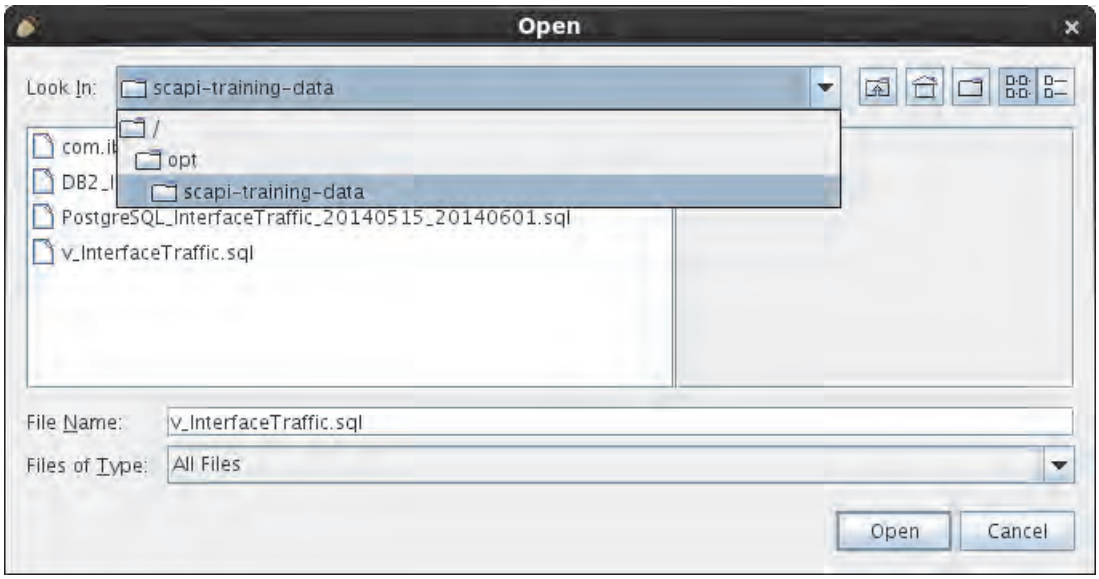


Note: To save time, the commands to create the view are in the **/opt/scapi-training-data/v_InterfaceTraffic.sql** file.

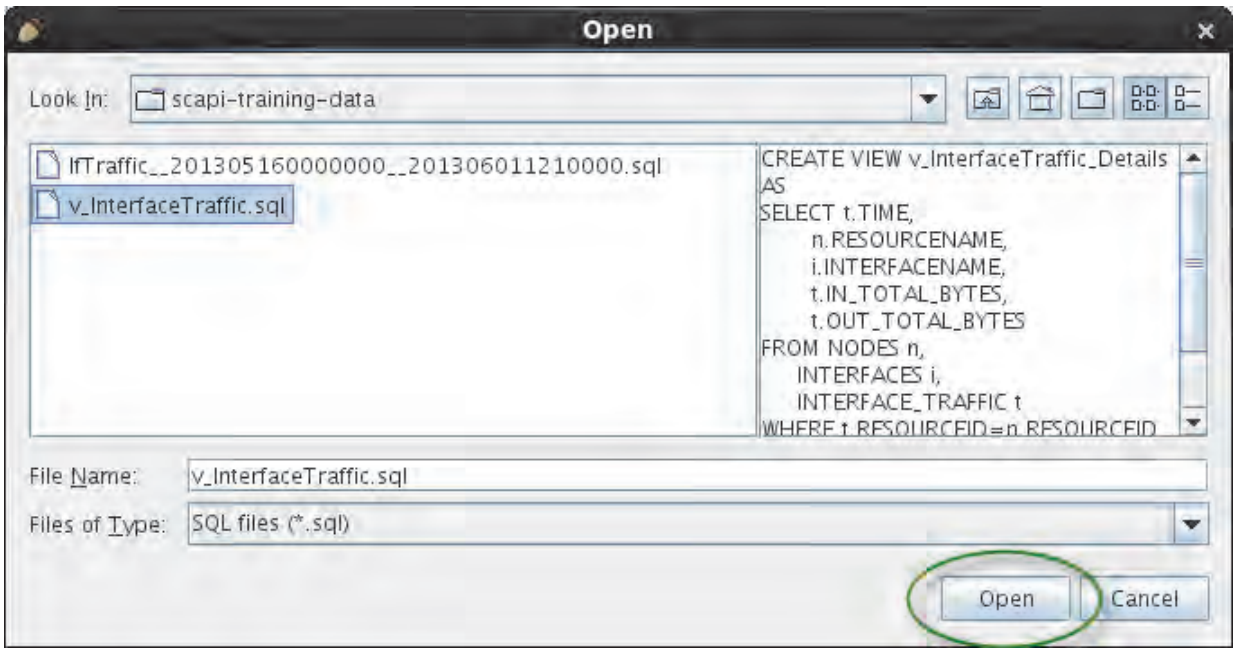
   b. Click the **Open a file** icon on the SQL tab.



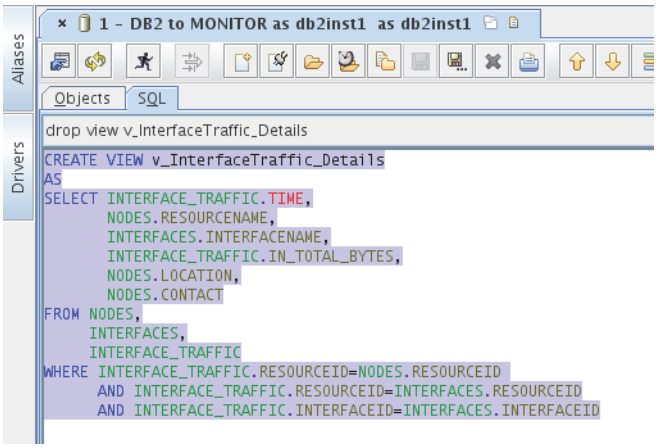
   c. Navigate to **/opt/scapi-training-data.**



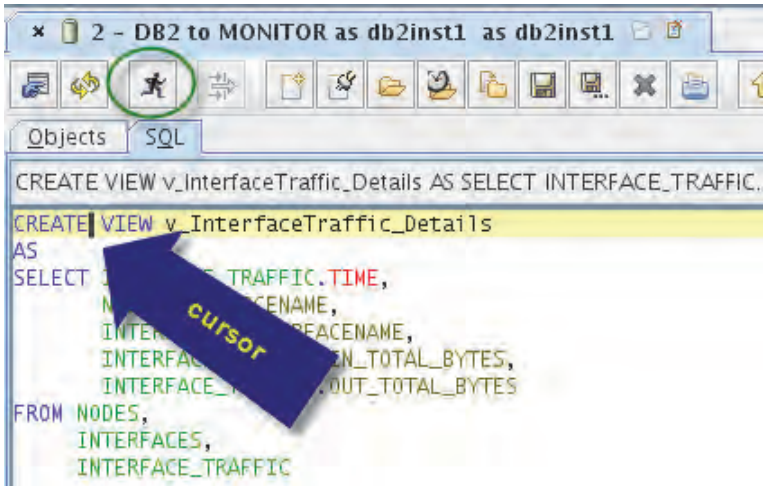
d. Select the **v_InterfaceTraffic.sql** file, and click **Open.**



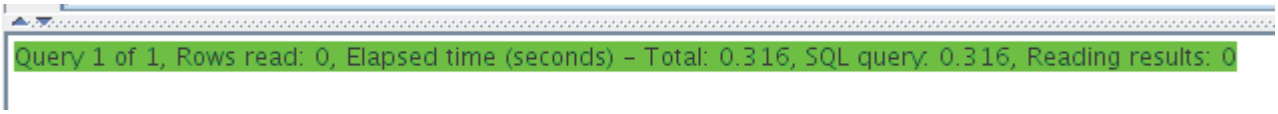
Note the SQL that is used to combine the three tables.



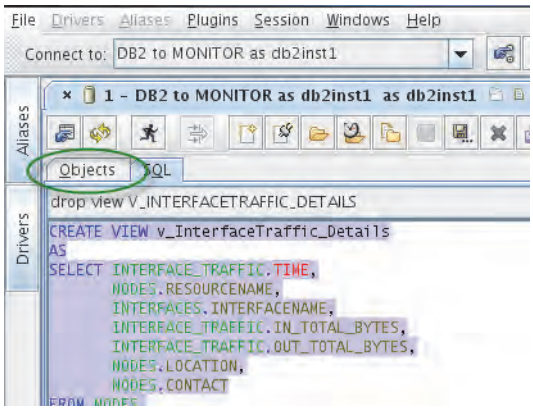
e. Place the cursor next to the CREATE statement and select the **Run SQL** icon.



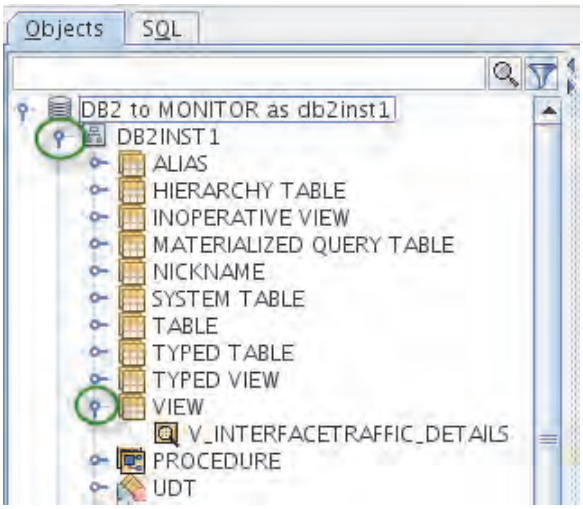
f. Verify that no errors were created.



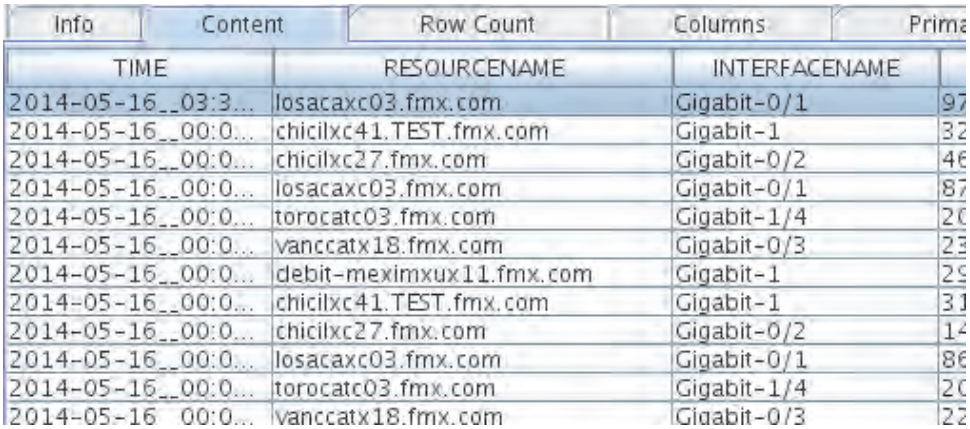
4. Validate that the view works correctly.
   a. Click the **Objects** tab in SQuirreL.



b. Expand **DB2INST1** > **VIEW.**



c. Select **V_INTERFACETRAFFIC_DETAILS**, and click the **Content** tab. Validate that the **RESOURCENAME** and **INTERFACENAME** are defined correctly.



5. Close SQuirreL.

Exercise 6 Adding a database data source to the model

Adding a database data source is like adding a CSV data source, except the connection methods are different. In this exercise, you connect to a data table that is in the DB2 database. The connection details are as follows:

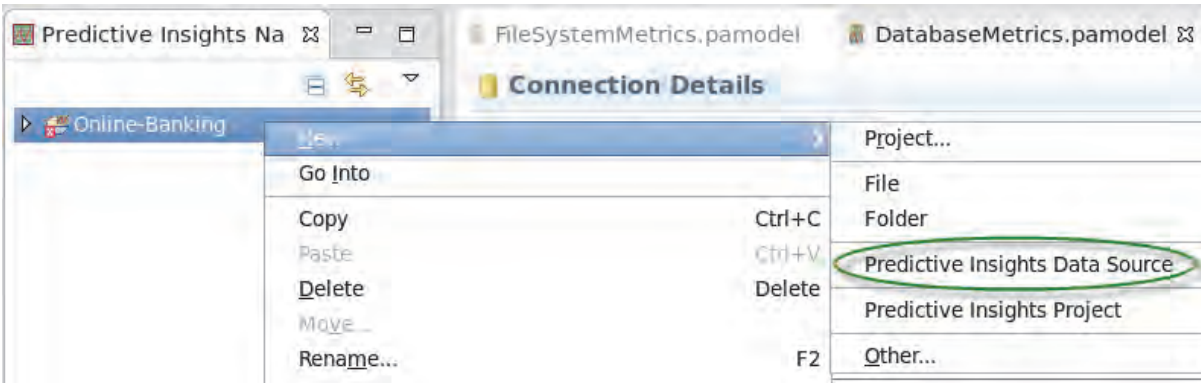
- Host name: **scapi.tivoli.edu**
- Database: **MONITOR**
- Port number: **50000**
- Schema: **DB2INST1**
- User name: **db2inst1**
- Password: **object00**

Complete the following steps:

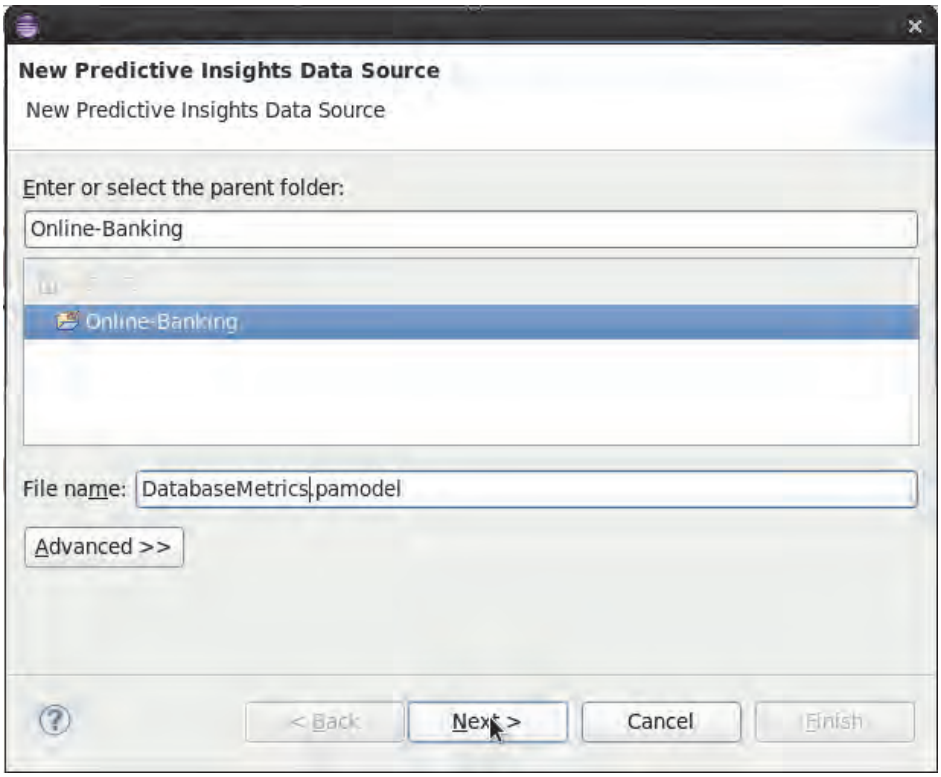
1. If it is not already started, start the data mediation client. In a terminal window, enter the following command:

   `/opt/IBM/scanalytics/analytics/bin/mediationtool.sh`

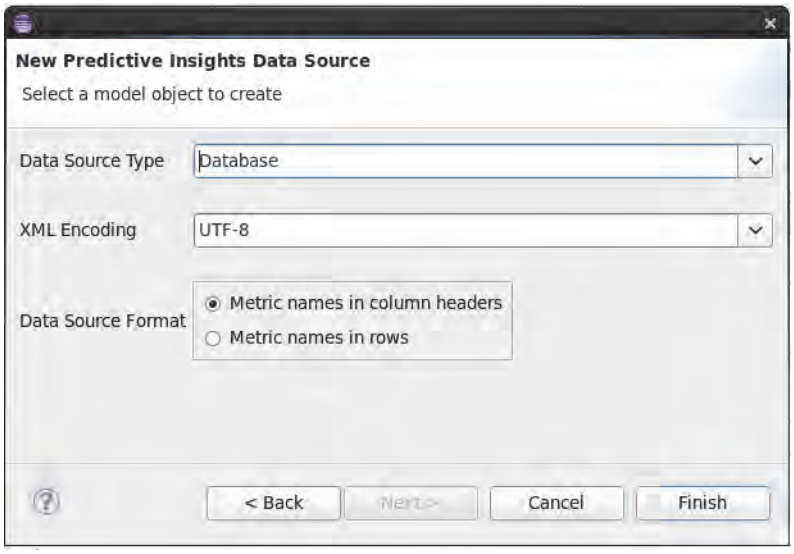
2. Create a new database data source. Right-click the **Online-Banking** project and select **New** > **Predictive Insights Data Source.**



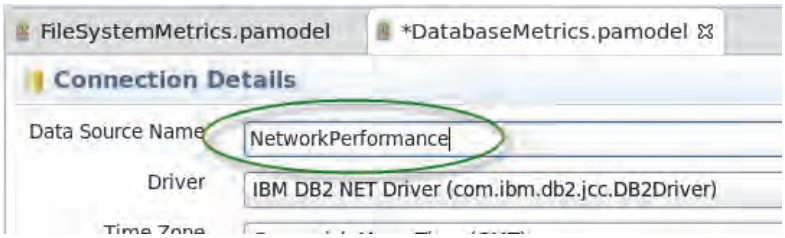
3. Name the data source **DatabaseMetrics.pamodel**, and click **Next**.



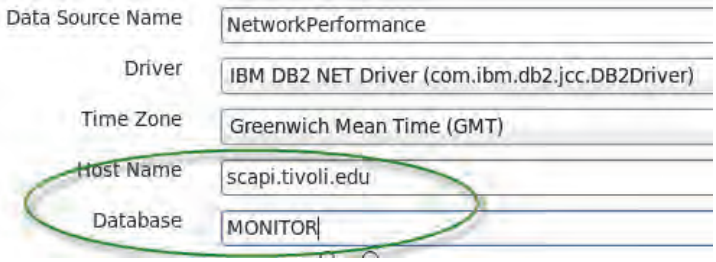
4. Select **Database** as the **Data Source Type** and leave the default database format. Click **Finish**.



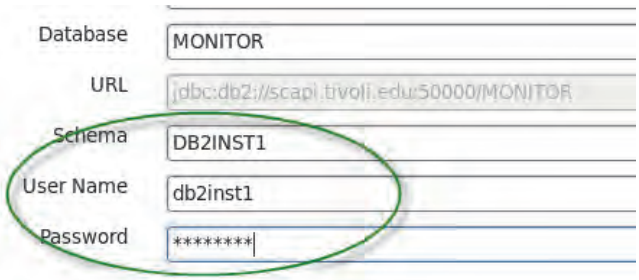
5. Change the data source name to **NetworkPerformance**. Enter the connection details to the DB2 database, and test the connection.
   a. Click the **Data Source Name** field and change **ITM** to **NetworkPerformance**.



   b. Enter the host name of **scapi.tivoli.edu** and database name of **MONITOR.**



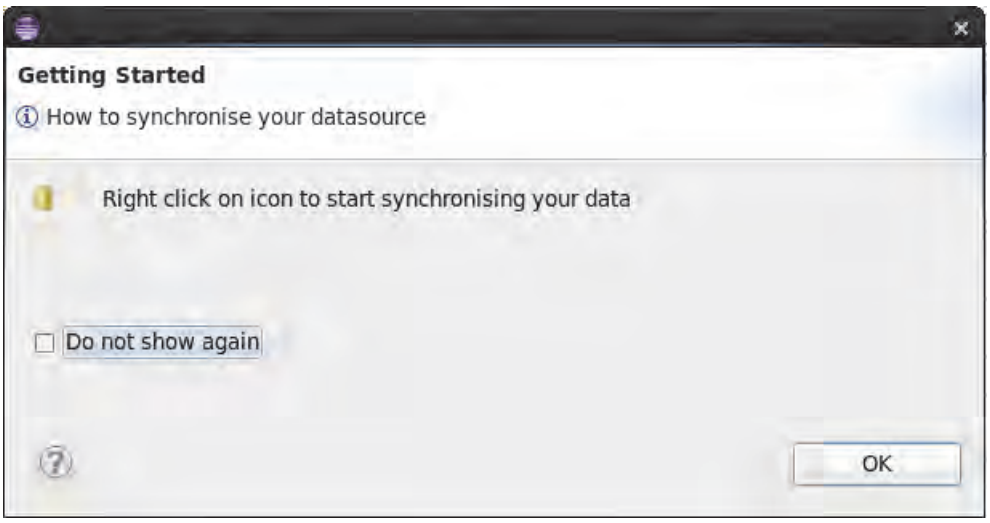
   c. Scroll down and enter the schema of **DB2INST1**, user name of **db2inst1** (note capitalization), and password of **object00.**



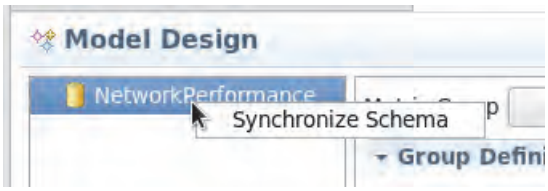
   d. Scroll down, and click **Test Connection**.

Note: If the connection is not successful, check the accuracy of your entries and test again. If that is not successful, contact your instructor.

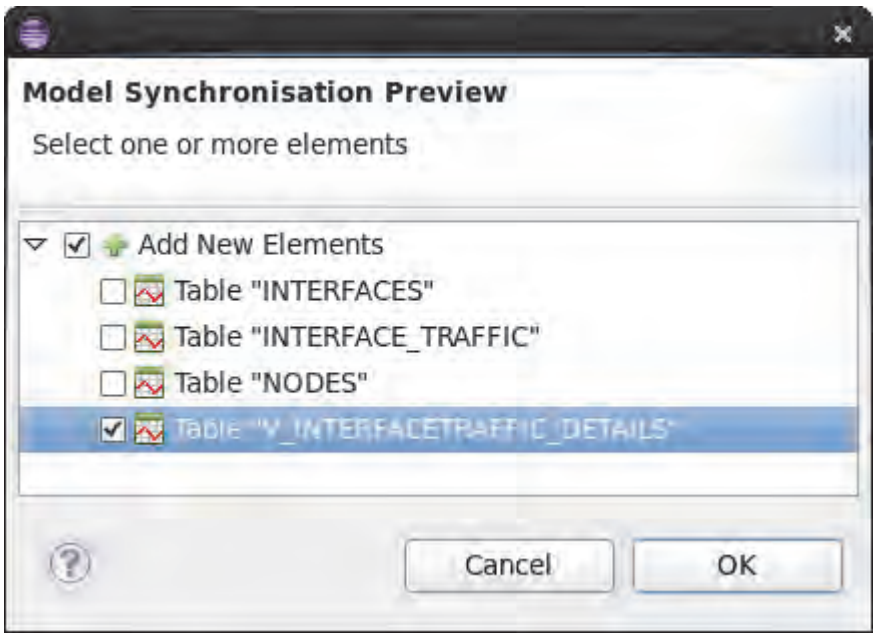
6. Synchronize schema, review the table contents, and add a new metric group called NetworkIO.
   a. Click the **Model Design** tab. You are shown a tool tip that describes how to synchronize your data source. Click **OK**.



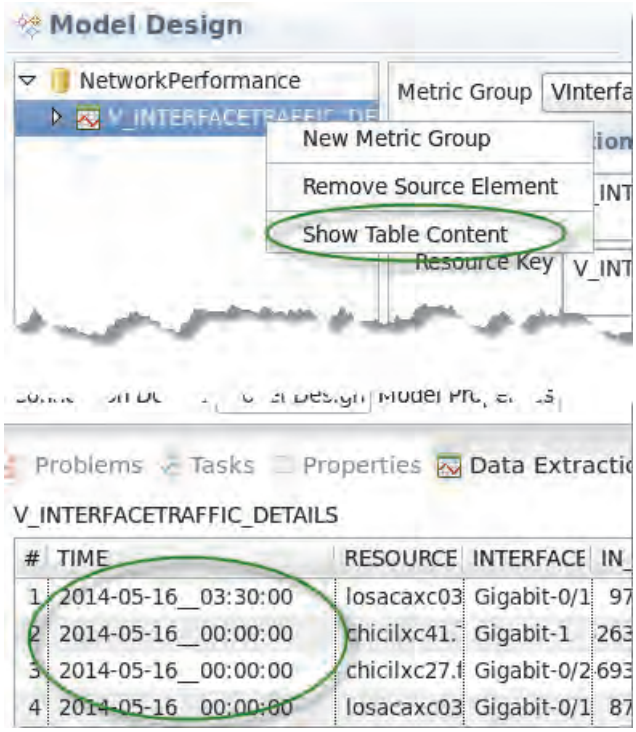
   b. Right-click **NetworkPerformance** database, and select **Synchronize Schema.**



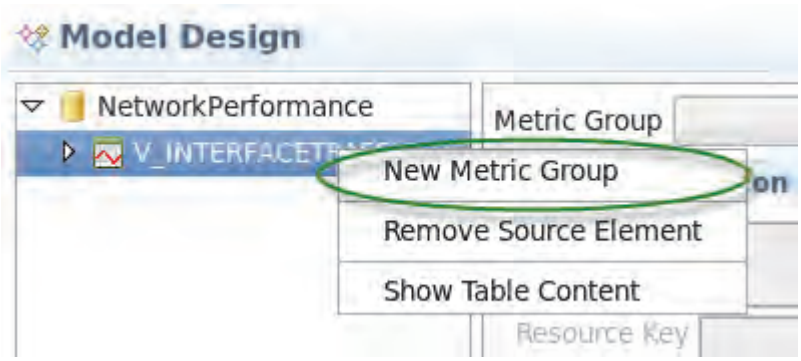
c. When prompted, add only **Table V_INTERFACETRAFFIC_DETAILS.** Click **OK**.



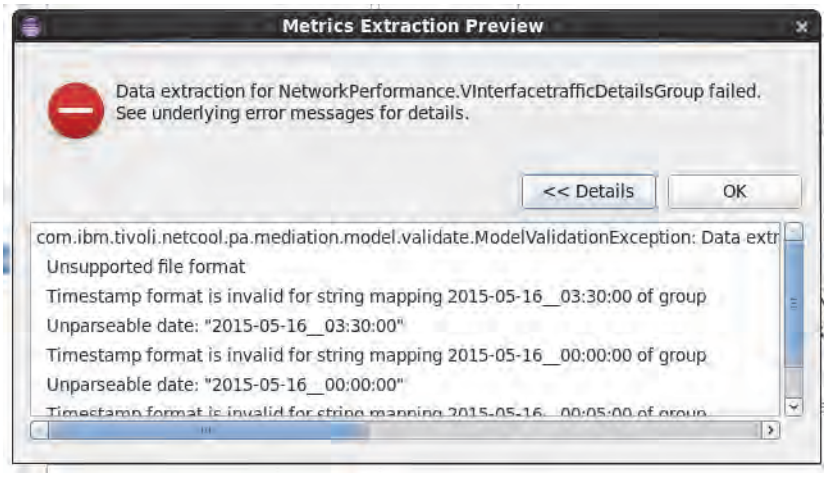
7. View the raw table data by right-clicking the **V_INTERFACETRAFFIC_DETAILS** table and selecting **Show Table Content**. Note that the time is in a human-readable form. However, you must use the mediation tool to parse this string and get, for example, the year, month, or day.



9. Right-click the new table and select **New Metric Group**. Use the default name, and click OK.

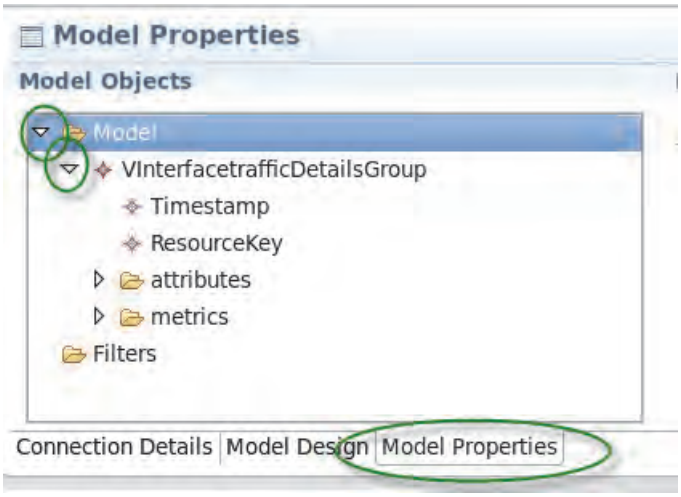


10. Preview the data, and note the error. Select **PreviewExtraction**. Note the error that is generated. Click the details button to get more information. Click OK to close the error message.

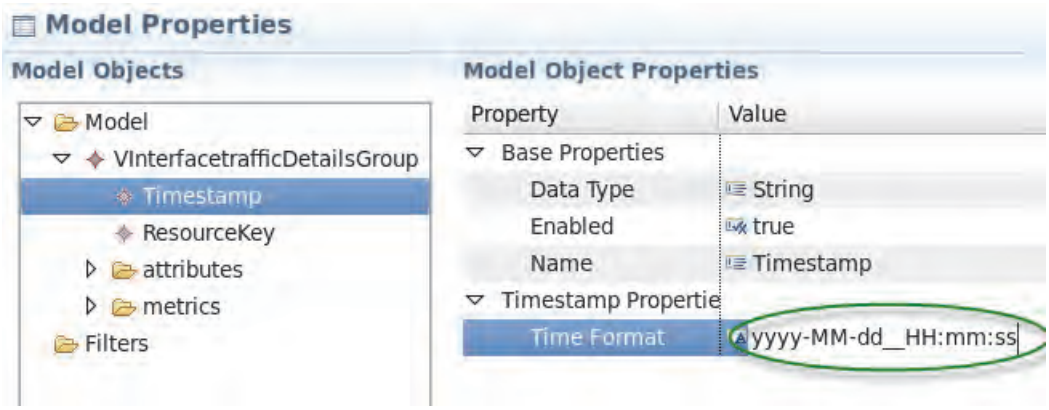


A problem exists with the time stamp format.

11. Create a time format for the time stamp to interpret the string.
    a. To define the format of the time stamp, select **Model Properties**. Expand **Model** > **VInterfacetrafficDetailsGroup.**



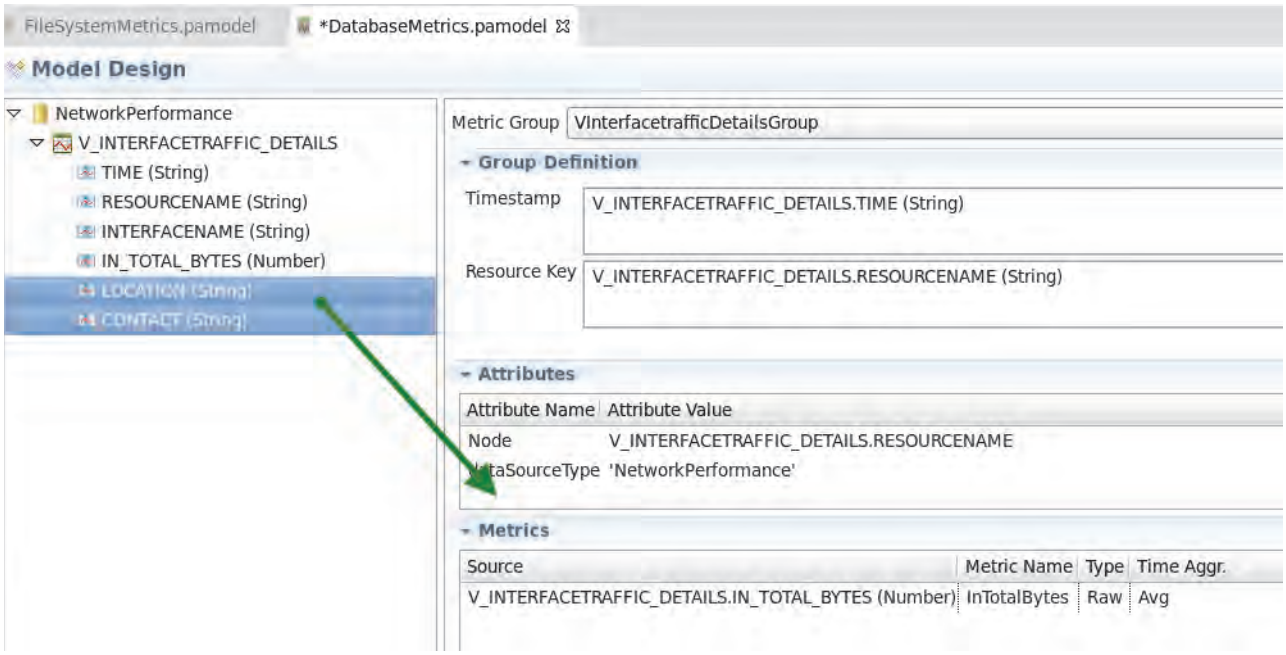
    b. Select **Timestamp**, and enter the following time stamp format in the **Time Format** field: `yyyy-MM-dd__HH:mm:ss (note there are two "_" between dd and HH)`



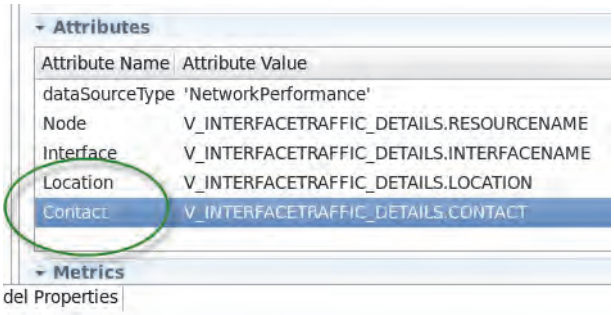
Important: Java time format uses uppercase and lowercase characters for various patterns. The lower case *s* references seconds. An uppercase *S* references milliseconds.

12. Add the **LOCATION** and **CONTACT** fields to the Attributes list

    a. Selecting the **LOCATION** and **CONTACT** fields and drag them to the Attributes window.

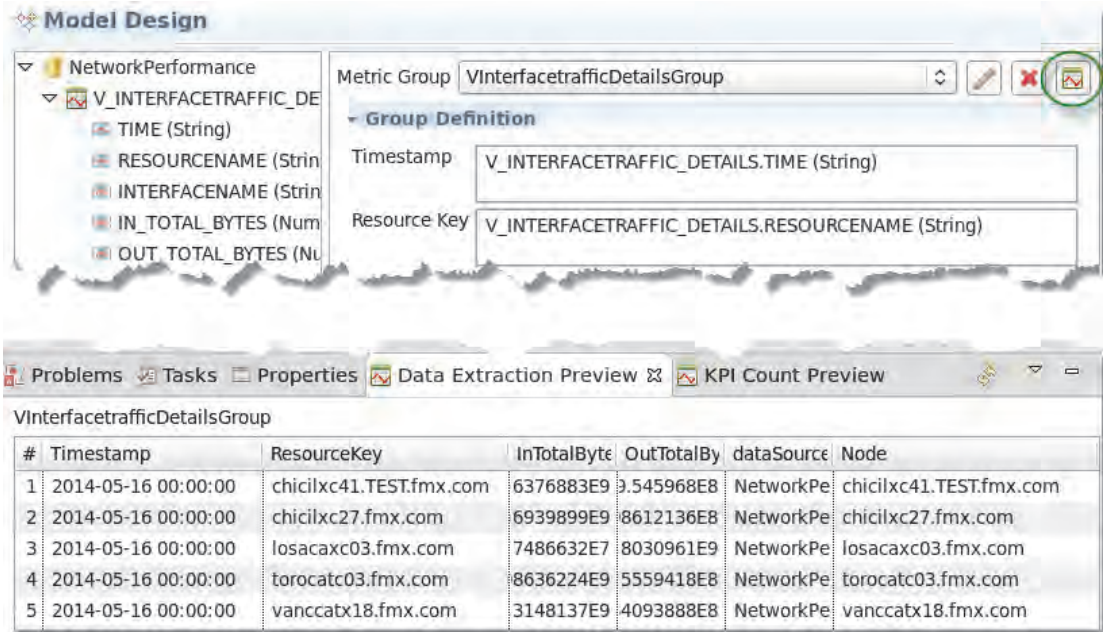


    b. Rename the newly added attributes by selecting the Attribute Name and type in the names of **Location** and **Contact**.



Note: The naming of these attributes is important as they are included in the alarm sent to OMNIbus and can be used for display in the Predictive Insights UI. Slight misspellings of these attribute names can create problems.

13. Revalidate extraction, and check for accuracy. Note the addition of the Location and Contact column.



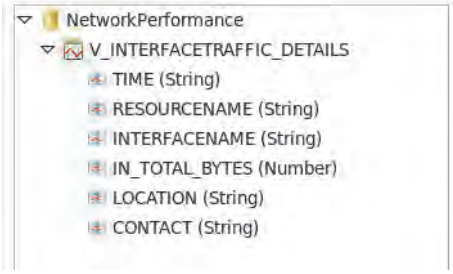
14. Save the data source.

Exercise 7 Creating a custom resource name

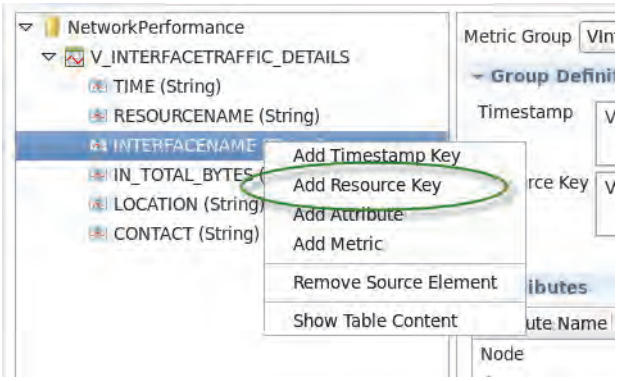
You might have to create a custom resource name to manage a subcomponent of a particular resource; for example, a server or router with multiple interfaces. In this case, you can create a new resource, which is the combination of the resource name and an interface. These two items must be included in the raw data stream. In this exercise, you learn how to create this resource key and understand the requirements.

Complete the following steps:

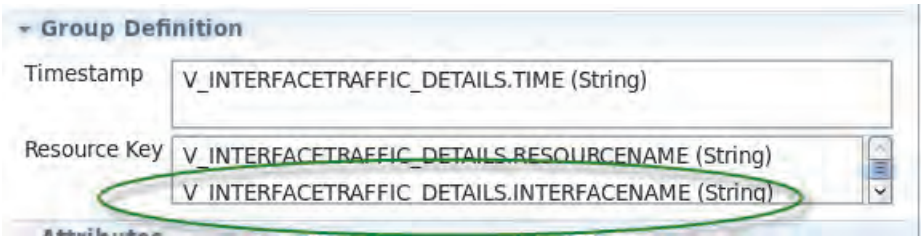
1. In the VInterfacetrafficDetailsGroup, add the **INTERFACENAME** as an extra metric key.
   a. Expand **V_INTERFACETRAFFIC_DETAILS** to view the schema details.



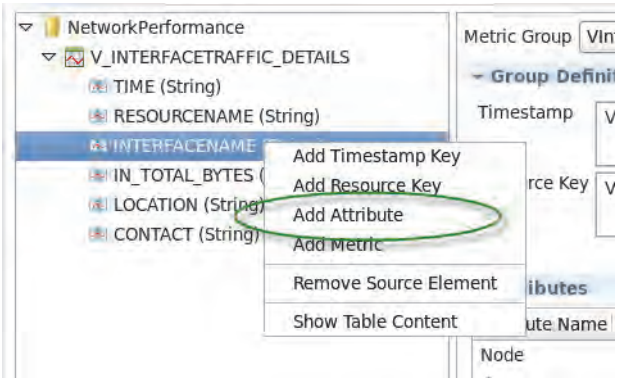
   b. Right-click the **INTERFACENAME** node and select **Add Resource Key**. You combine two items from the data table, RESOURCENAME and INTERFACENAME, to create a key.



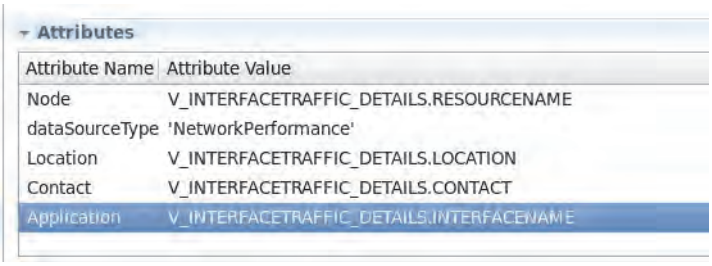
   c. Note that the additional key is added to the **Resource Key**.



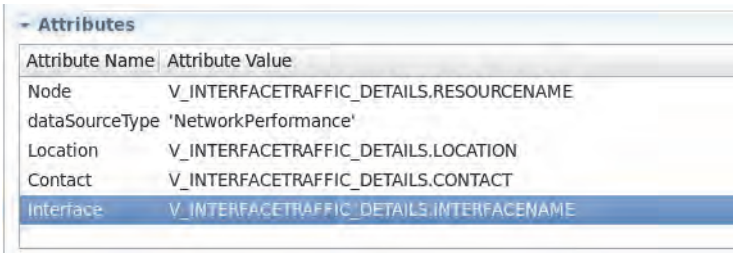
2. Add **INTERFACENAME** as an extra attribute. Remember that each resource key must have a corresponding attribute. Change the attribute name from application to interface.
   a. Right-click the INTERFACENAME node, and select **Add Attribute**.



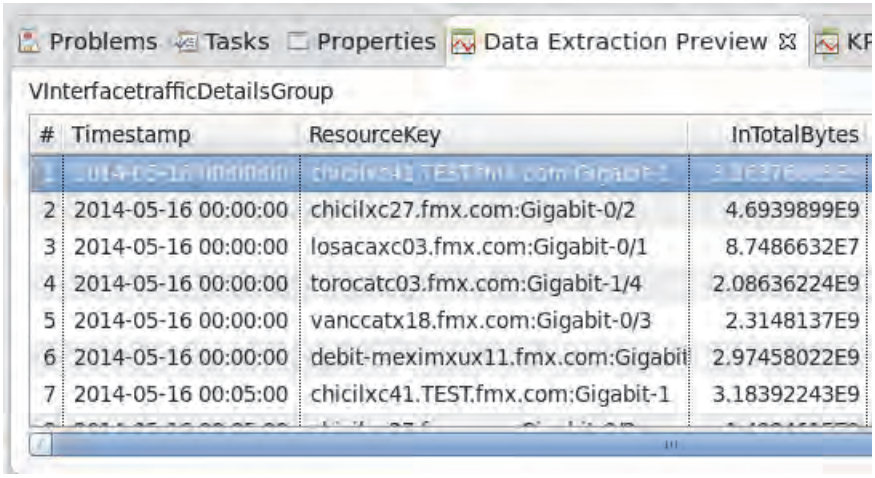
   b. Scroll down, and note the addition of V_INTERFACETRAFFIC_DETAILS.INTERFACENAME as a new attribute.



   c. Make the attribute name more meaningful by selecting it and changing it from **Application** to **Interface**. Interface is not a default selection in the selection list. You can create a new attribute name by entering the name using the keyboard.



3. Click **Preview the extraction**. Review the extracted data. Note that the **ResourceKey** has the interface number that is appended to it, and an **Interface** column was added to your extracted data.

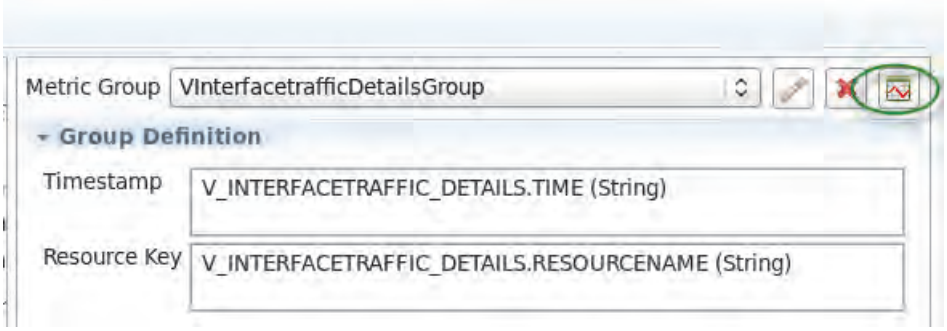


4. Save the data source.

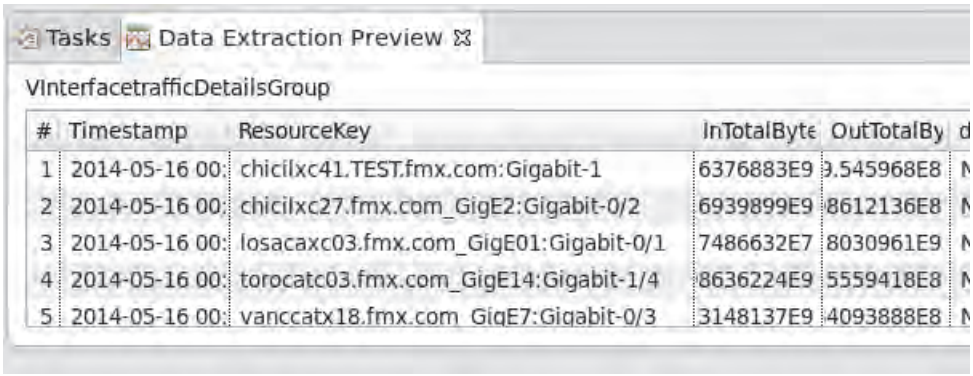
Exercise 8 Filtering unwanted resources

Sometimes, you find resources that you do not want in your analysis. These resources are typically associated with test devices. Their inclusion in Predictive Insights can lead to false alarms because they are used for nonproduction activities. In this exercise, you filter them from the data model. You use the VinterfacetrafficDetailsGroup metric group to complete this exercise.

1. Select the **VinterfacetrafficDetailsGroup** metric group, and preview its data. Note the TEST server.
   a. Click the **Preview extraction** button to preview the **VinterfacetrafficDetailsGroup** data.

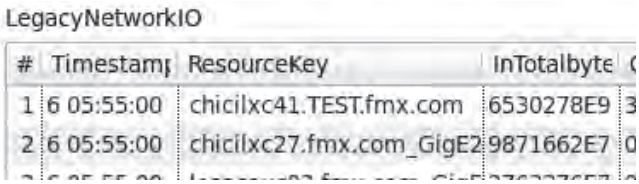


Note the **Data Extraction Preview** in the lower pane.

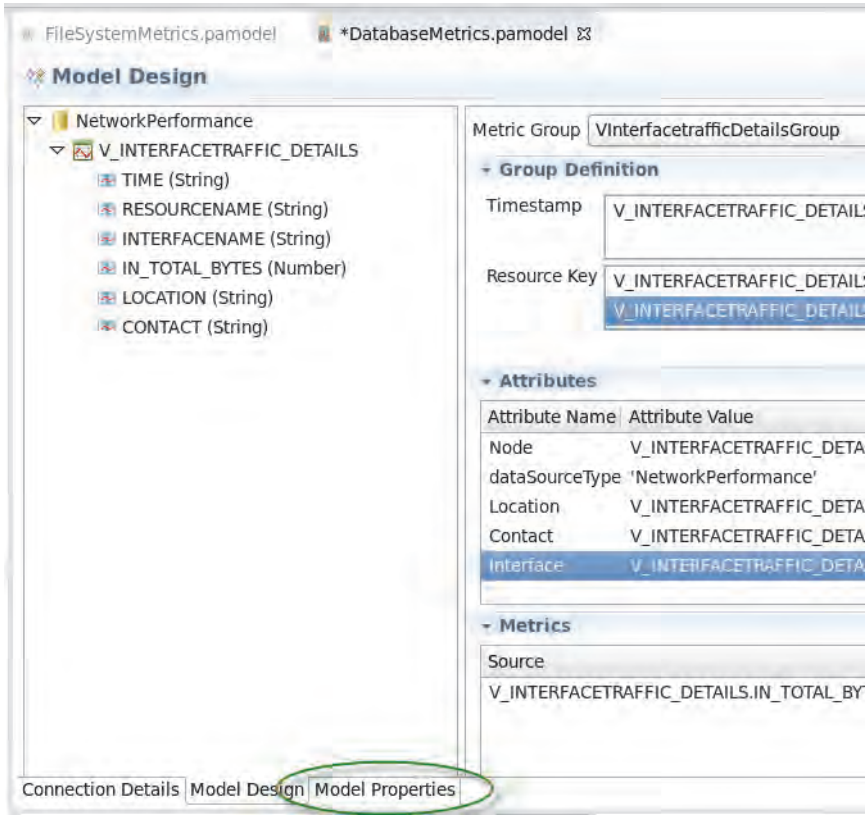


Note the TEST server in the Data Extraction Preview. You may need to scroll down through the data to see this resource

b. Check the device, **chicilxc41.TEST.fmx.com**.

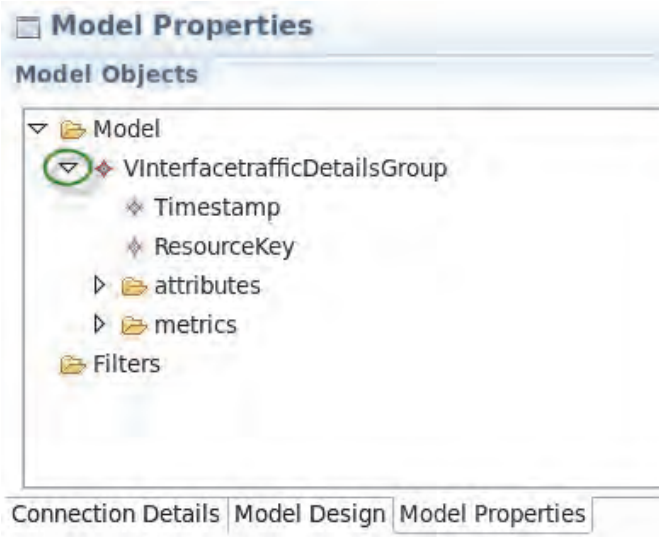


2. On the **Model Properties** tab, place a resource filter to exclude any server with *TEST* in its name.
   a. Click the **Model Properties** tab.

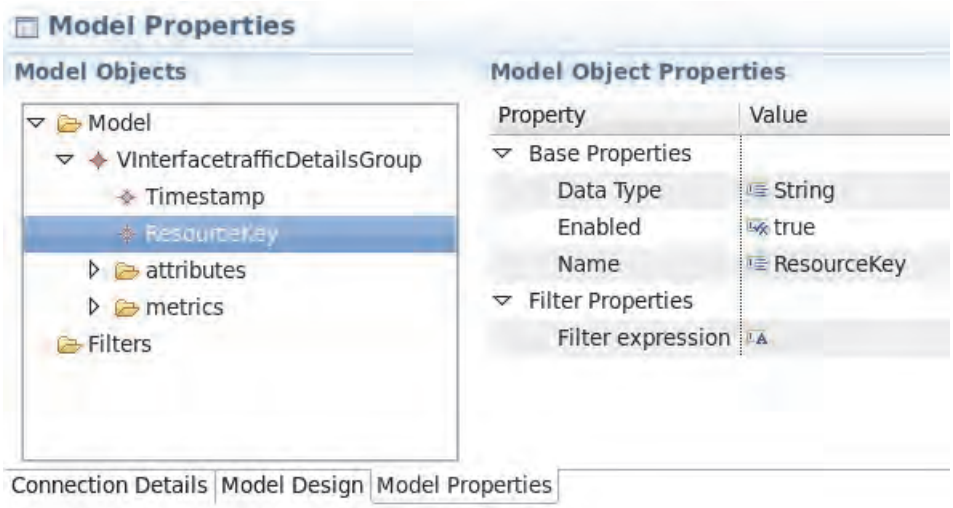


   b. Expand the **Model** folder.

c. Expand the **VInterfacetrafficDetailsGroup** metric group.

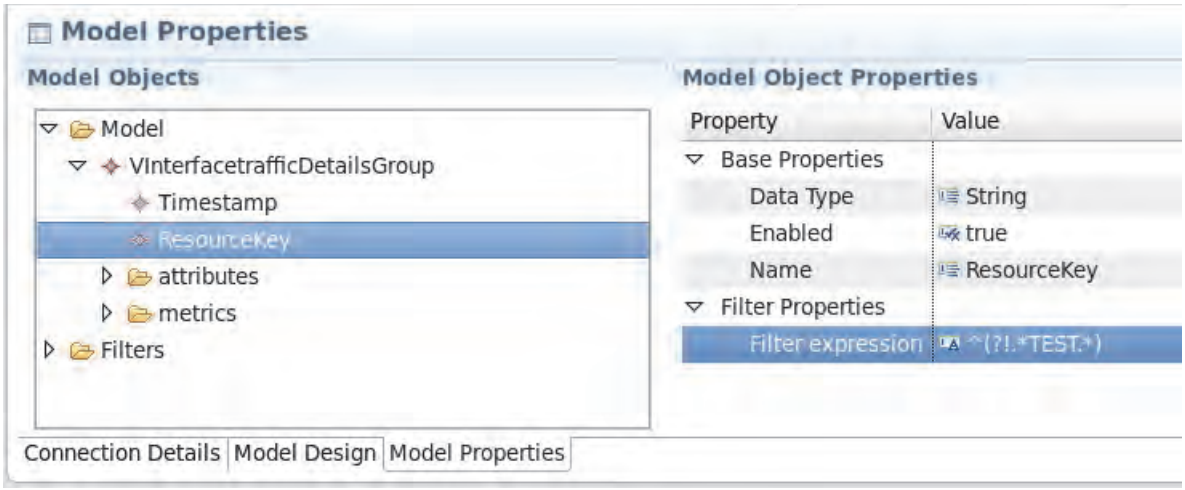


d. Select the ResourceKey, and place a filter to exclude *TEST* devices. Select the **ResourceKey** node under **VInterfacetrafficDetailsGroup**.



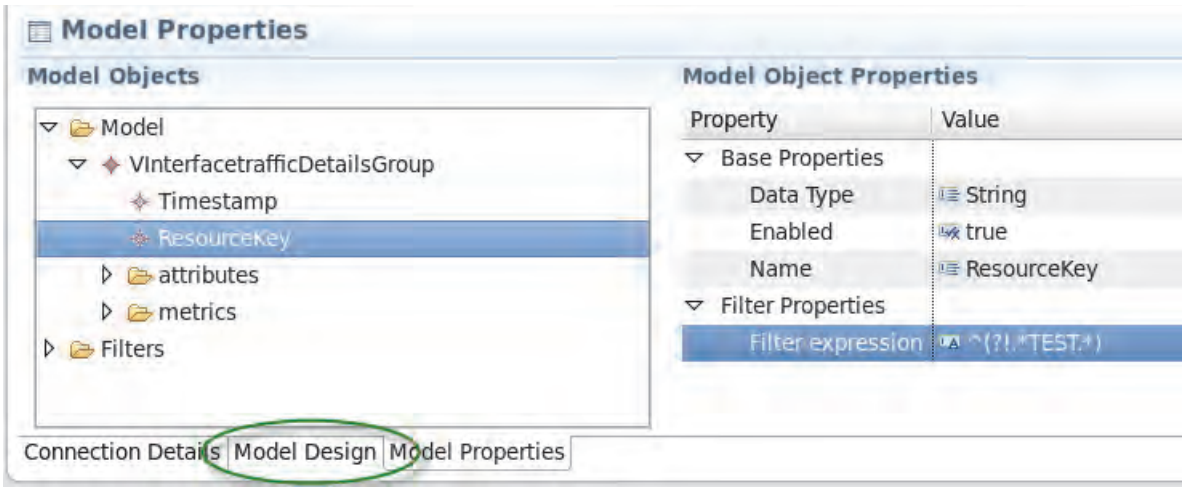
e. In the **Filter expression** field, enter the following regular expression to exclude TEST devices:

```
^(?!.*TEST.*)
```

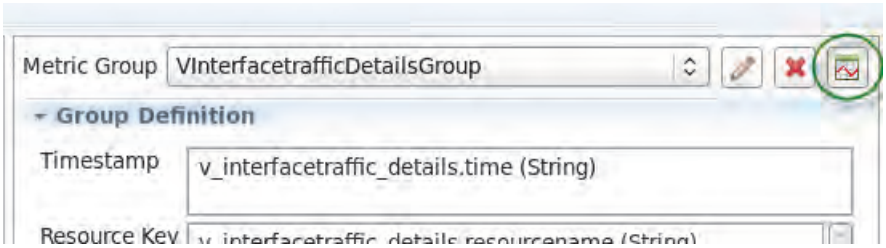


3. Preview the data to confirm that TEST devices are removed.

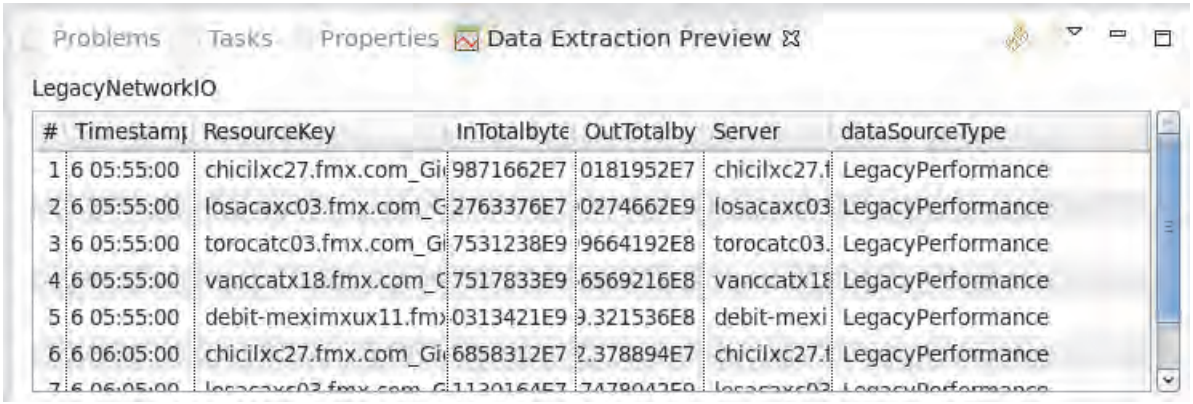
a. Return to the **Model Design** tab.



b. Click the **Preview the extraction** button.



Note the removal of the TEST device from the extracted data. Scroll through the data to confirm its deletion.



4. Save the changes.

Exercise 9 Adding a new JDBC driver

Currently, only seven drivers are defined in the data mediation client. Only one of them has a JDBC software driver included in the installation (that is, DB2.) If you find a data source that is not a DB2 database, you must download and add the vendor's JDBC Java file to the server and mediation client class paths. If the database is not one of the supported seven, you must complete the additional step to add it to the data mediation pull-down list.

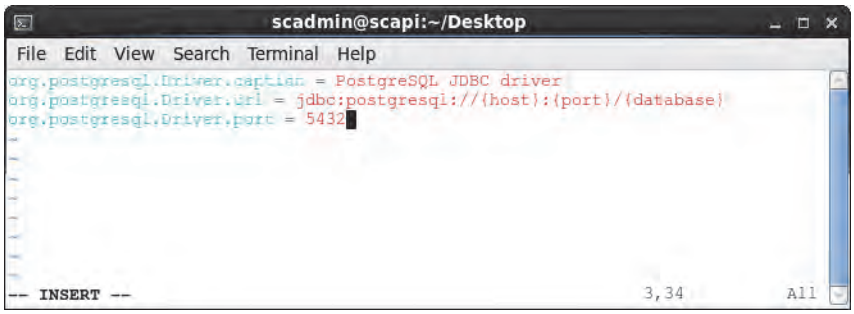
In the following two exercises, you connect to a PostgreSQL database that contains performance data. To enable this connection, you must modify the data mediation client by adding the PostgreSQL JDBC driver to the list so that you can connect to and model the data. Second, you must add the JDBC driver to the mediation client and server so that it also can connect to and extract data from this database.

Adding new drivers to the mediation client requires the creation of a **driver.properties** file that must exist in the workspace created when you started the mediation client (the default location in the user's home directory.) You add specific information about each database driver you want to add to this file. You then need to configure the mediation client and the analytics server to be able to find the JDBC jar file associated with the database.

1. Create the **driver.properties** file for the mediation client.
   a. The workspace for the current project is in **/home/scadmin/workspace**
   b. In a device terminal, create a drivers.properties file with the following command
      `touch /home/scadmin/workspace/driver.properties`

2. Configure the driver.properties file for using the PostgreSQL JDBC client
   a. Open the drivers.properties file with your preferred Linux editor (vi, gedit, etc.)
   b. Add the following three lines to the file

```
org.postgresql.Driver.caption = PostgreSQL JDBC driver
org.postgresql.Driver.url = jdbc:postgresql://{host}:{port}/{database}
org.postgresql.Driver.port = 5432
```

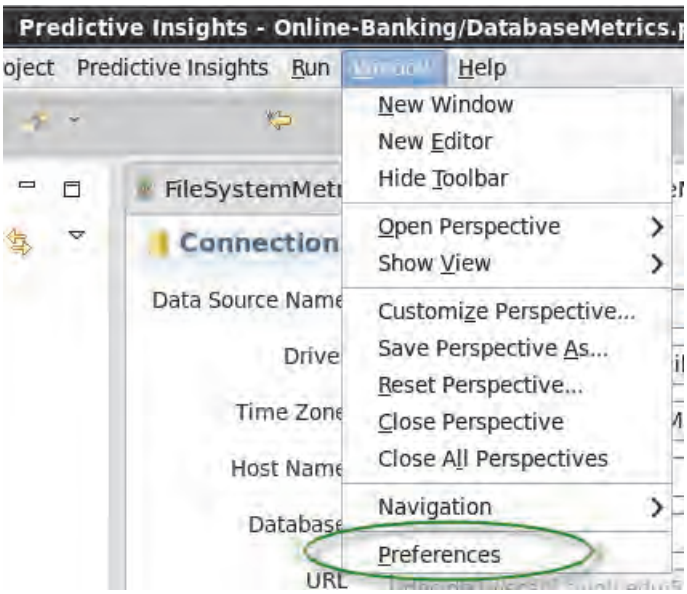


   c. Save the file.

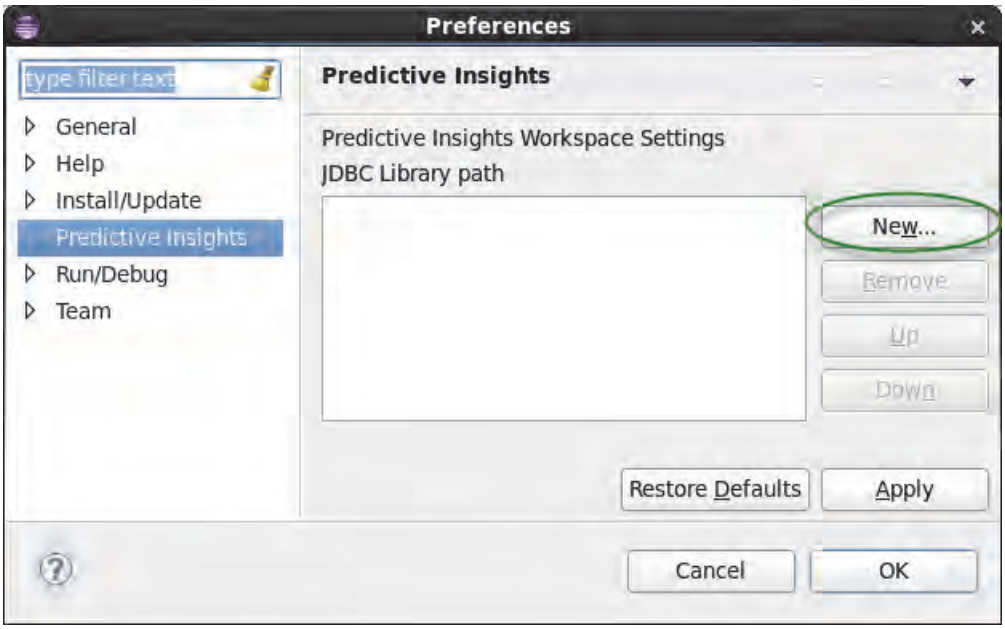
3. Configure the mediation client to point to the PostgreSQL JDBC jar file.
   a. If the mediation client is not already open, then start it with the following command.

```
/opt/IBM/scanalytics/analytics/bin/mediationtool.sh
```

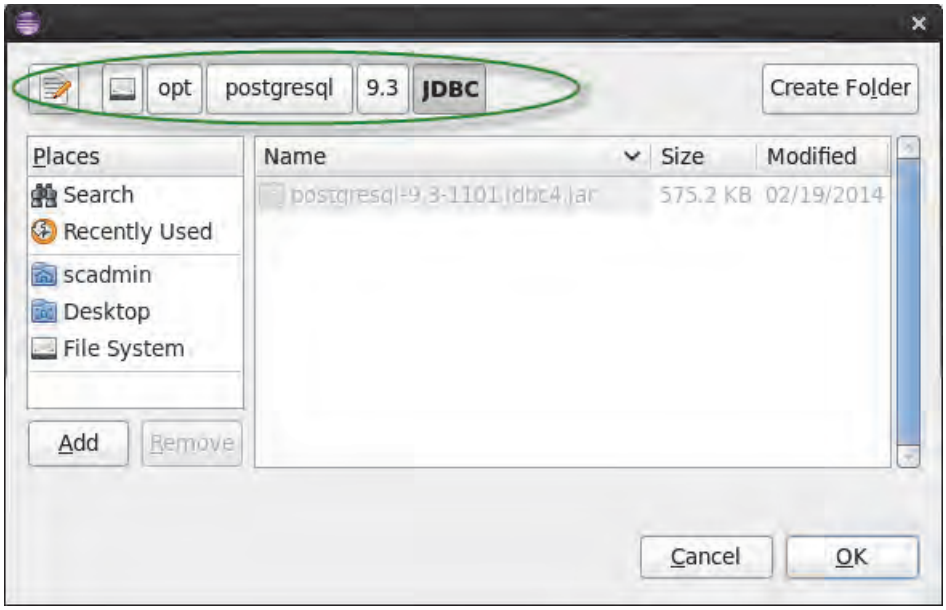
   b. Select **Window** > **Preferences**.



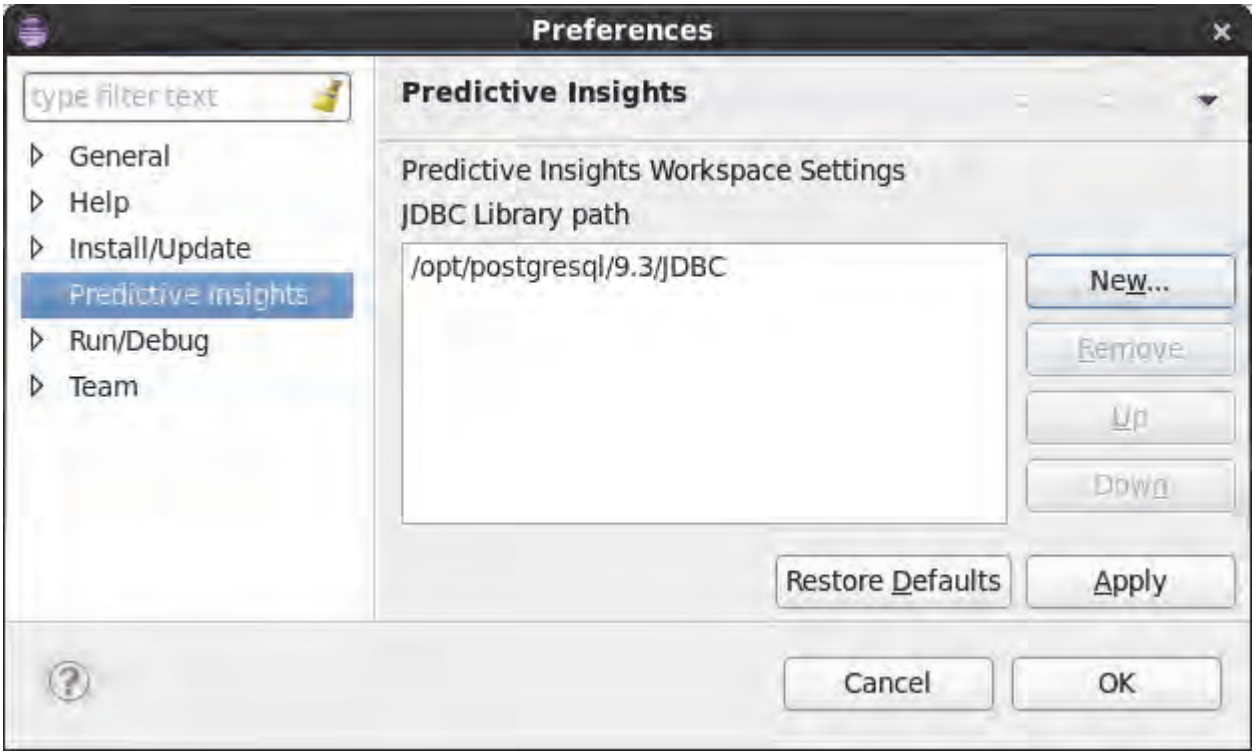
c. In the Preferences window, select **Predictive Insights** and click **New** to add a JDBC library path.



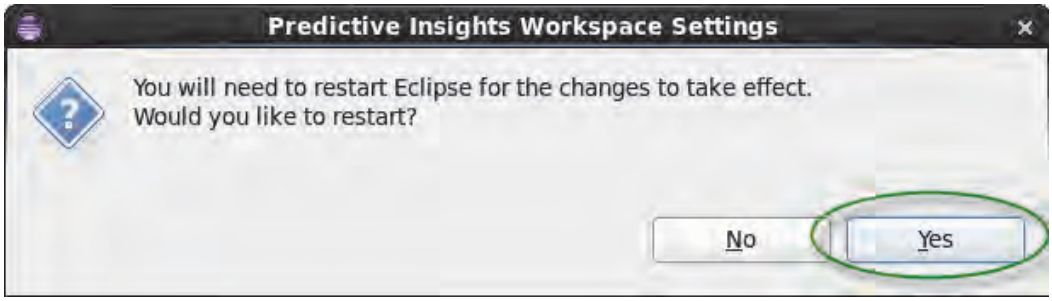
d. In the file manager that opens, locate **/opt/postgresql/9.3/JDBC** and click **OK**.



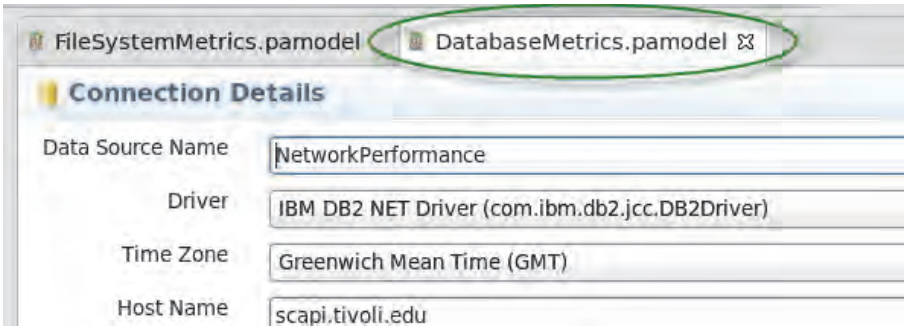
e. Click OK in the Preferences window.



f. When prompted, restart Eclipse to make the changes.

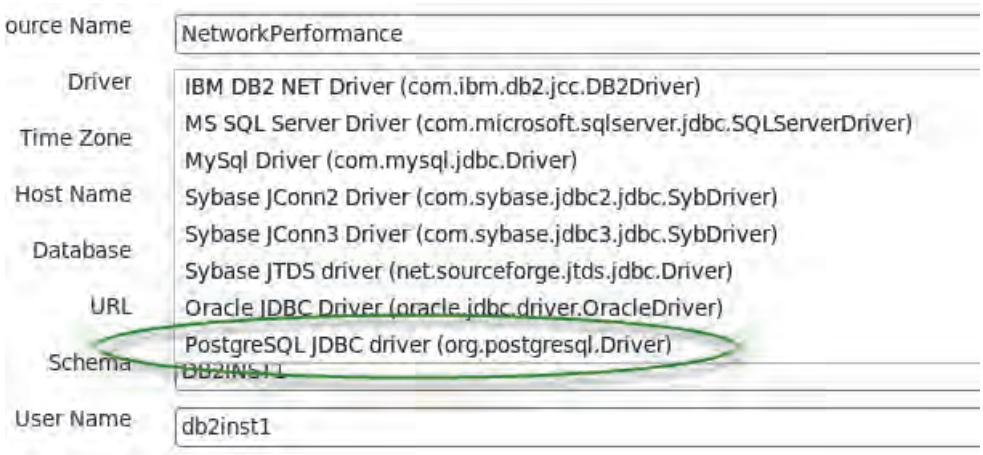


4. Check the mediation client for a new list for the PostgreSQL database connector.
   a. When Eclipse restarts, find and click the **DatabaseMetrics.pamodel** tab. It might already be displayed.



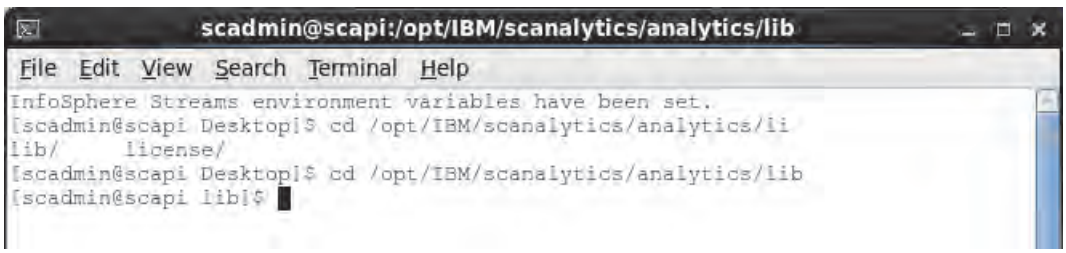
b. Click the **Driver** list to see if the PostgreSQL driver is added. Validate that the IBM DB2 NET driver is still defined for this data source.

Important: Select only the IBM DB2 NET driver. *Do not* select a different driver.



5. Add the vendor's JDBC JAR file to the class path of the mediation server. Go to the **$TASP_HOME/lib** directory and create a link to the JDBC driver.
   a. In a terminal window, change directory to **/opt/IBM/scanalytics/analytics/lib**.

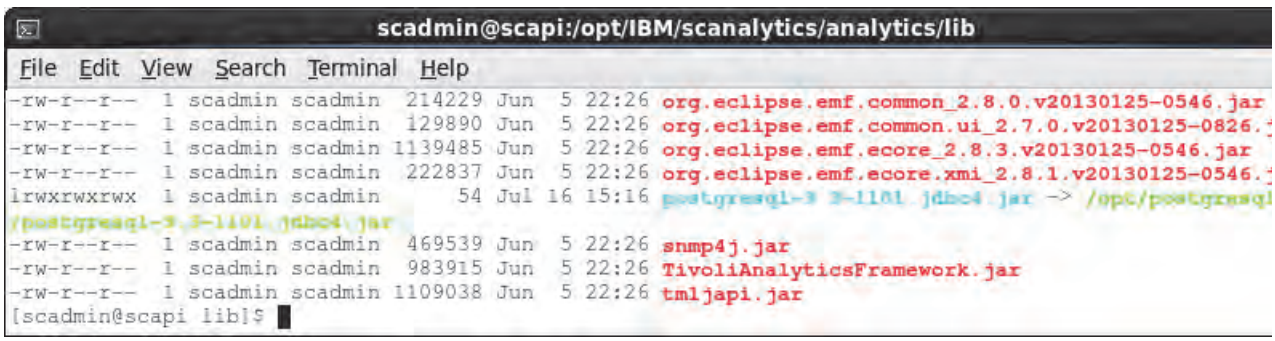
      ```
      cd /opt/IBM/scanalytics/analytics/lib
      ```



   b. Create a symbolic link to the **postgresql-9.3-1101.jdbc4.jar** file in the **/opt/postgresql/9.3/JDBC** directory.

      ```
      ln -s /opt/postgresql/9.3/JDBC/postgresql-9.3-1101.jdbc4.jar .
      ```

   c. Validate that the link was created with a directory listing.



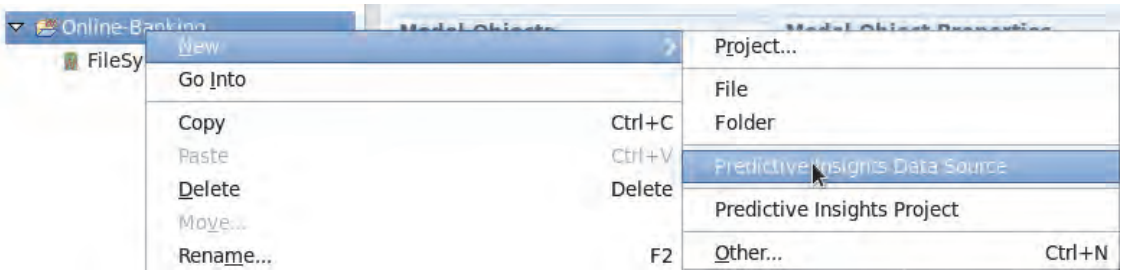
Exercise 10 Modeling the PostgreSQL data source

Like with the earlier DB2 data source, you connect to the PostgreSQL database and synchronize its tables. However, you need to remember that this data source has a **skinny schema**. The connection details are as follows:

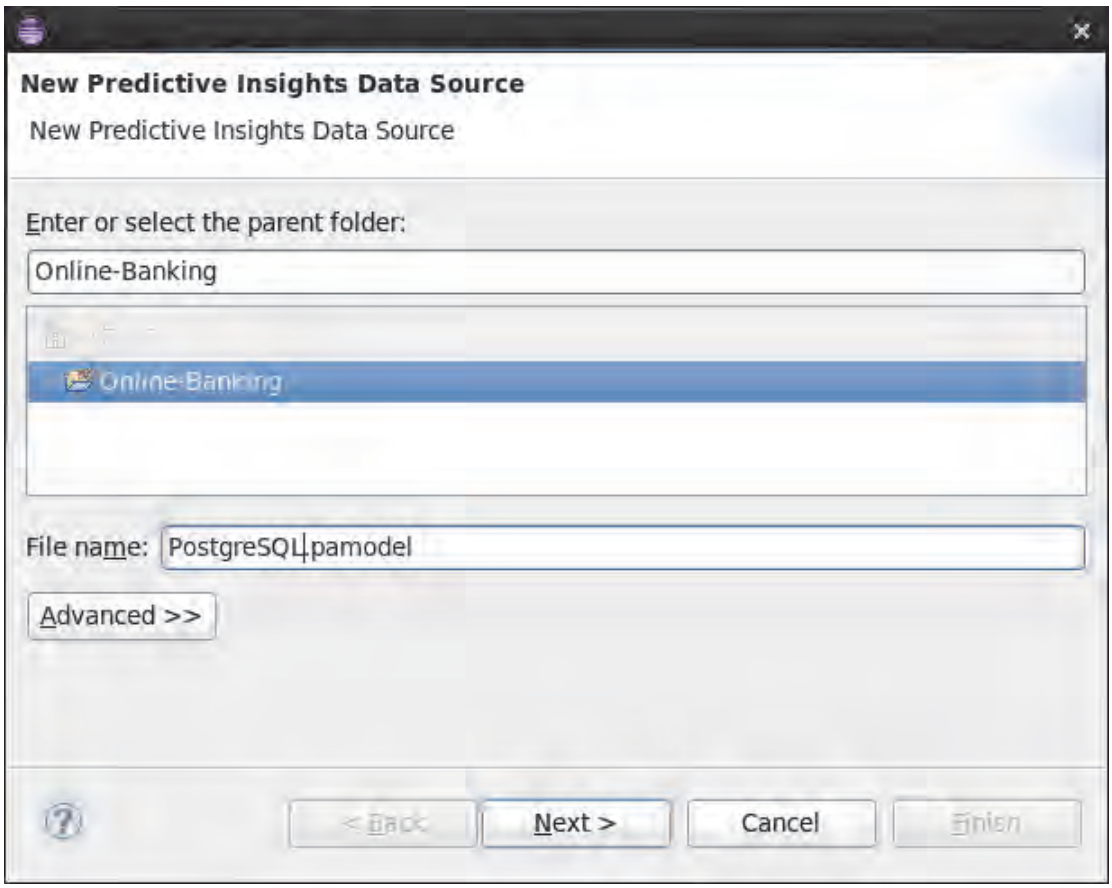
- Host name: **scapi.tivoli.edu**
- Database: **PERFMONITORDB**
- Schema: **public**
- User name: **perfmon**
- Password: **object00**

Complete the following steps:

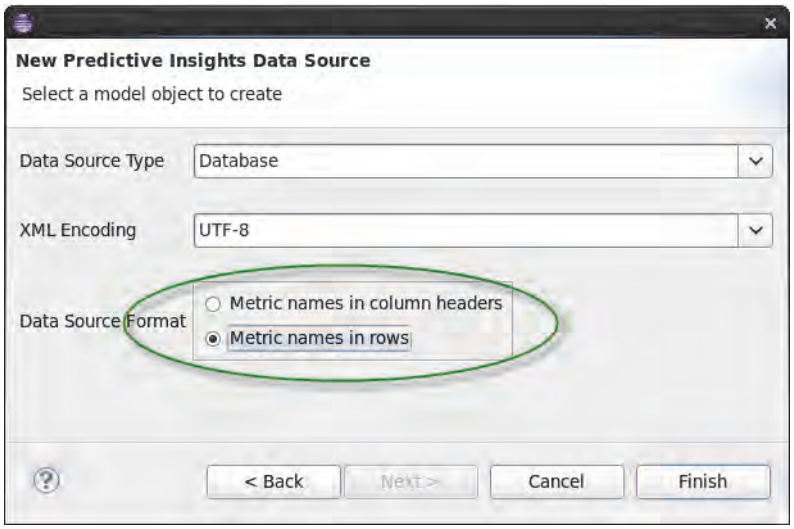
1. Create a database data source and call it **PostgreSQL.pamodel**.
   a. Right-click the **Online-Banking** project, and select **New** > **Predictive Insights Data Source**.



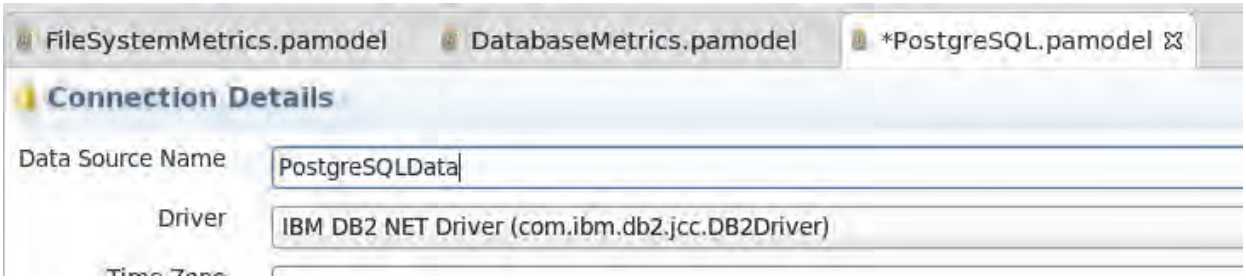
b. In the New Predictive Insights Data Source window, enter the name **PostgreSQL.pamodel**. Click **Next**.



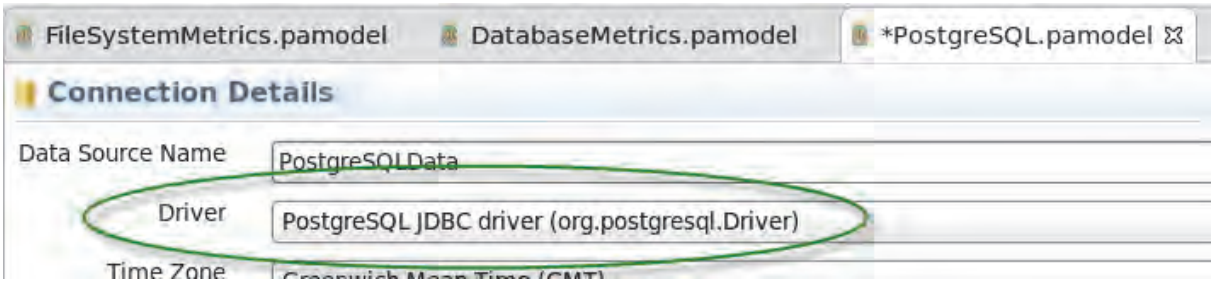
c. From the **Data Source Type** menu, select **Database**. You must select the **Metric names in rows** for the **Data Source Format**. Click **Finish**.



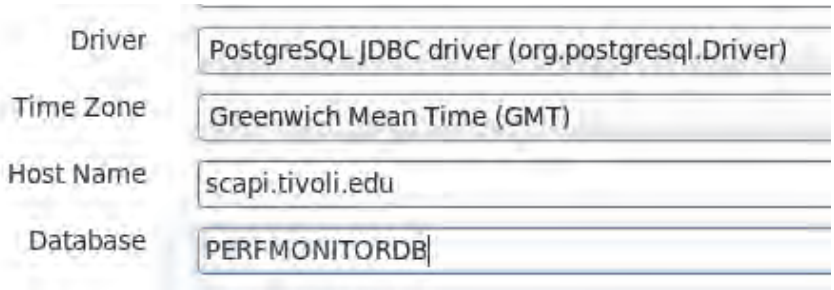
2. Change the data source name to PostgreSQLData. Select the PostgreSQL driver and enter the connection details to the PostgreSQL database, and test the connection.
   a. Click the **Data Source Name** field and change **ITM** to **PostgreSQLData**.



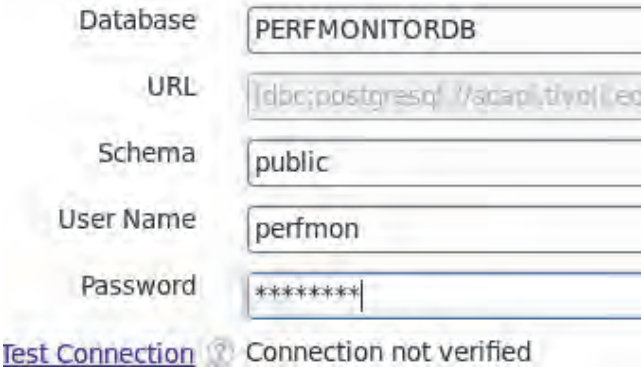
   b. Select the PostgreSQL JDBC driver (org.postgresql.Driver)



   c. Enter the host name of **scapi.tivoli.edu** and database name of **PERFMONITORDB.**

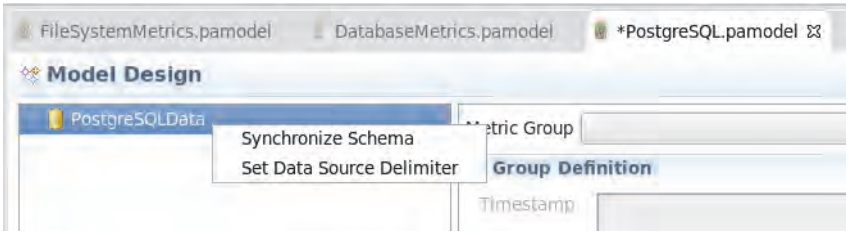


   d. Scroll down and enter the schema of **public**, user name of **perfmon**, and password of **object00.**

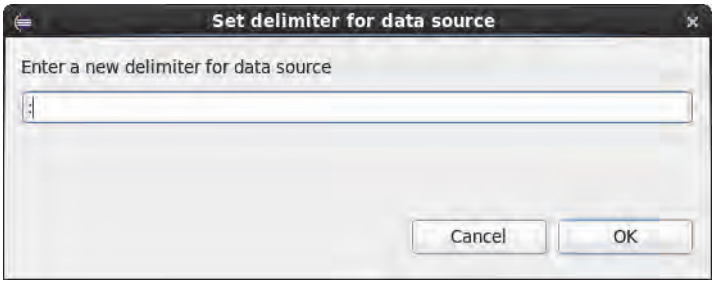


e. Scroll down, and click **Test Connection**.

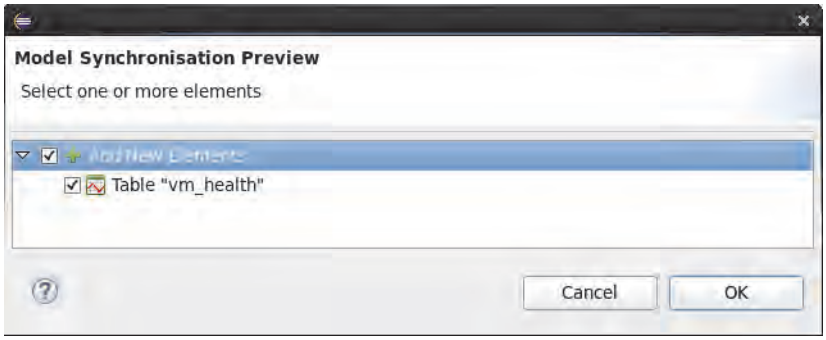
3. Synchronize schema, review the table contents, and add a new metric group called NetworkIO.
   a. Click the **Model Design** tab. Right-click **PostgreSQLData** database, and select **Synchronize Schema.**



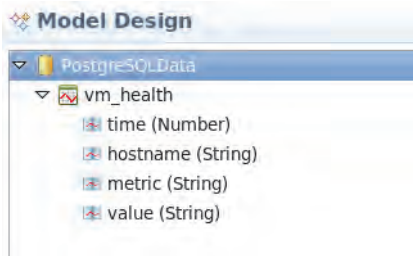
Note: If you selected **Metric names in rows for the Data Source Format**, the Mediation tool queries the metric name column in the first 1,000 rows of data and adds each unique metric name to the Metrics field. If the query finds a *colon in a metric name*, the Mediation Tool displays an error stating that a metric name is invalid because it contains a colon, which is the default data source delimiter. You must change the default data source delimiter to one or more characters that is not in any metric name. To change the default data source delimiter, right click the data source icon, which is the top level of the data source tree view, and click Set Data Source Delimiter.



b. Add the table **vm_health**. Click **OK**.

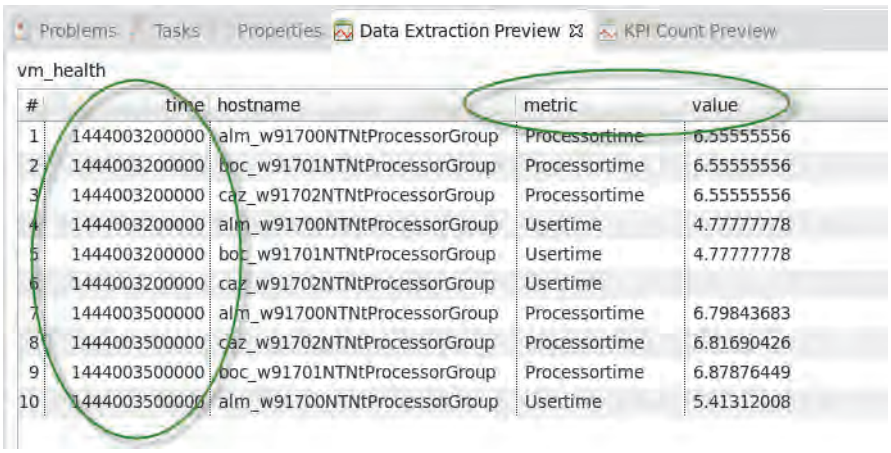


c. Review the table schema for available data by expanding vm_health.



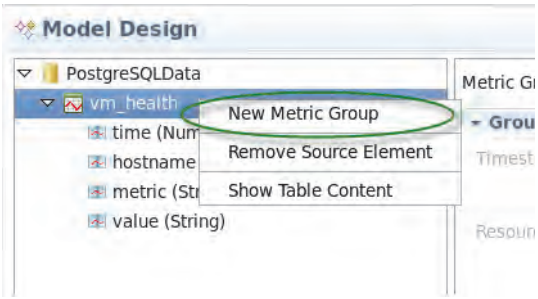
d. Right-click vm_health, and select **Show Table Content**.

e. Review the data that is retrieved from the table. Note the time stamp. It is in UNIX epoch format using milliseconds. Also, note the **metric** and **value** columns. These columns are why this data is considered skinny.

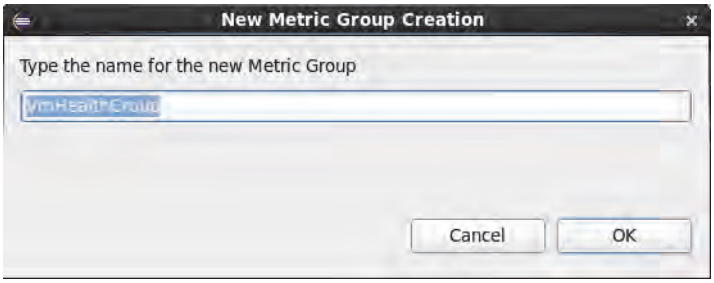


#	time	hostname	metric	value
1	1444003200000	alm_w91700NTNtProcessorGroup	Processortime	6.55555556
2	1444003200000	boc_w91701NTNtProcessorGroup	Processortime	6.55555556
3	1444003200000	caz_w91702NTNtProcessorGroup	Processortime	6.55555556
4	1444003200000	alm_w91700NTNtProcessorGroup	Usertime	4.77777778
5	1444003200000	boc_w91701NTNtProcessorGroup	Usertime	4.77777778
6	1444003200000	caz_w91702NTNtProcessorGroup	Usertime	
7	1444003500000	alm_w91700NTNtProcessorGroup	Processortime	6.79843683
8	1444003500000	caz_w91702NTNtProcessorGroup	Processortime	6.81690426
9	1444003500000	boc_w91701NTNtProcessorGroup	Processortime	6.87876449
10	1444003500000	alm_w91700NTNtProcessorGroup	Usertime	5.41312008

f. Right-click the **vm_health** table, and select **New Metric Group**.

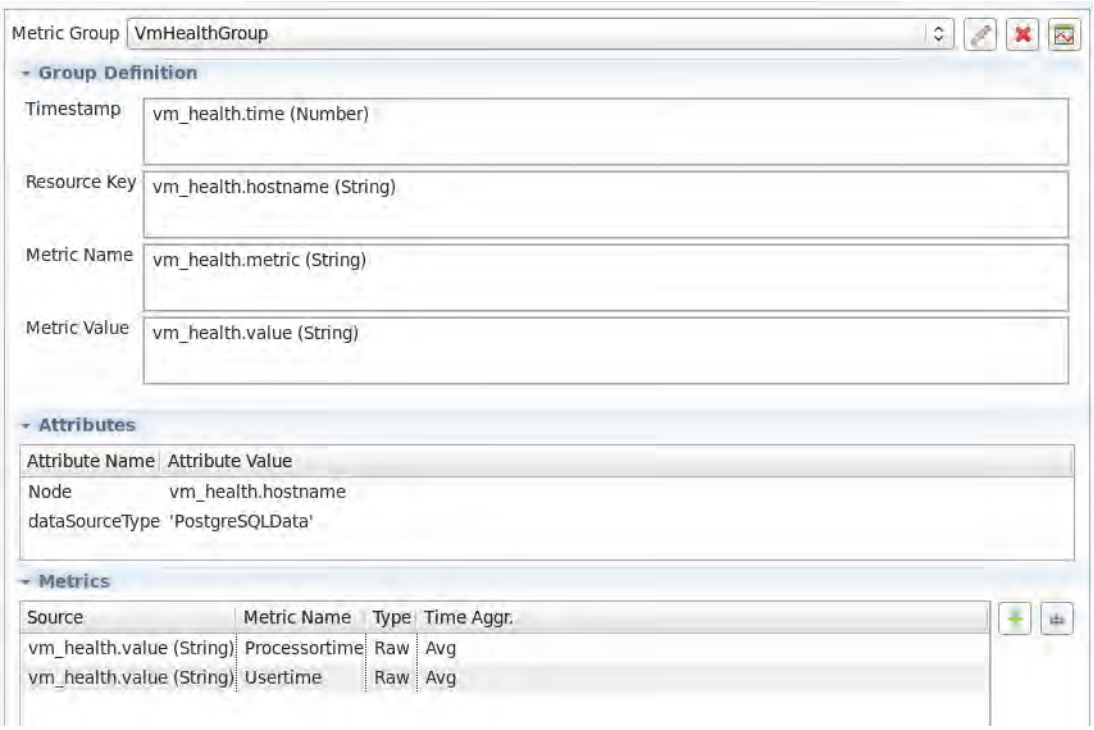


g. Leave the default name of the metric group. Click **OK**.

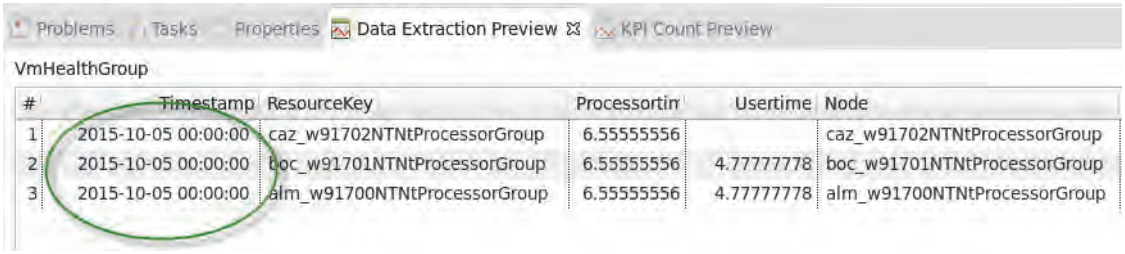


4. Review the time stamp, resource key, metric name, and metric value to ensure they are using the correct column in the database. Preview the data.

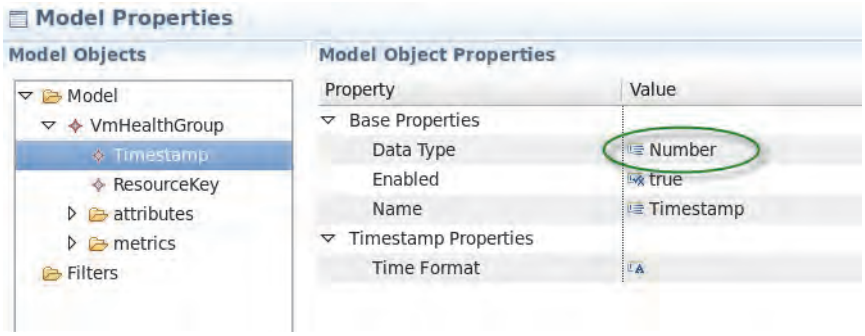
a. Scroll down and review the time stamp, resource key, metric name, and metric value. In some cases you may have to rearrange these values as the mediation tool incorrectly chooses them.



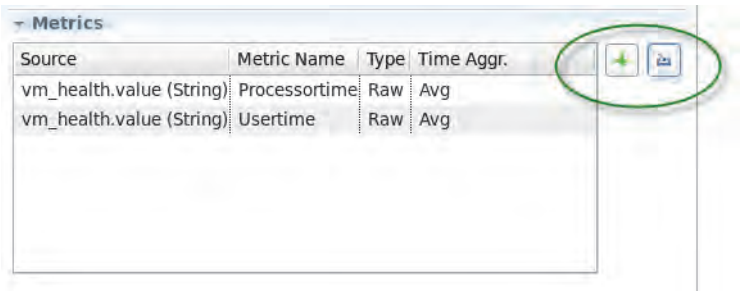
b. Click the **Preview the extraction**. Review the extracted data. Make sure the epoch time stamp is interpreted correctly. If the time stamp reflects a date in 1970, change the data type for the time stamp to **number**.



Note: If the UNIX epoch time is in seconds (a 10-digit number), use **integer** as your data type. If the UNIX epoch time is in milliseconds (a 13-digit number), use **number** as your data type. You make this change on the Model Properties page for the time stamp.



Important: The mediation client attempts to determine the metric names in the data source by finding all the unique names in the metric column of the data source by perusing the first 1000 rows of data. If there are other unique metric names that were not found, you must add them manually. You can add individual metrics or have a file with a comprehensive list of metrics. Refer to online documentation for details.



5. Save the data source.

Exercise 11 Deploying the model

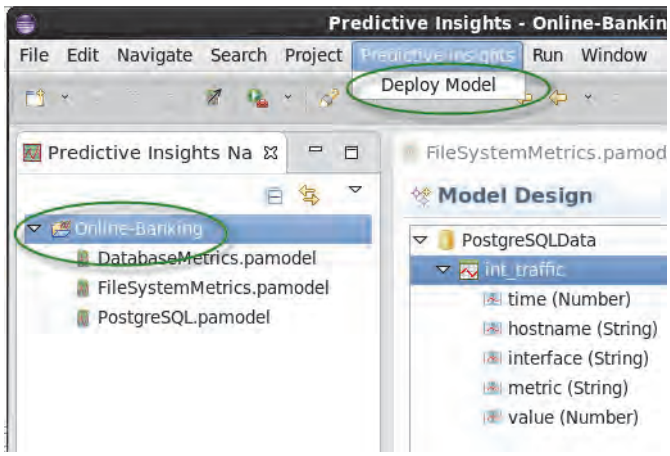
Before you can begin your analysis, you must deploy the model to the analytics server. You developed an XML file that describes how to connect to data sources and how to extract their data. Now you must send this information to the server so that it can extract the data and enter it into its databases.

The connection details are as follows:

- Host name: **scapi.tivoli.edu**
- Database: **SCAPIDB**
- Port Number: **50000**
- User name: **scadmin**
- Password: **object00**

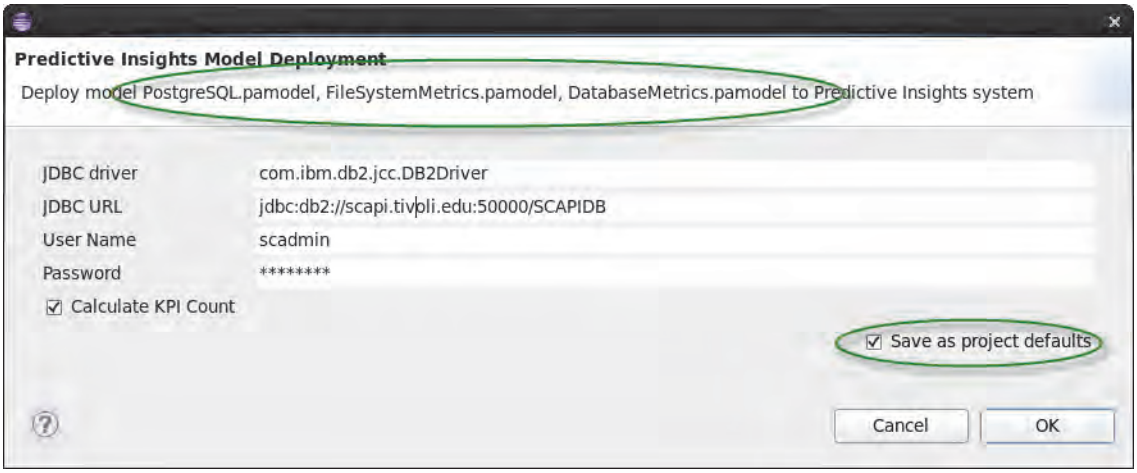
Complete the following steps:

1. Select the **Online-Banking** project, and select **Predictive Insights** > **Deploy Model**.



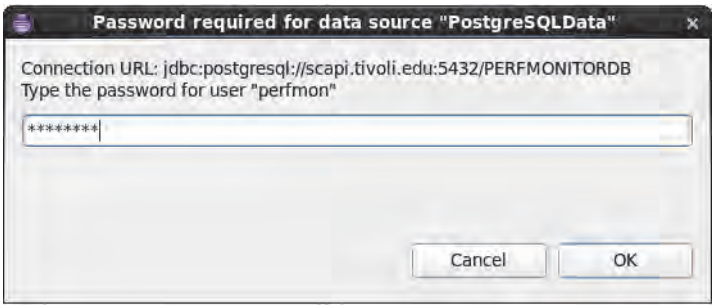
Important: The **Deploy Model** menu selection might sometimes be unavailable when you attempt to deploy. If so, click an item in the GUI to fix this problem. For example, change the Metric Group that you are viewing.

2. Enter the Predictive Insights database location URL and credentials by entering the following information into the Predictive Insights Model Deployment window. *Note that the wizard describes which data sources it is deploying*. Select **Save as project defaults**.
   - Host name: **scapi.tivoli.edu**
   - Database: **SCAPIDB**
   - Port number: **50000**
   - User name: **scadmin**
   - Password: **object00**

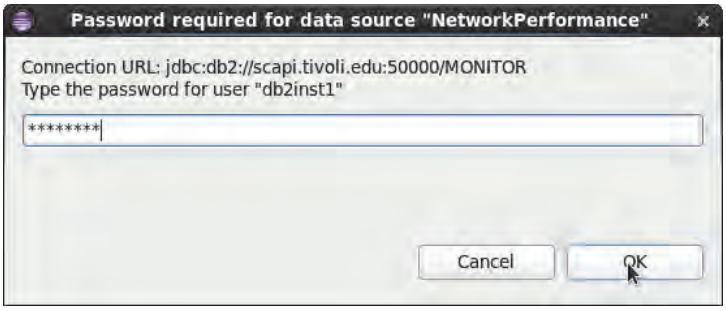


3. Click **OK**.

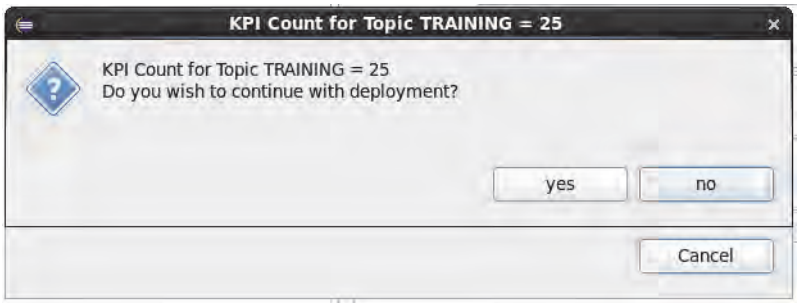
4. Enter the password **object00** for the PERFMONITORDB database. Click **OK**.



5. Enter the password **object00** for the MONITOR database. Click **OK**.

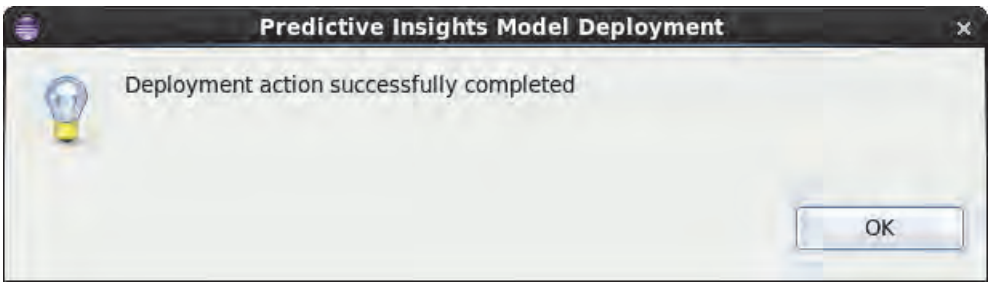


6. Validate that the deployment is successful. The system calculates an estimate for the total number of KPIs that are associated with the topic. If you see 25 KPIs in the **Count for Topic**, click **yes** to continue with deployment.



Important: If you have fewer than 25 KPIs, you might have attempted to deploy only a single data source. If this is the case, then do not continue with the deployment. Return to step one and attempt the deployment again and make note of the names of the data sources you are deploying as noted in step 2.

7. Click **OK** after successful deployment.



Close the mediation tool.

Unit 5 Configuring and operating the server exercises

In this set of exercises, you configure the server for important attributes that determine the aggregation interval that it uses for data extraction and how many weeks it is used for training. You also start the analysis of the historical data that was modeled in Unit 4, Exercise 11 on page 62. You check various logs and file systems to see the progress of the data extraction. Finally, you review the alarms that are generated by Predictive Insights.

Exercise 1 Configuring server settings

In this exercise, you configure the new server. These configuration settings determine important factors about your server. Most importantly is how often data is collected from the data sources. Other configuration settings control how you manage the alarms.

Note: In previous releases of Predictive Insights, you had to set how many weeks of data you wanted to include in the model. In the newer releases, the algorithms used to find anomalies have defined minimum and maximum number of weeks of data needed for the model. Most of these models use 2 weeks for minimums. These changes make it unnecessary for users to configure the number of training weeks needed before a model is generated.

In this exercise, you modify the following settings:

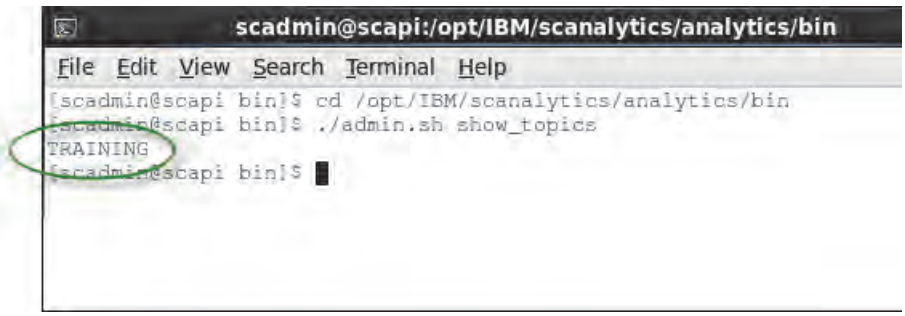
- **system.aggregation.interval**: This aggregation interval is used by this algorithm. Data is normalized to the same interval so that it can be processed by the algorithms. Typically, you set the aggregation interval to the data collection interval. Alternatively you might set it to the smallest common multiple of data collection intervals if several data sources are fed to a single algorithm. Typical values are 5 minutes, 15 minutes, or 1 hour.

The following setting is used only because the data in these exercises is historical. The alarms that are generated typically clear themselves after anomalies in the retrieved data are gone. In these exercises, you want to keep the alarms in the database.

- **system.alarm.autoclear**: If this value is set to *true*, previous alarms that are generated by an algorithm are cleared unless they are generated again during the next aggregation interval. If this value is set to *false*, alarms are never cleared by the system. If set to false, alarms remain in OMNIbus (or the third-party SNMP manager) until they are manually cleared by an operator.

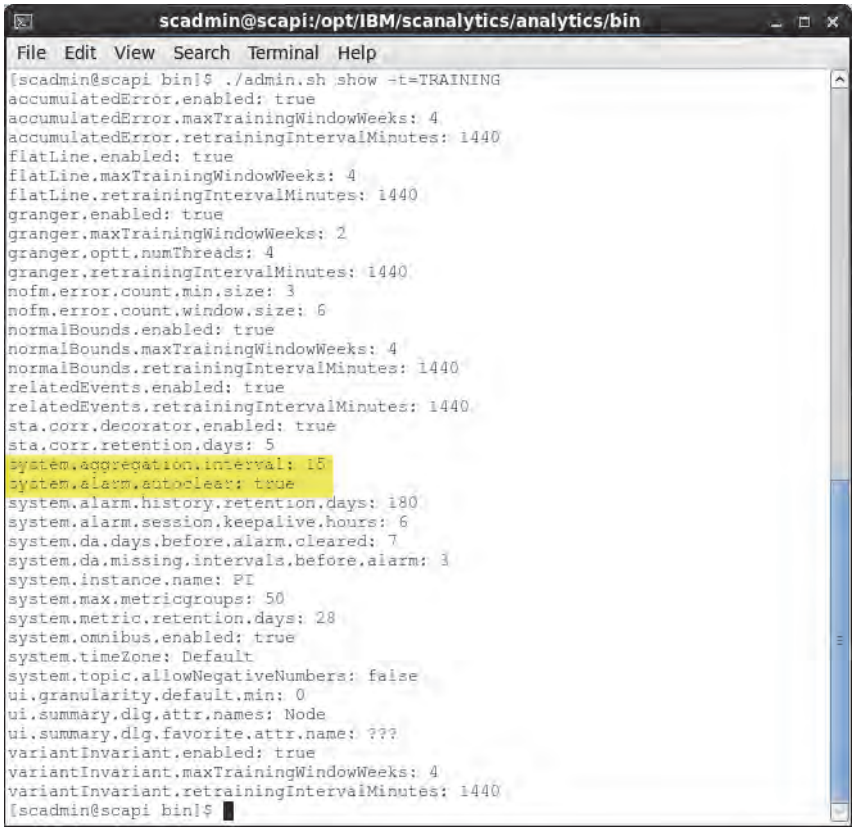
Complete the following steps:

1. Review the name of your topic.
   a. Open a terminal window and change to the bin directory of the analytic server.
      `cd /opt/IBM/scanalytics/analytics/bin`
   b. Enter the following command to see the topic names that are installed on this analytics server:
      `./admin.sh show_topics`



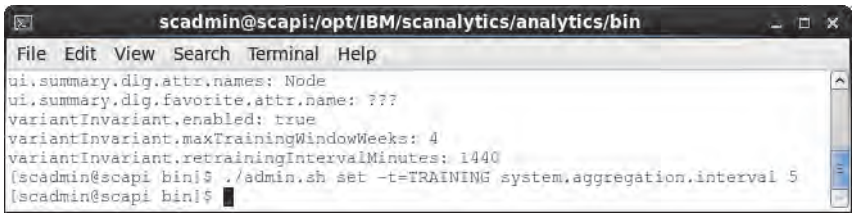
   Note the topic name TRAINING. It is used in the next steps.

2. List all the configurable attributes associated to the TRAINING instance.
   a. Enter the following command
      `./admin.sh show -t=TRAINING`
   b. Note the two attributes that are to be changed



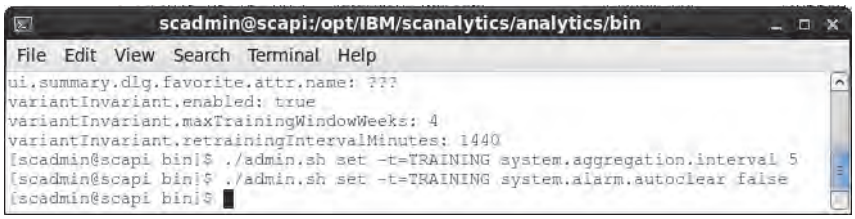
3. Configure the aggregation.interval to 5 minutes. Enter the following command to change the aggregation.interval:

   `./admin.sh set -t=TRAINING system.aggregation.interval 5`



4. Set system.alarm.autoclear to false. Enter the following command to change the alarm.autoclear:

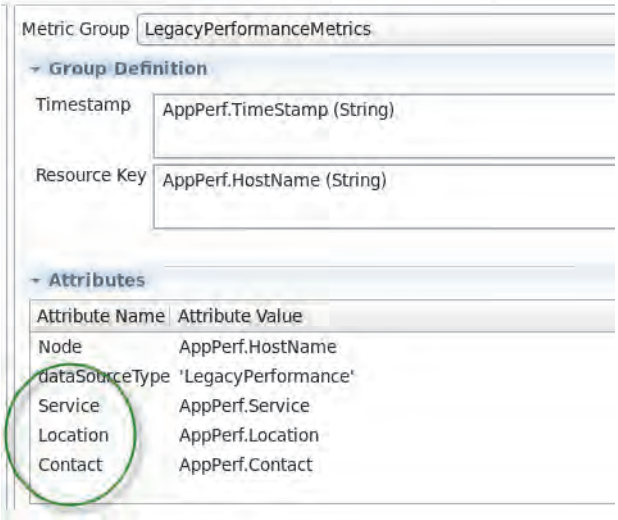
   `./admin.sh set -t=TRAINING system.alarm.autoclear false`



Note: The choice for aggregation interval is based on the fact that the data in both the database and CSV files is collected at 5-minute intervals. If they are collected at 2-minute and 5-minute intervals, you set the aggregation interval to 10 minutes and average the results that are extracted from each data source.

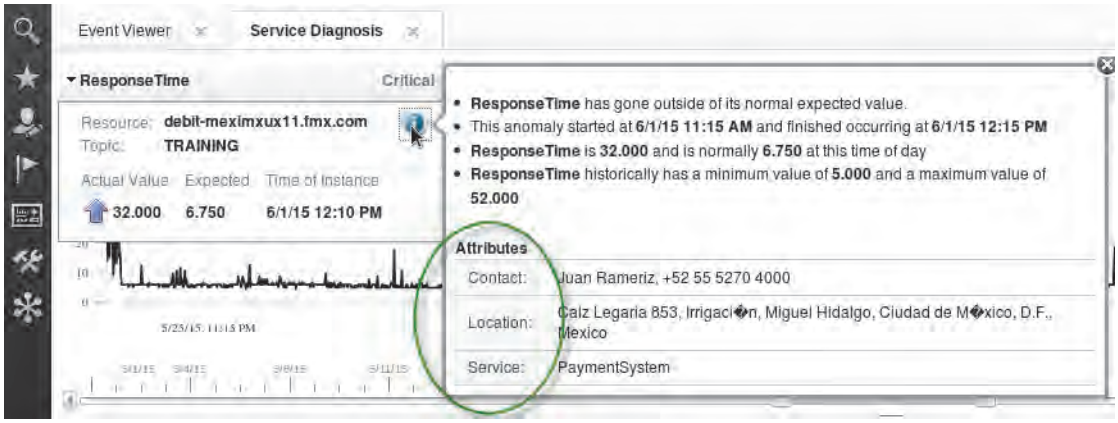
Exercise 2 Configuring the GUI to include useful attributes

When you created your data model, you added additional fields to your model that did not contain metric values. For example, you had attributes denoting the service the server supported, the location of the server, and who to contact if there are issues.



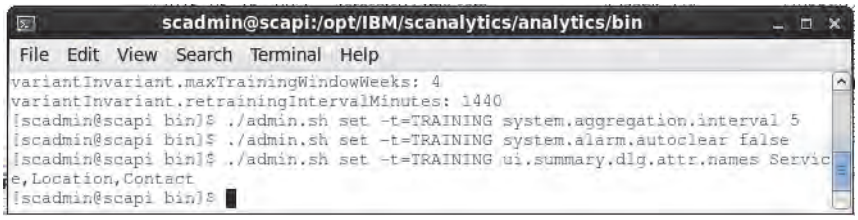
These attributes follow both the anomaly and the alarm that is generated from it. These attributes can be used by the Predictive Insights filtering tool to discard or elevate an alarm as well as by the OMNIbus rules file to enrich it.

It might also be useful to include these attributes in the Predictive Insights GUI to help operators troubleshoot issues and give them additional information. These attributes are designed to show in the following location.



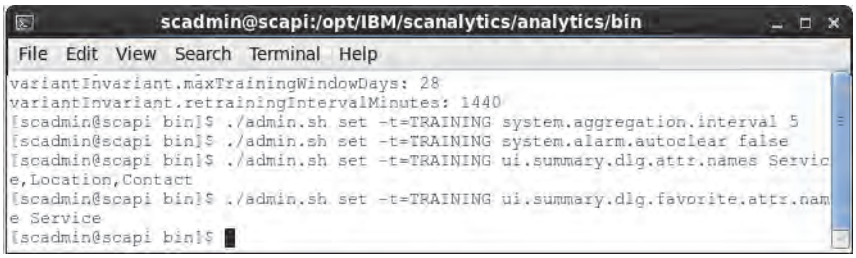
1. Add attributes to user interface:
   a. Enter the following command:

```
./admin.sh set -t=TRAINING ui.summary.dlg.attr.names Service,Location,Contact
```



2. Add the Service attribute to be the favorite attribute.
   a. Enter the following command:

```
./admin.sh set -t=TRAINING ui.summary.dlg.favorite.attr.name Service
```



Exercise 3 Using the filtered_alarms.txt file to increase severity of specific alarms

There are two places in Predictive Insights you can discard or enrich alarms:

- **filtered_alarms.txt**: This file is a filtering tool used by Predictive Insights that is designed to forward or discard an alarm to the OMNIbus probe. It allows you to look at the resource name, the metric group, metric, and its value to determine whether to forward it or not. If it is forwarded, the severity of the alarm can be modified if you so choose.
- **stdin-tasp.rules**: This file is used by the OMNIbus probe created for Predictive Insights. It has all the function of any other rules file to manipulate and enrich the alarm before it is forwarded to OMNIbus and saved in its database.

In the following exercise, you use the **filtered_alarms.txt** file to increase the severity of an alarm when the metric **InTotalBytes** is significantly higher than normal. Here are the rules you want to use:

- If the difference between the expected value and actual value for InTotalBytes on any resource is less than 500,000,000, discard the alarm
- If the difference between the expected value and actual value for InTotalBytes on any resource is less than 1,000,000,000, set the alarm to minor
- If the difference between the expected value and actual value for InTotalBytes on any resource is less than 2,000,000,000, set the alarm to major
- If the difference between the expected value and actual value for InTotalBytes on any resource is higher than 2,000,000,000, set the alarm to critical

You are also going to be keeping an eye on a specific resource. The **debit-meximxux11.fmx.com** server has been experiencing strange behavior recently and has been placed on a watch list. Any anomalies from this server need to be raised to critical immediately.

1. Open and review the comments in the **filtered_alarms.txt** file
   a. In a device terminal, open the filtered_alarms.txt file with your favorite editor, for example:

      ```
      vi
      /opt/IBM/scanalytics/analytics/spl/instances/AnalyticsTRAINING/config/filtered_alarms.txt
      ```

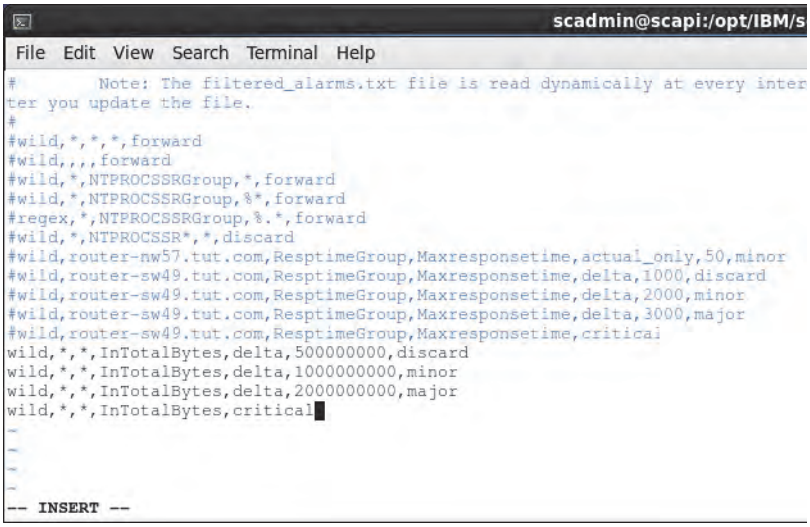
   b. Review the comments in this file to help you understand how each filter rule works.

Note: The **filtered_alarms.txt** file works much like an access control list. Each rule is applied, in order from top to bottom, to the alarm in that was generated by Predictive Insights. The first rule that matches based upon the resource name, the metric group, metric name, or a combination of these is used to discard or alter the severity of the alarm. All the others are ignored. If no rule is matched, the alarm is forwarded.

2. Add a filter that looks at the delta between the actual and expected values for the metric **InTotalBytes** and sets the severity of the alarms in the following manner. Note the wild cards that are needed for the resource and metric group positions in the filter.
   - If the delta between actual and expected values is less than 500,000,000, discard the alarm
   - If the delta between actual and expected values is less than 1,000,000,000, set the severity to minor
   - If the delta between actual and expected values is less than 2,000,000,000, set the severity to major
   - If the delta between actual and expected values is higher than 2,000,000,000, set the severity to critical

   a. Add the following four lines to the **filtered_alarms.txt** file.

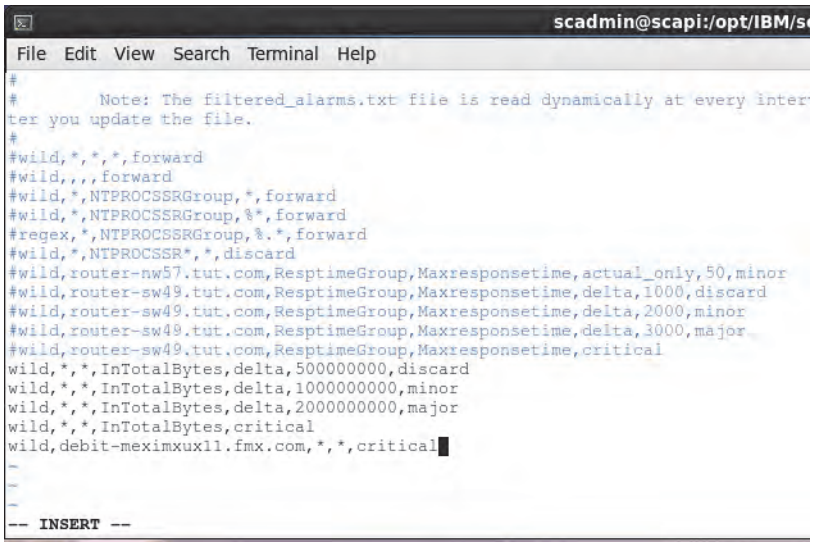
```
wild,*,*,InTotalBytes,delta,500000000,discard
wild,*,*,InTotalBytes,delta,1000000000,minor
wild,*,*,InTotalBytes,delta,2000000000,major
wild,*,*,InTotalBytes,critical
```



Note: Comparison of actual and expected values to a threshold are based on LESS THAN logic. For example, is the delta between the actual and expected values LESS THAN a specific threshold. With this logic, you must consider how you want to build your rules if you want to control severity. In the example, you have to order the rules smallest to largest or all InTotalBytes anomalies would be considered critical.

3. You must ensure that any anomaly from the **debit-meximxux11.fmx.com** node is raised to critical. Note the wild cards used for the metric group and metric name positions.
   a. Add the following line to the **filtered_alarms.txt** file.

      `wild,debit-meximxux11.fmx.com,*,*,critical`



   b. Save the file.

Important: Double-check the metric and resource name spelling and capitalization. If there are errors, the filter does not match the values you created in your model and does not work.

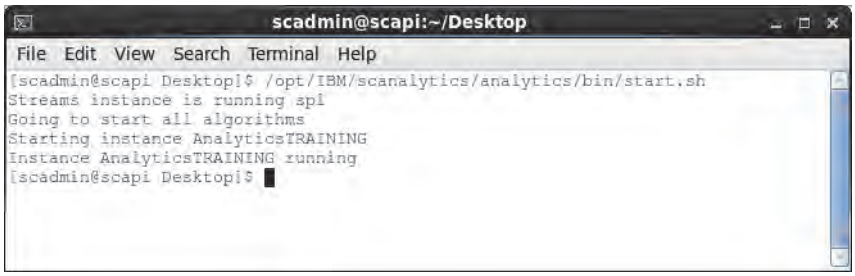
Exercise 4 Starting the analysis of the Online-Banking project

With the server configured, you can begin the analysis of your data sources. You must first start the analytics server. After you start it, you must check that all of its processes are running and healthy. You can then start the analysis of the Online-Banking model that you created. Because this analysis is historical, using data that was created in May of 2014, you must use an end date on the **run_extractor_instance** command. Otherwise, the extractor processes to the current system clock.

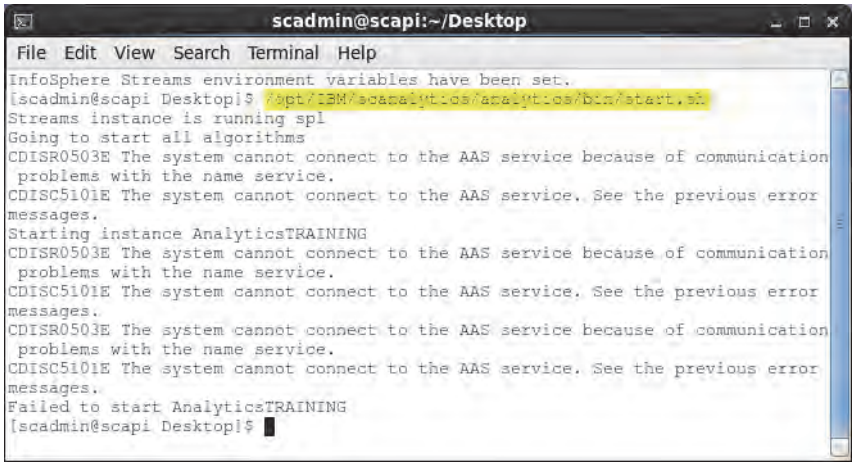
1. Start the Predictive Insights server with the **start.sh** command. In a Terminal window, enter this command:

   `/opt/IBM/scanalytics/analytics/bin/start.sh`

The startup process takes a few minutes.



Note: On occasion, the server does not start correctly and you get the following error:

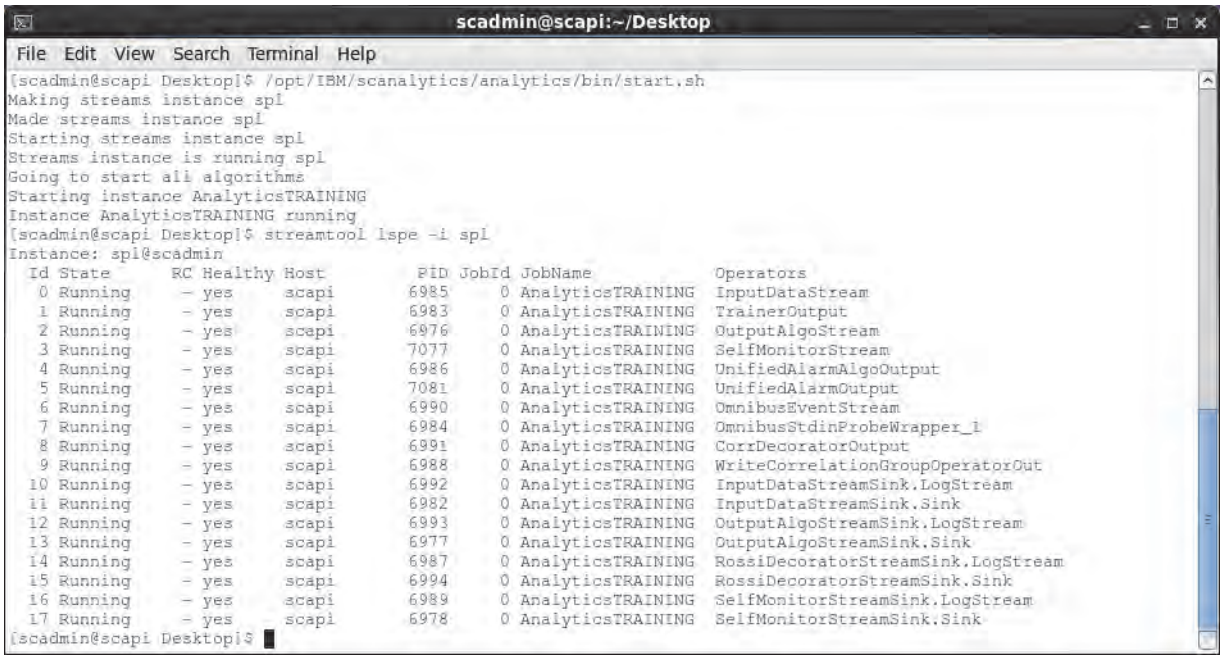


If that occurs, run the following two commands and attempt to restart the server. Note that spl is lowercase.

```
streamtool stopinstance -i spl --force
streamtool rminstance -i spl --noprompt
```

2. Confirm that all the Predictive Insight processes started.
   a. Enter the following command to get the status of the running server. *LSPE* and SPL is in lowercase.

      ```
      streamtool lspe -i spl
      ```

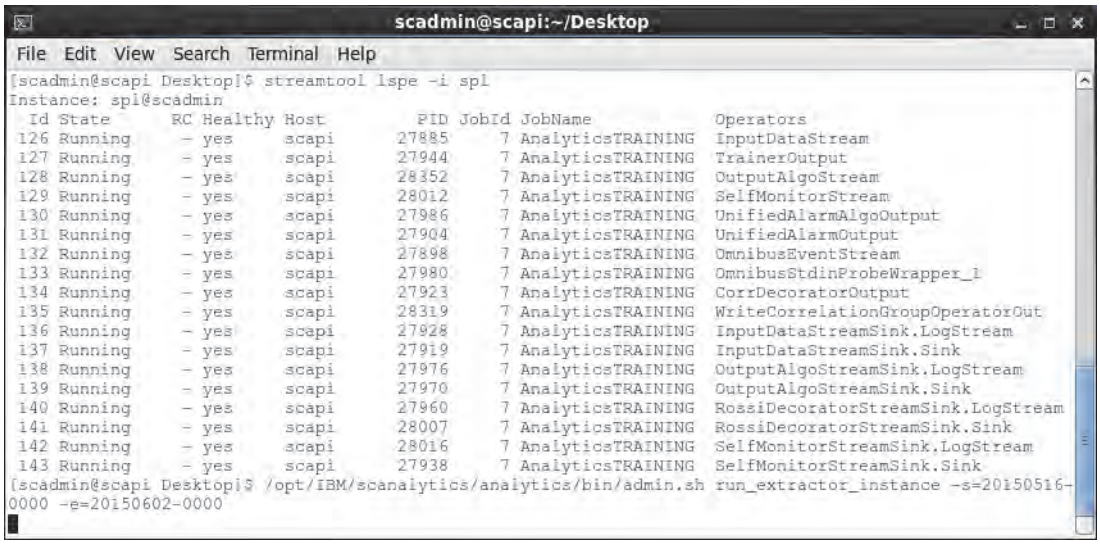


   b. If one of the processes is not running and healthy, use the following commands and check the processes again:

      ```
      /opt/IBM/scanalytics/analytics/bin/stop.sh
      streamtool stopinstance -i spl --force
      streamtool rminstance -i spl --noprompt
      /opt/IBM/scanalytics/analytics/bin/start.sh
      ```

3. Start the extraction with the command **run_extractor_instance** that has a start time of 4 October 2015 and an end time of 25 October 2015. Use midnight as the start and stop times for those dates. Enter the following command to start the extraction process:

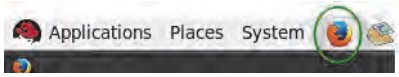
   `/opt/IBM/scanalytics/analytics/bin/admin.sh run_extractor_instance -s=20151004-0000 -e=20151025-0000`



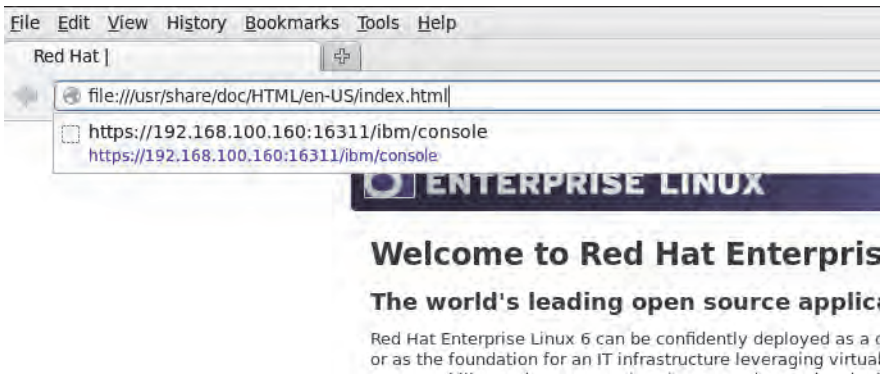
Exercise 5 Confirming data extraction

After the extraction begins, ensure that data is retrieved from your data sources. In this exercise, you review the messages, logs and directories that confirm that data is collected by the analytics server.

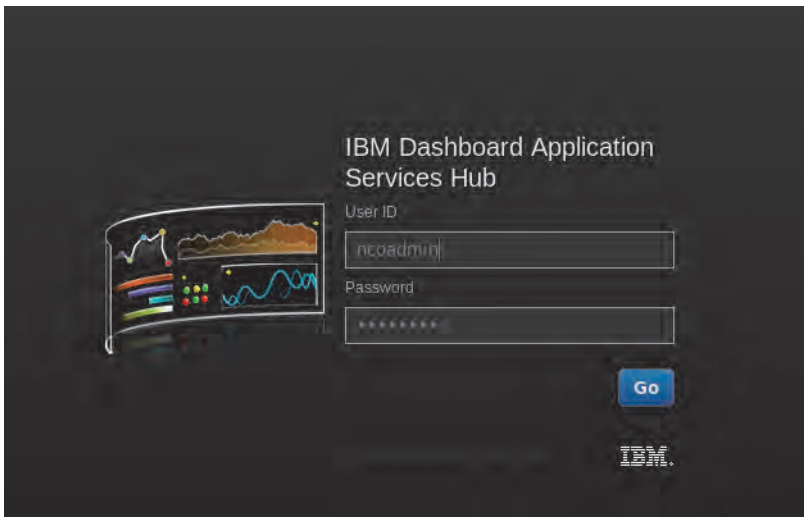
1. Check the Event Viewer in DASH to view Predictive Insights messages:
   a. Open the Firefox browser.



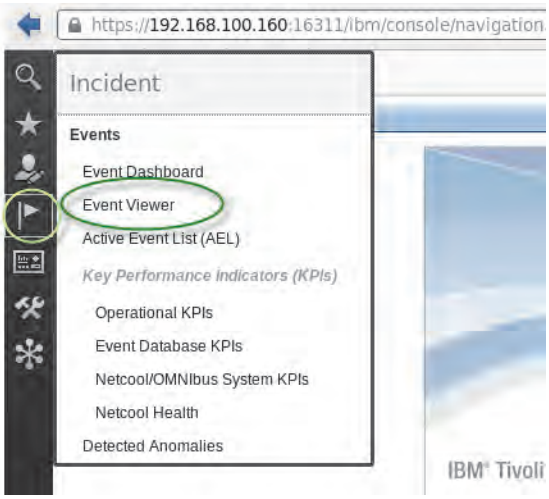
   b. Point the browser to **https://192.168.100.160:16311/ibm/console**.



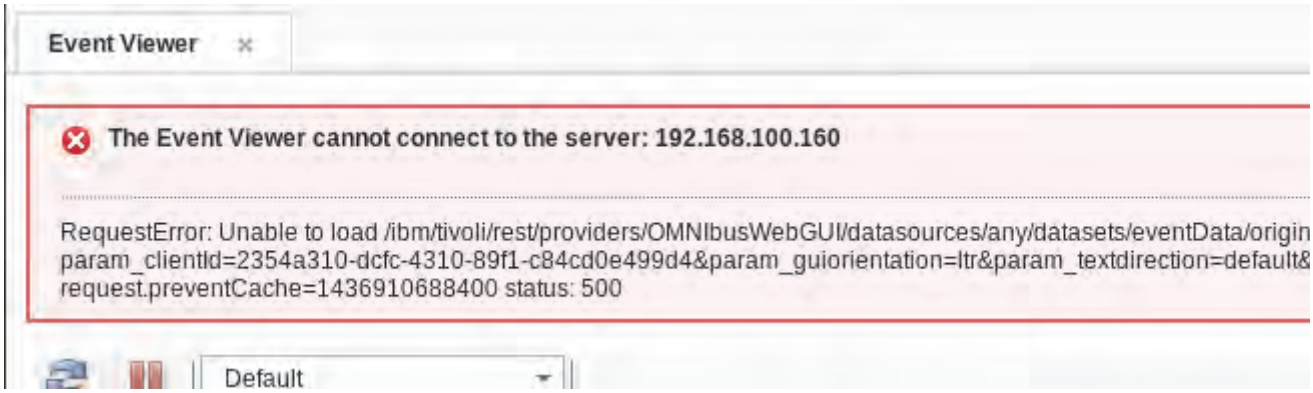
c. Log in with the user name **ncoadmin** and password **object00**.



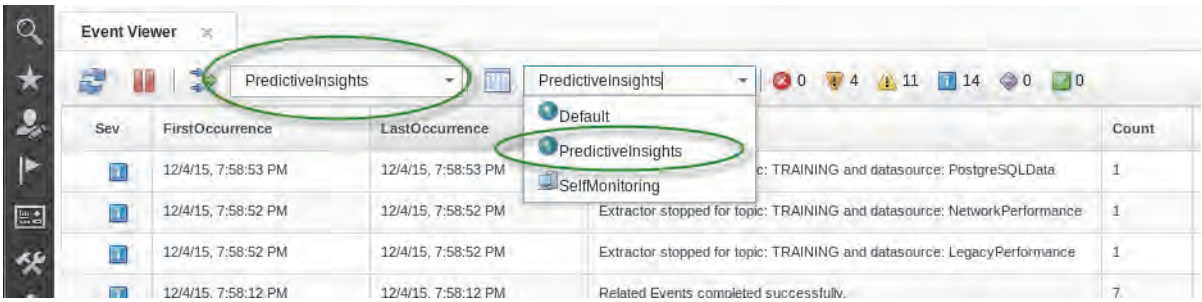
d. Select the Event Viewer from the side menu.



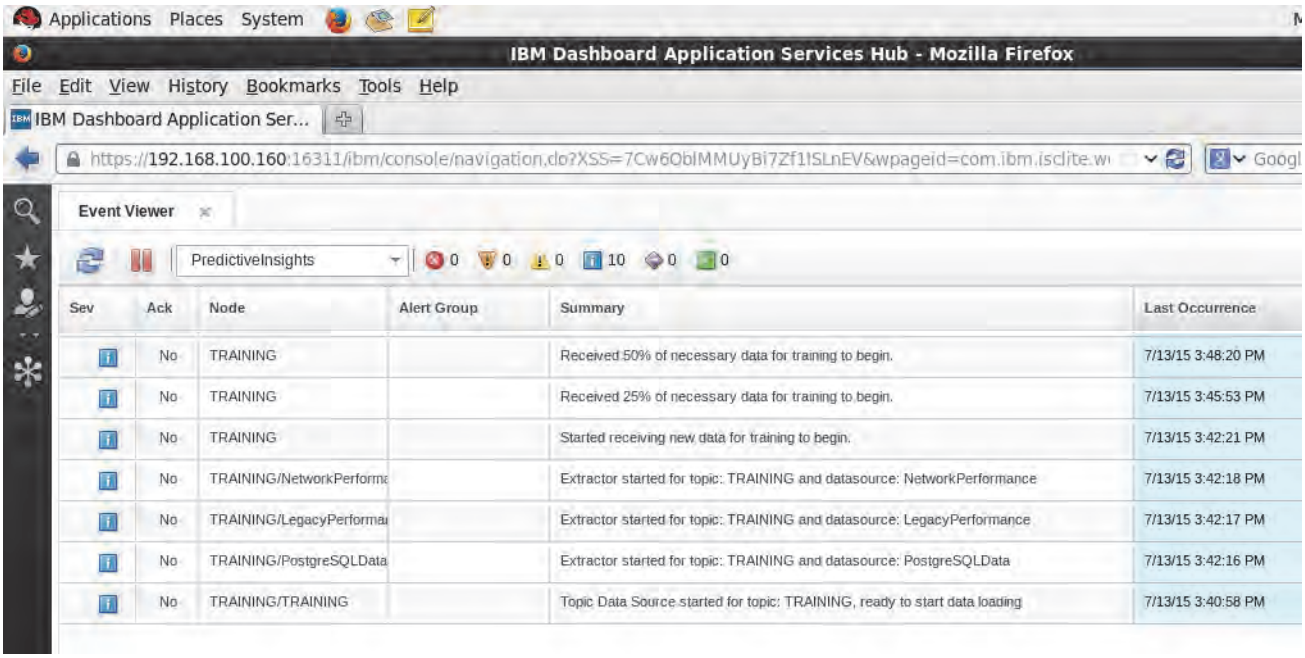
Note: If you get the following error, use the refresh button to clear it.



e. Choose the PredictiveInsights filter and Predictive Insights view.

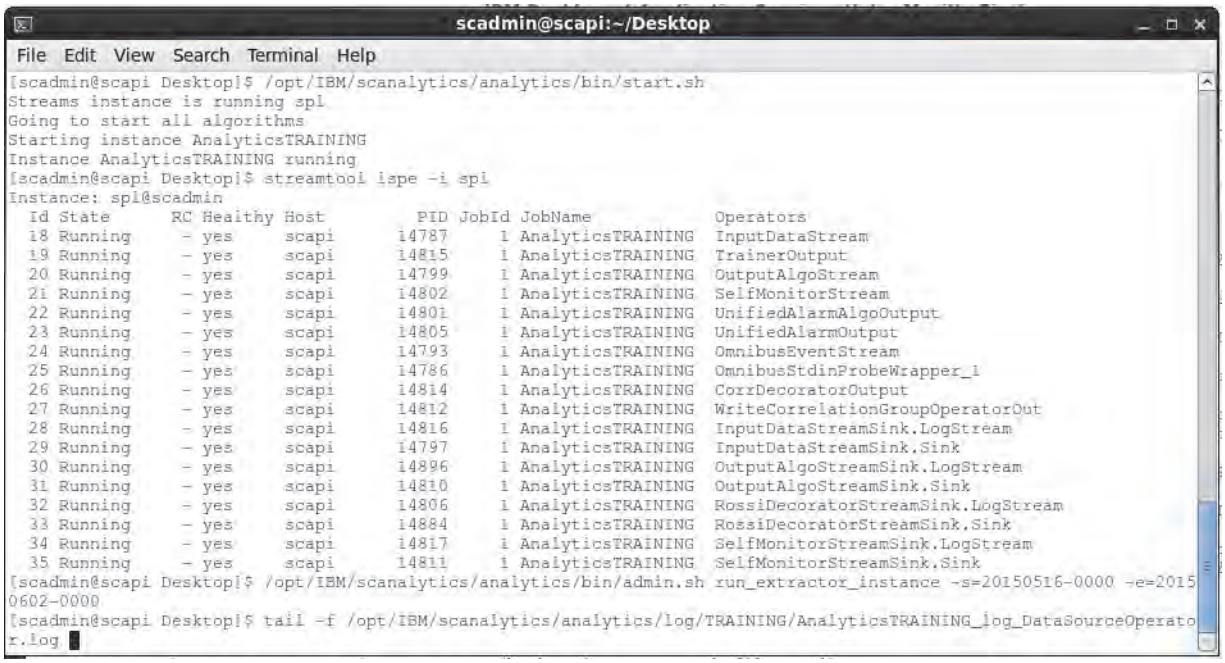


f. Review the messages about the topics that have started.

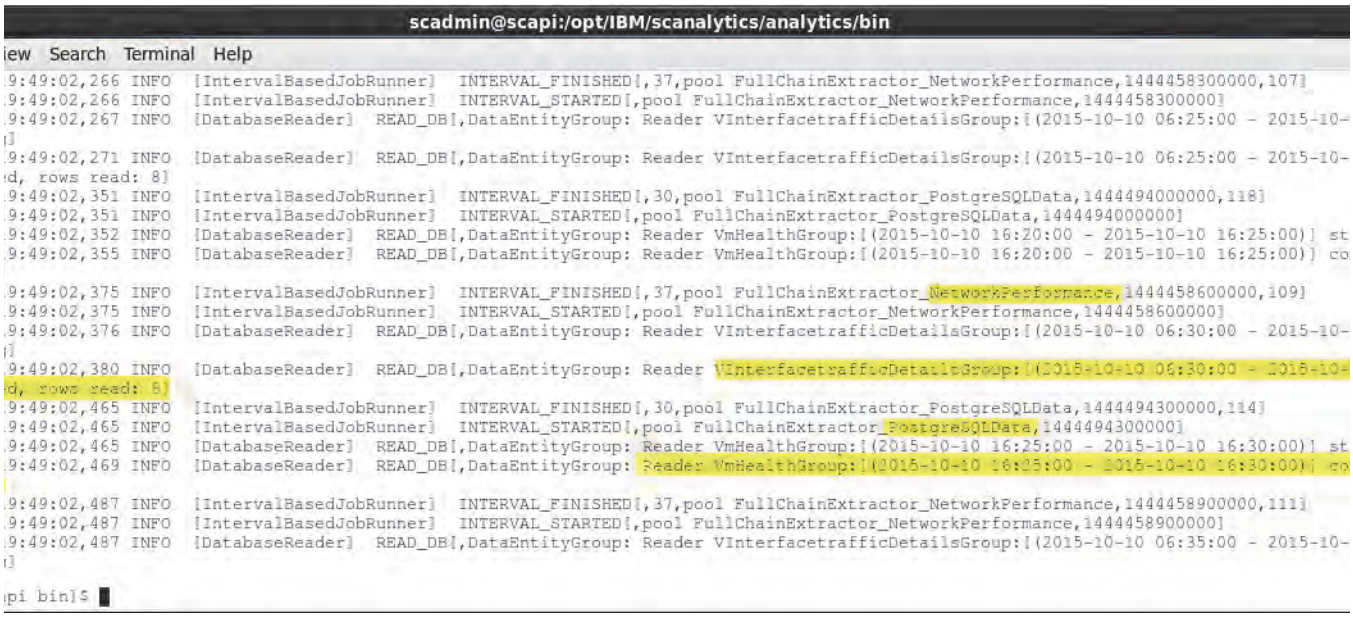


2. Review the **/opt/IBM/scanalytics/analytics/log/TRAINING/AnalyticsTRAINING_log_DataSourceOperator.log** log to confirm data is being found by the extractor:
   a. In a device terminal, enter the following command:

      ```
      tail -f
      /opt/IBM/scanalytics/analytics/log/TRAINING/AnalyticsTRAINING_log_DataSourceOperator.log
      ```



   b. After scrolling messages for a few moments, stop the tail and review the log. Note the data source name, the time stamps for extracting data, and the amount of data that it retrieved.

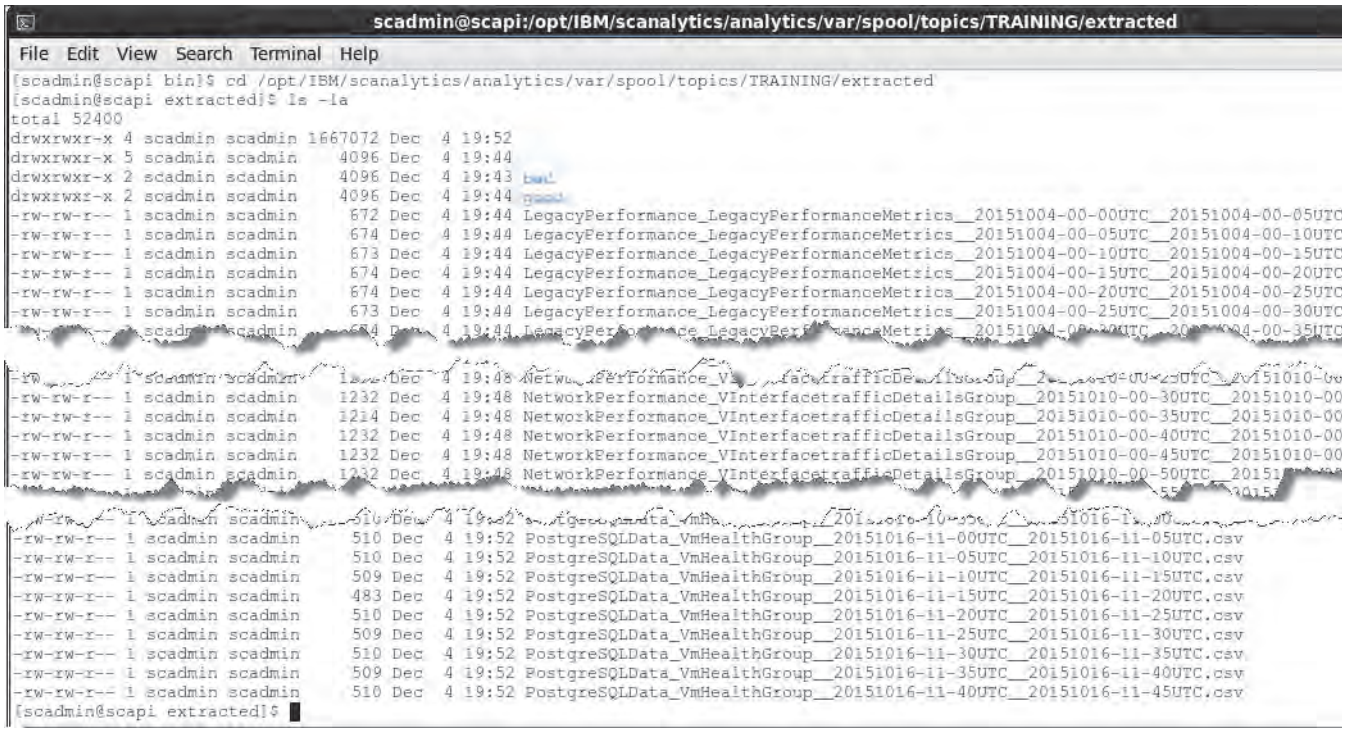


3. Review the **/opt/IBM/scanalytics/analytics/var/spool/topics/TRAINING/extracted** directory.

As data is extracted from the data sources, a copy of the data is placed in the extracted directory.

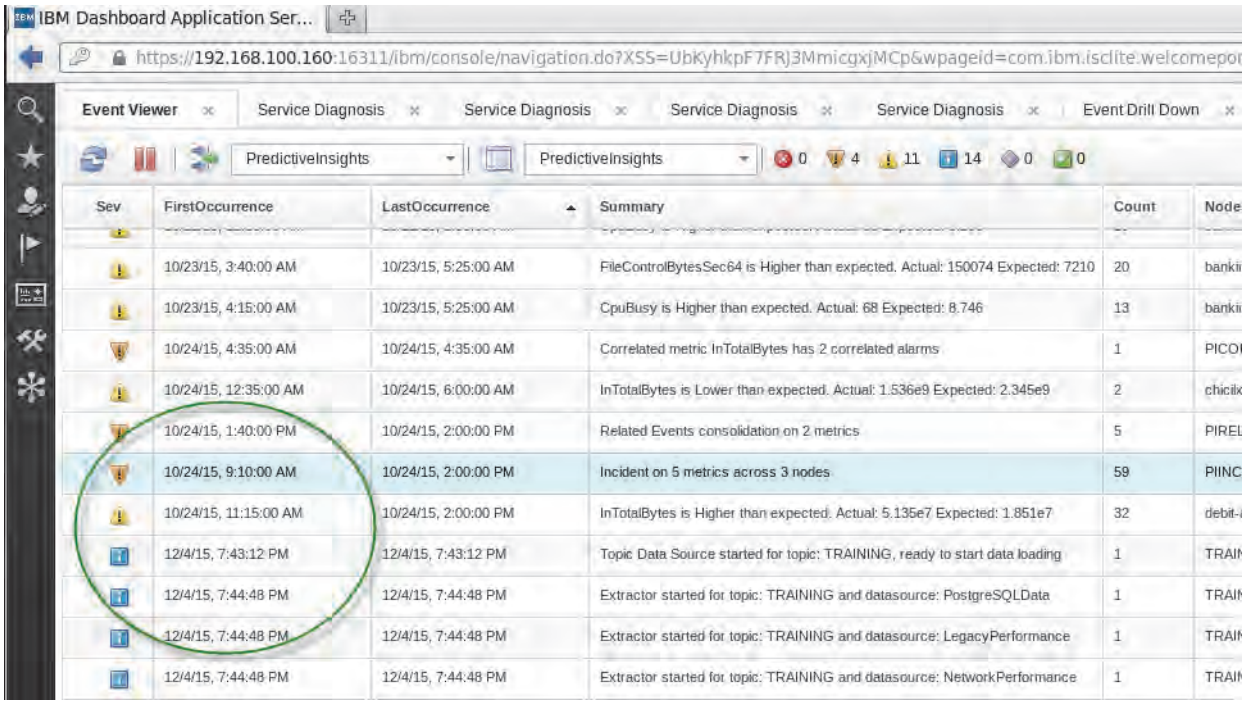
a. Change directory to the following location:
cd /opt/IBM/scanalytics/analytics/var/spool/topics/TRAINING/extracted

b. List the files in this directory.



Note: Review the format of the files. It denotes the data source name, the metric group, and the time stamps of the data extraction. You can use these files to restore the analytics model, if needed. This directory keeps only the data for restoring the current analytics model. Older data is removed automatically.

4. Wait for the first anomaly to show. It can take up to 15 minutes before you see your first anomaly. After you see the first anomaly, the remaining ones show shortly thereafter. Note the time stamp differences. Because you are reviewing historical data, there is a time skew between the analysis and the data.



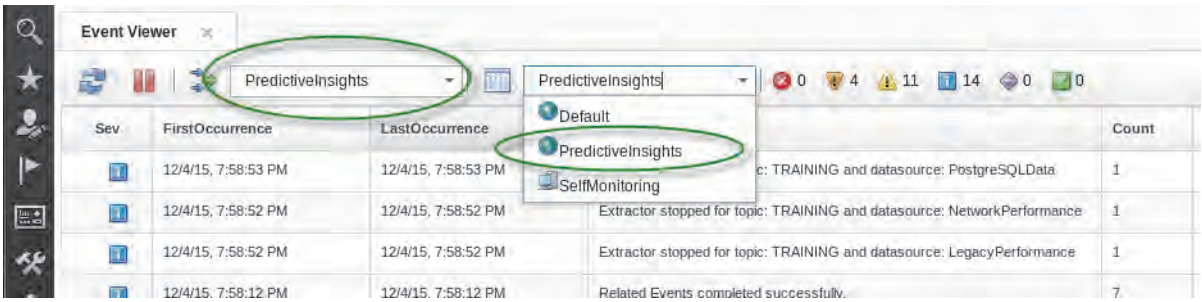
Note: Keep your browser open to use in subsequent exercises.

Exercise 6 Confirming the escalation of alarms by filtered-alarms.txt

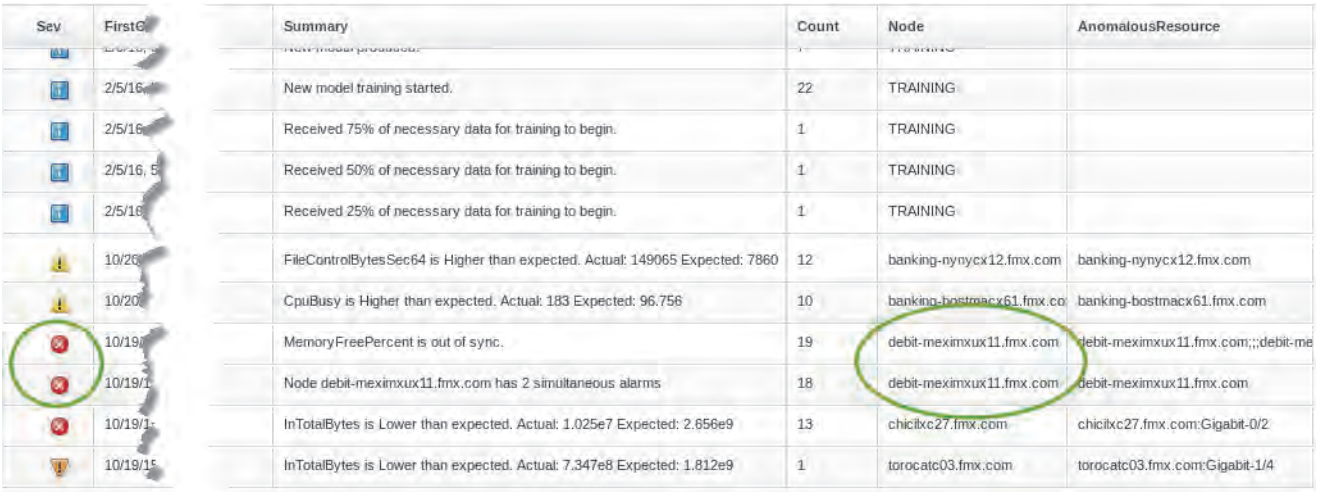
In Exercise 3, “Using the filtered_alarms.txt file to increase severity of specific alarms,” on page 6, you added filters to the **filtered-alarm.txt** file to escalate alarms on the metric **InTotalBytes** and the resource **debit-meximxux11.fmx.com**. Confirm the alarms were escalated accordingly.

Note: By default, most alarms from Predictive Insights are considered minor. Only consolidated alarms are treated as major.

1. Return to the **Event Viewer** and note the alarms that were generated. Set the filter to **PredictiveInsights** and the view to **PredictiveInsights**.

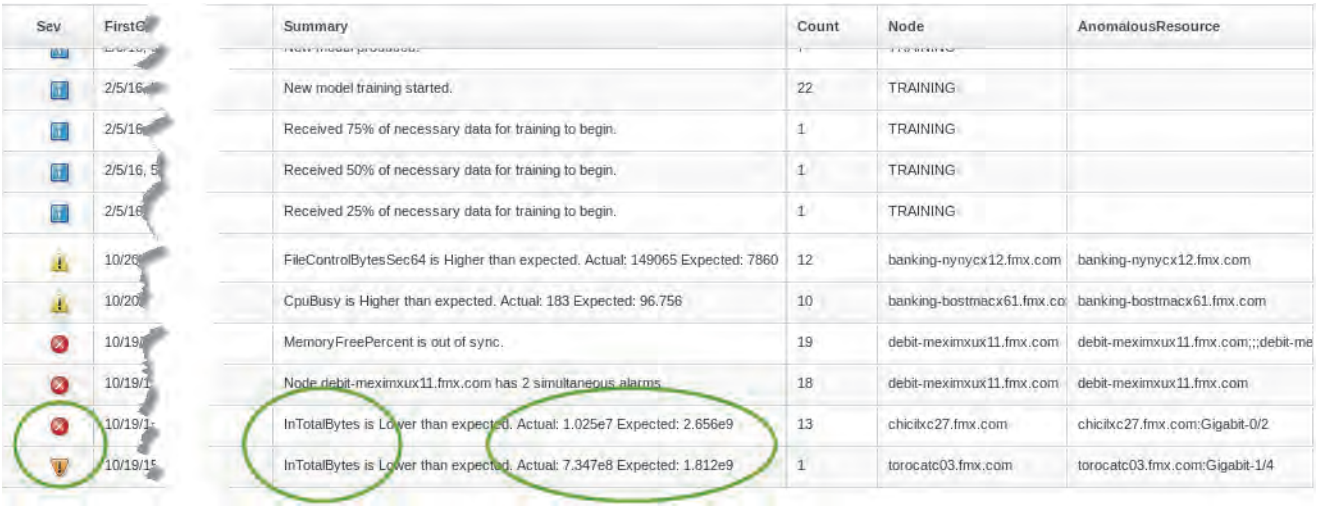


2. Look for the resource **debit-meximxux11.fmx.com** in the Node column. Note the severity of the alarms are critical.



3. Look for the **InTotalBytes** metric in the Summary column. Note the alarms are either critical or major based on how big the difference is between the actual and expected values. Remember that in Exercise 3, “Using the filtered_alarms.txt file to increase severity of specific alarms,” on page 6, you

set a series of filters for this metric that discarded the alarm if the difference was low and then increased its severity as the differences got larger.



Sev	FirstO	Summary	Count	Node	AnomalousResource
	2/5/16,	New model training started.	22	TRAINING	
	2/5/16,	Received 75% of necessary data for training to begin.	1	TRAINING	
	2/5/16, 5	Received 50% of necessary data for training to begin.	1	TRAINING	
	2/5/16,	Received 25% of necessary data for training to begin.	1	TRAINING	
	10/26/	FileControlBytesSec64 is Higher than expected. Actual: 149065 Expected: 7860	12	banking-nynycx12.fmx.com	banking-nynycx12.fmx.com
	10/20/	CpuBusy is Higher than expected. Actual: 183 Expected: 96.756	10	banking-bostmacx61.fmx.co	banking-bostmacx61.fmx.com
	10/19/	MemoryFreePercent is out of sync.	19	debit-meximxux11.fmx.com	debit-meximxux11.fmx.com;;;debit-me
	10/19/1	Node debit-meximxux11.fmx.com has 2 simultaneous alarms	18	debit-meximxux11.fmx.com	debit-meximxux11.fmx.com
	10/19/1	InTotalBytes is Lower than expected. Actual: 1.025e7 Expected: 2.656e9	13	chicitxc27.fmx.com	chicitxc27.fmx.com:Gigabit-0/2
	10/19/15	InTotalBytes is Lower than expected. Actual: 7.347e8 Expected: 1.812e9	1	torocatc03.fmx.com	torocatc03.fmx.com:Gigabit-1/4

Exercise 7 Reviewing the features of an anomaly

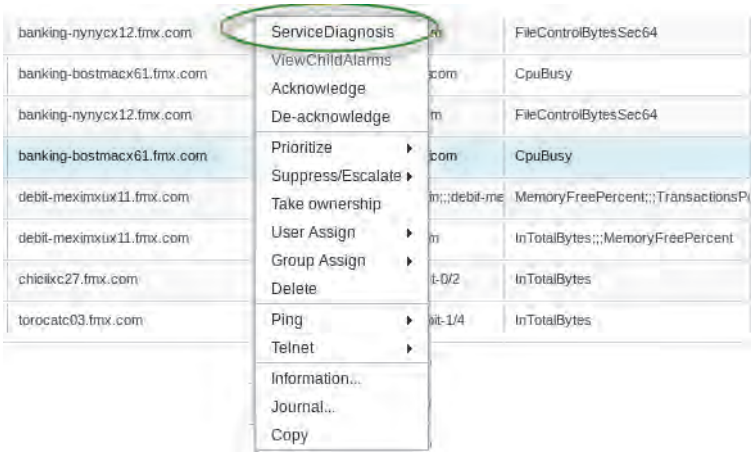
In this exercise, you review an alarm and learn the information that can be found in the Predictive Insights user interface.

Complete the following steps:

1. Find the anomaly associated to the **banking-bostmacx61.fmx.com** resource that last occurred at 2pm on 10/20/15. Open the anomaly using the ServiceDiagnosis menu pick.

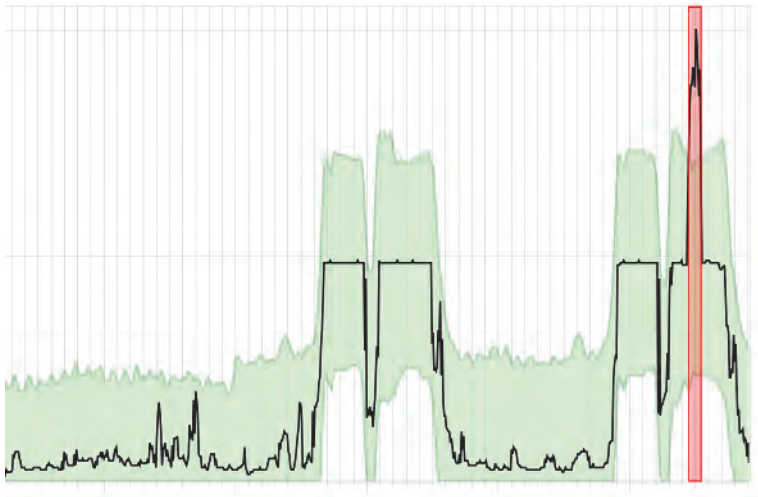
Sev	FirstOccurrence	LastOccurrence	Summary	Count	Node	Anomalou
	10/22/15, 12:15:00 AM	10/22/15, 1:55:00 AM	CpuBusy is Higher than expected. Actual: 68 Expected: 8.206	19	banking-bostmacx61.fmx.com	banking-bo
	10/20/15, 1:00:00 PM	10/20/15, 2:05:00 PM	FileControlBytesSec64 is Higher than expected. Actual: 149065 Expecte	12	banking-nynycx12.fmx.com	banking-ny
	10/20/15, 1:05:00 PM	10/20/15, 2:00:00 PM	CpuBusy is Higher than expected. Actual: 183 Expected: 96.756	10	banking-bostmacx61.fmx.com	banking-bo
	10/19/15, 7:35:00 AM	10/19/15, 1:20:00 PM	MemoryFreePercent is out of sync.	19	debit-meximxux11.fmx.com	debit-mexi
	10/19/15, 8:40:00 AM	10/19/15, 1:15:00 PM	Node debit-meximxux11.fmx.com has 2 simultaneous alarms	18	debit-meximxux11.fmx.com	debit-mexi

   a. Right click on this anomaly and select ServiceDiagnosis



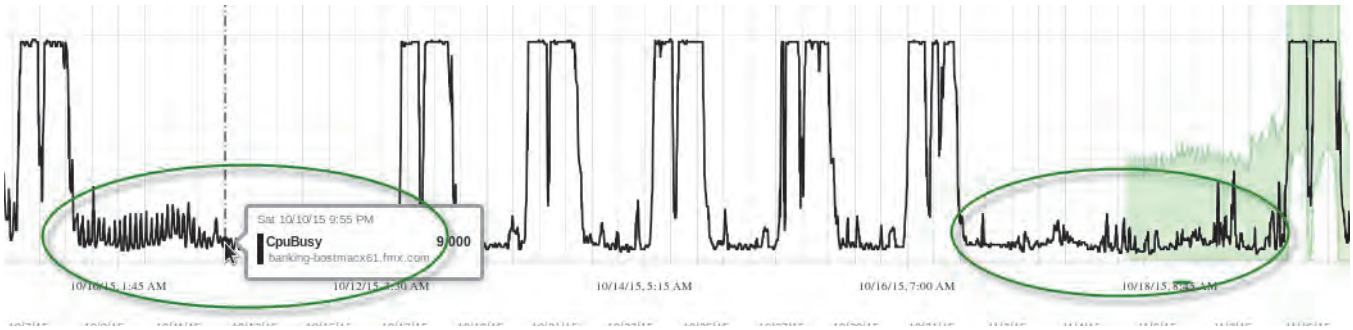
   b. If you are asked to authenticate to the Predictive Insights server, use the user name **ncoadmin** and password **object00**.

Note the variable thresholds: the green area on the graph. That area represents the upper and lower thresholds for this metric, which vary from day to day.

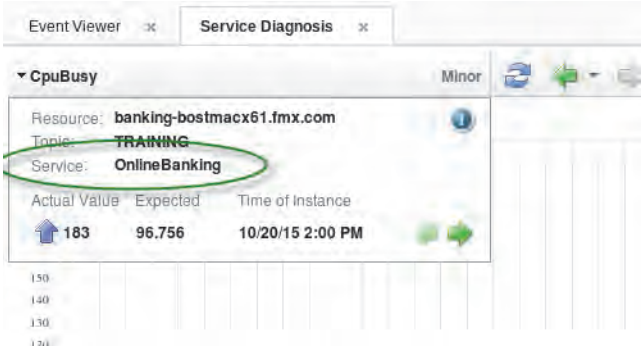


2. To view all of the previous data, select the left slider and move it to sometime around Oct 4th.

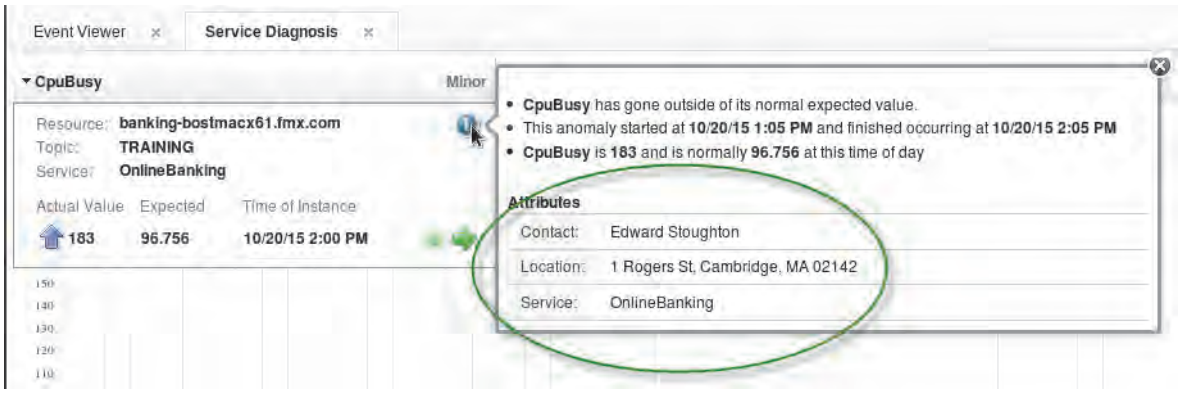
Note the higher peaks happening on the weekdays. However, on the weekends the activity is much less. Mousing over the graph provides you the value and time of the data.



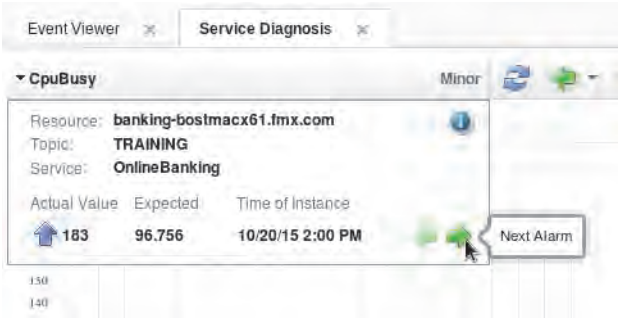
3. Note the custom attributes that are part of the user interface.
   a. Review the details about the anomaly in the upper left corner of the screen. Note the Service attribute that was configured for the server earlier.



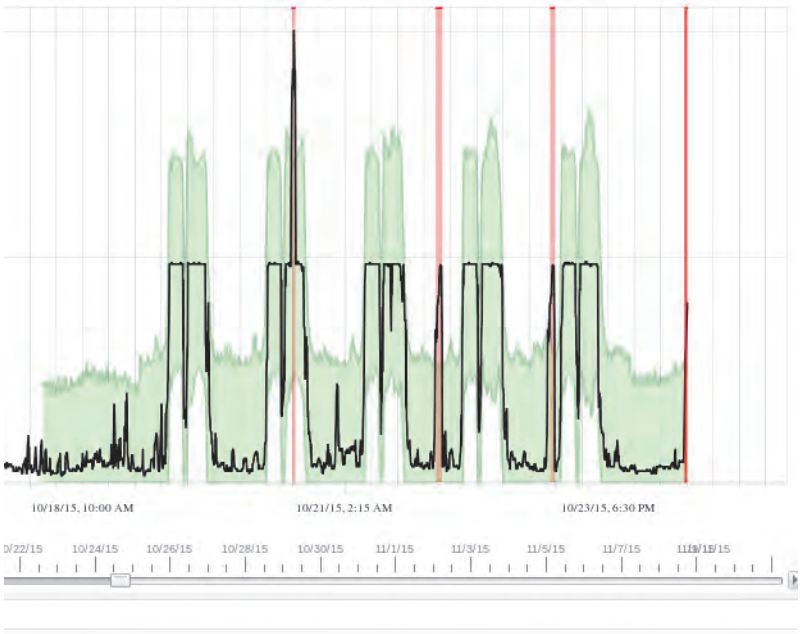
   b. Mouse over the information icon and see the other attributes that you configured to be displayed in the user interface



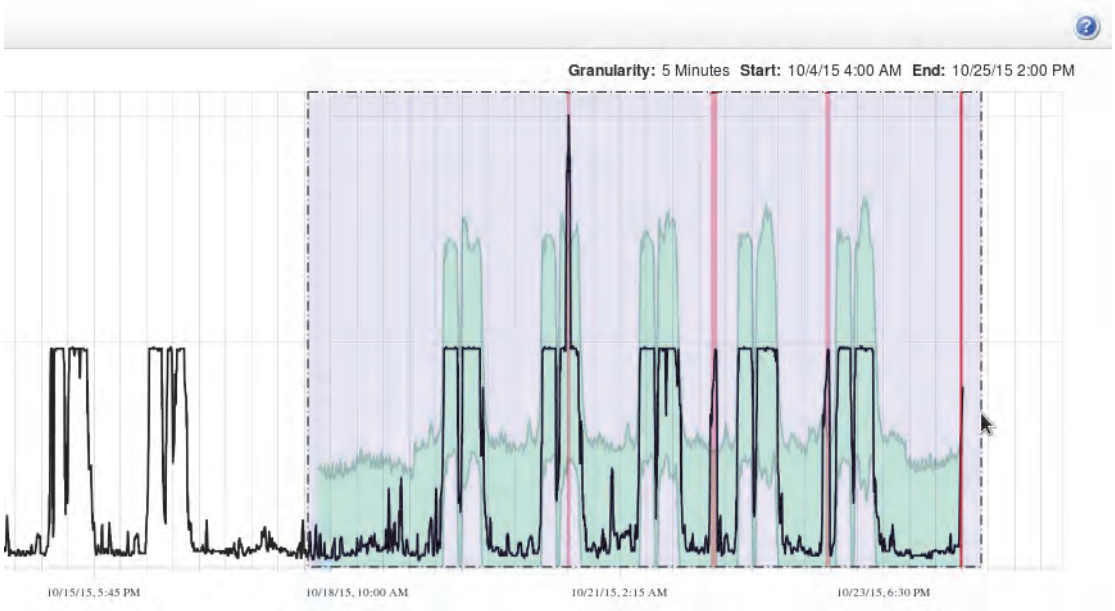
4. Peruse the other anomalies that have occurred on this KPI. Its important to understand that you opened this anomaly up on October 20th. There is three more days of data that was consumed by the historical analysis where more anomalies occurred after Oct 20th.
   a. Click on the green arrow to move forward to the next alarm.



b. Repeat this process until you see all four anomalies that occurred on this KPI.

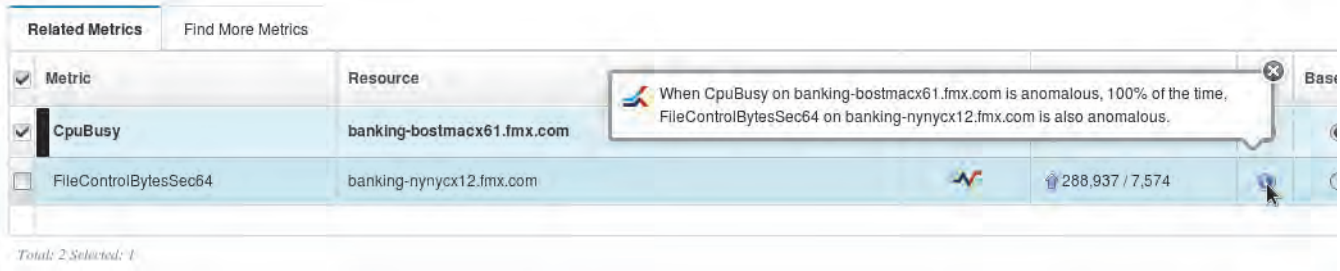


5. Zoom into the details of these anomalies.Using the slide bar to zooming out and in can be done, a more accurate way to zoom in is to click and drag across the graph to view the area of it you are interested in.
   a. Click to the left of the left-most anomaly and drag to the right so the shaded box covers all four anomalies.



   b. Note the change in the view.

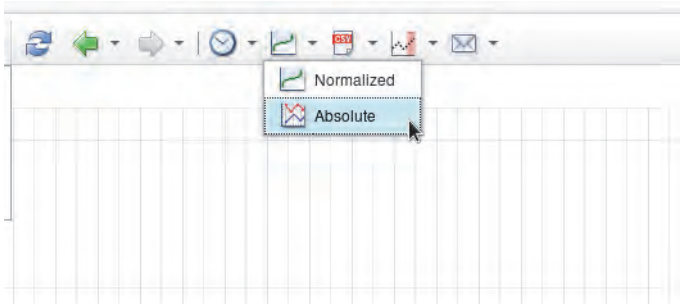
6. Review the related metrics in the lower half of the interface.
   a. Hover over the information icon for the FileControlBytesSec64 metric to understand why it is related to CpuBusy. Note that they are separate resources. Note that these two metrics are anomalous at the same time. Metrics that are anomalous at the same time are called **related metrics**.



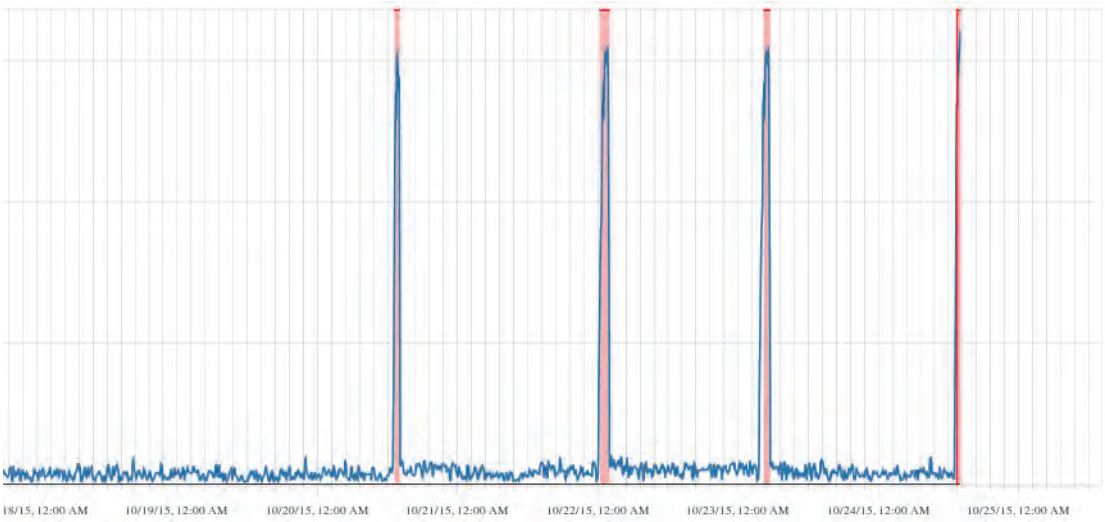
   b. Click the check box to display the FileControlBytesSecs64 data with CpuBusy data. See how the data spikes at the same time.

It is important to realize that the user interface has normalized and absolute views for displaying its data. As soon as you view more than one data stream, the default is to display the data normalized.

7. Select the absolute view to see the FileControlBytesSecs64 and CpuBusy data streams.



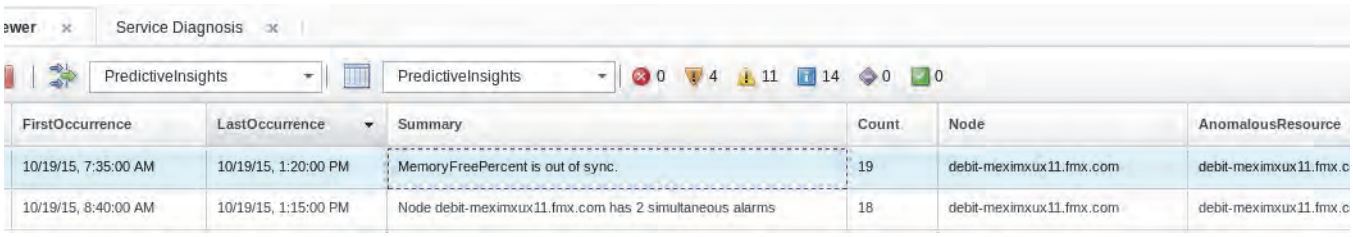
Note the change in the view. This is because the CpuBusy values range in the 10s to 100s. TheFileControlBytesSecs64 values range in the 1,000s to 100,000. The CpuBusy data is rendered as a straight line because of these differences.



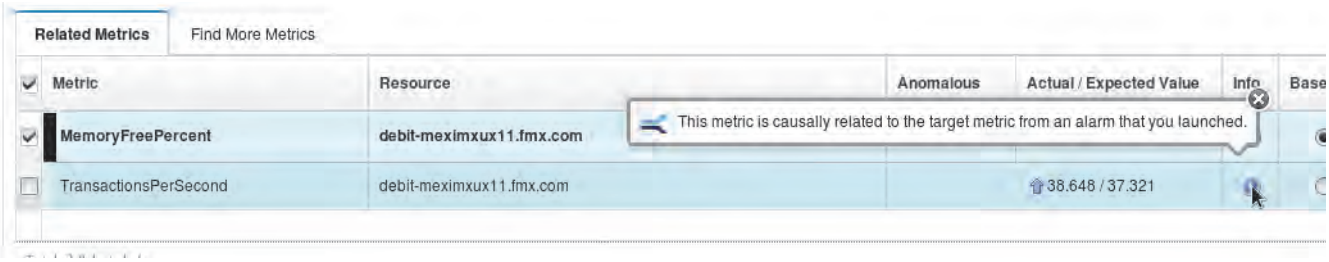
Exercise 8 Reviewing a Granger alarm

A Granger alarm is generated when a behavior between KPIs breaks down. In this exercise, you review a KPI that has become out-of-synch with the KPIs that are mathematically linked with each other.

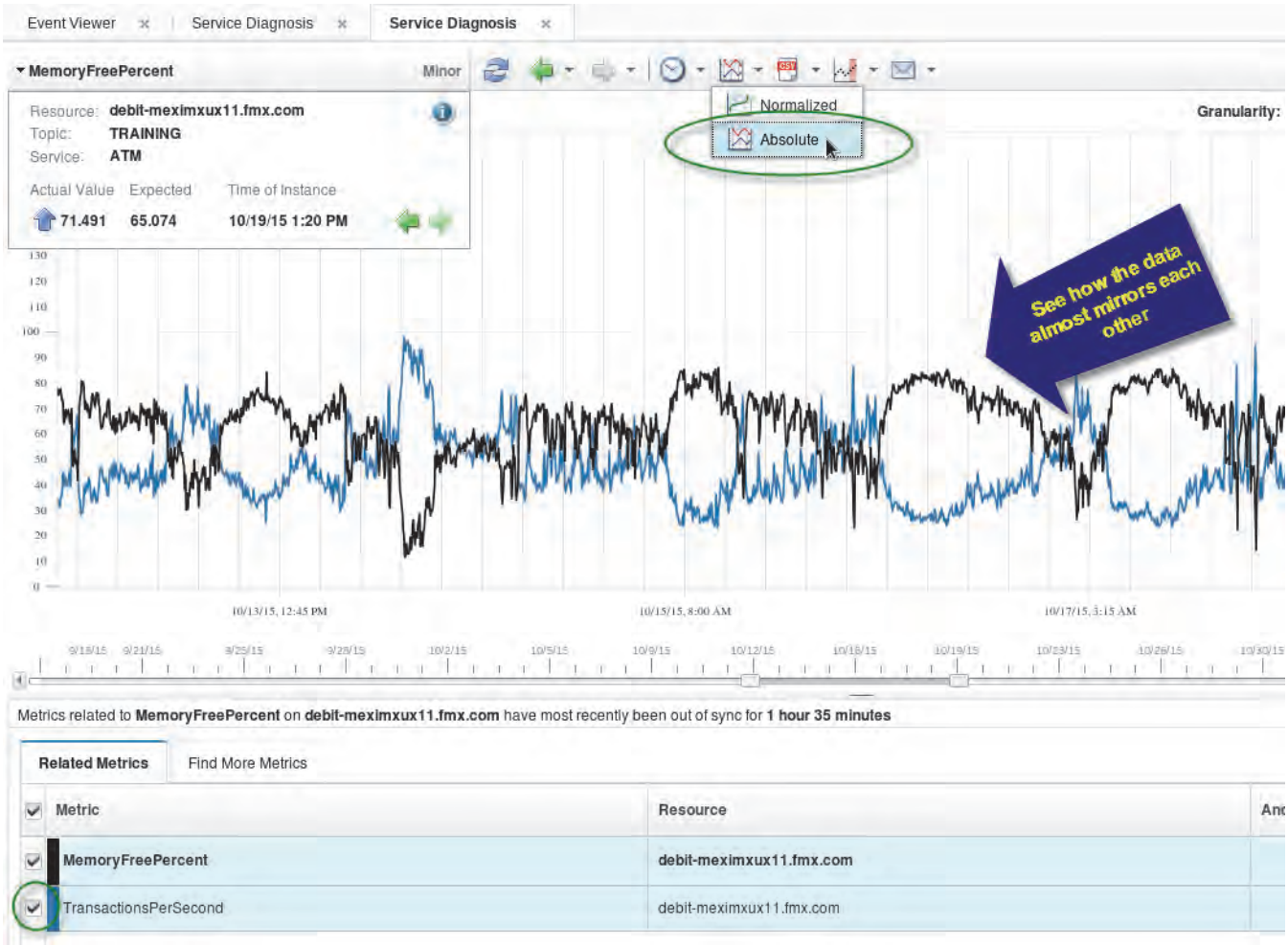
1. Return to the event viewer and search for the alarm that last occurred on Oct 19th, at 1:20pm. and has the Summary **MemoryFreePercent is out of sync**. Right click on it and select ServiceDiagnosis.



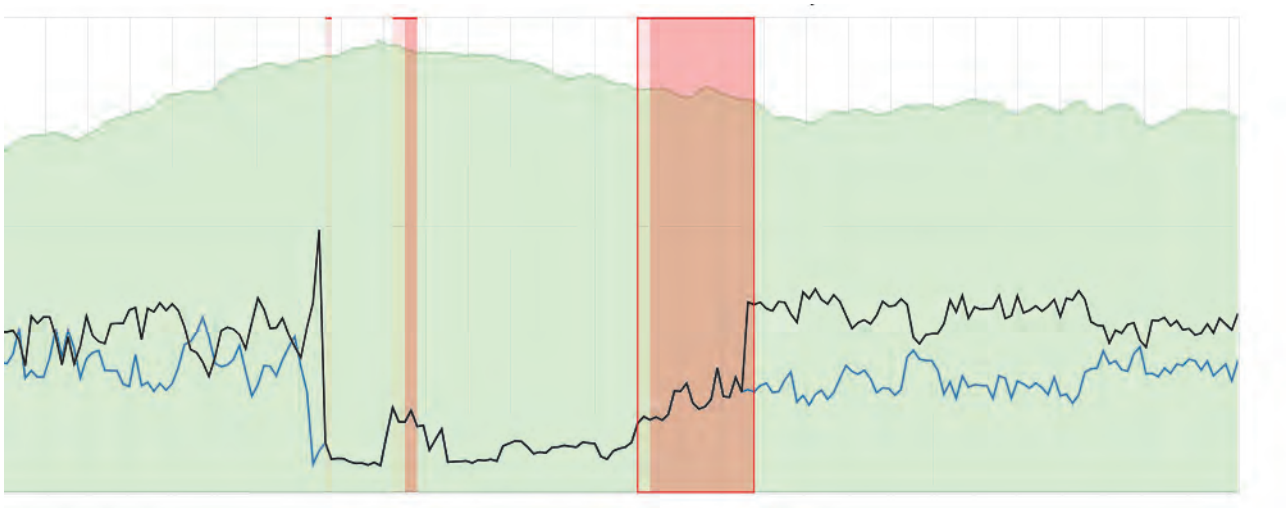
2. Hold your mouse pointer over the information icon in the **Related Metrics** tab for **TransactionsPerSecond**. Note that it is causally related to **MemoryFreePercent**. This relationship means that MemoryFreePercent tends to predict TransactionsPerSecond.



3. Display the data associated to **TransactionsPerSecond** and view the data using the absolute view. Notice how the data almost mirrors each other around an imaginary plane. You can see that as the amount of free memory increases, the number of transactions decrease.



Notice that the anomaly occurs when this inverse relationship breaks down and becomes proportional. The anomaly goes away once the inverse relationship is restored.



Exercise 9 Reviewing correlated metrics

In an effort to help operators troubleshoot an alarm, Predictive Insights displays *related metrics*. The related metric tab displays metrics that have one or more of the following properties:

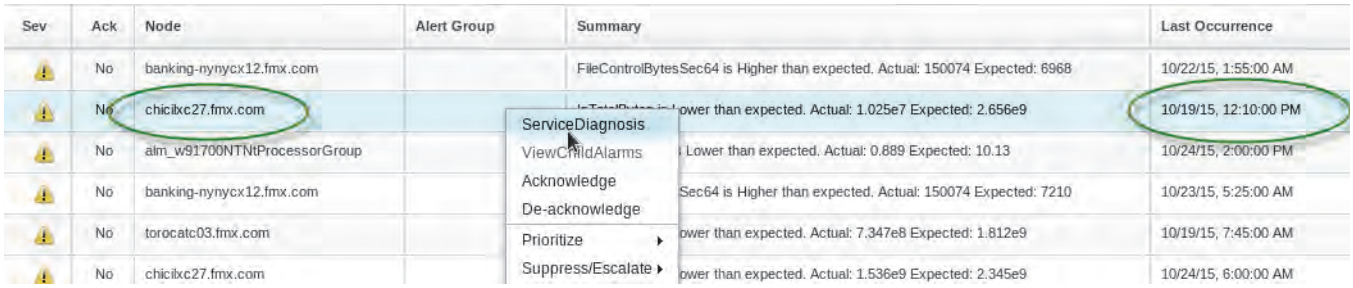
- Metrics that have a **causal** relationship to the alarm you are currently viewing (as seen in the previous exercise)

- Metrics that display the tendency to be anomalous at the same time which are referred to as **related**.

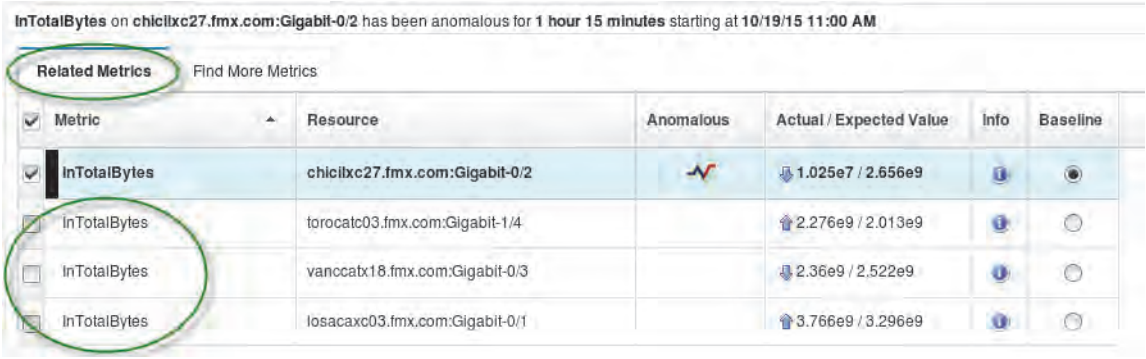
- Metrics that have a tight relationship in their data which is referred to as **correlated**.

In this exercise, you review an alarm and the associated metrics that Predictive Insights believes has statistical significance.

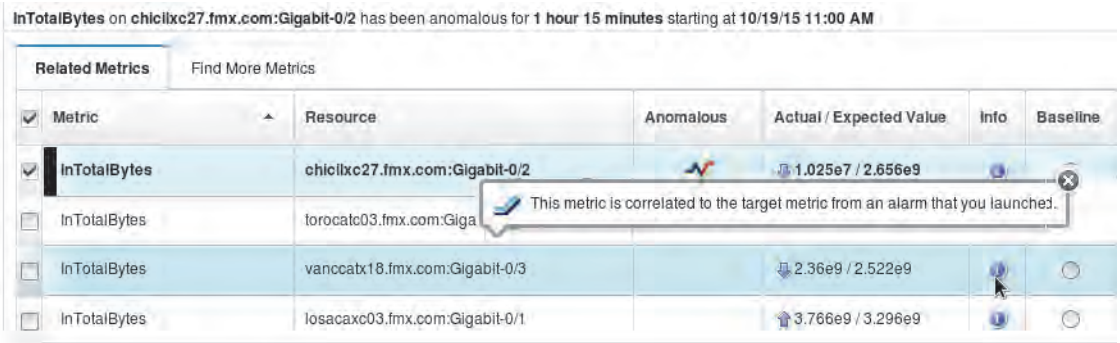
1. Return to the event viewer and search for the anomaly that last occurred on the **chicilxc27.fmx.node** at **12:10pm on October 19th**. Click right on the anomaly and select **ServiceDiagnosis**



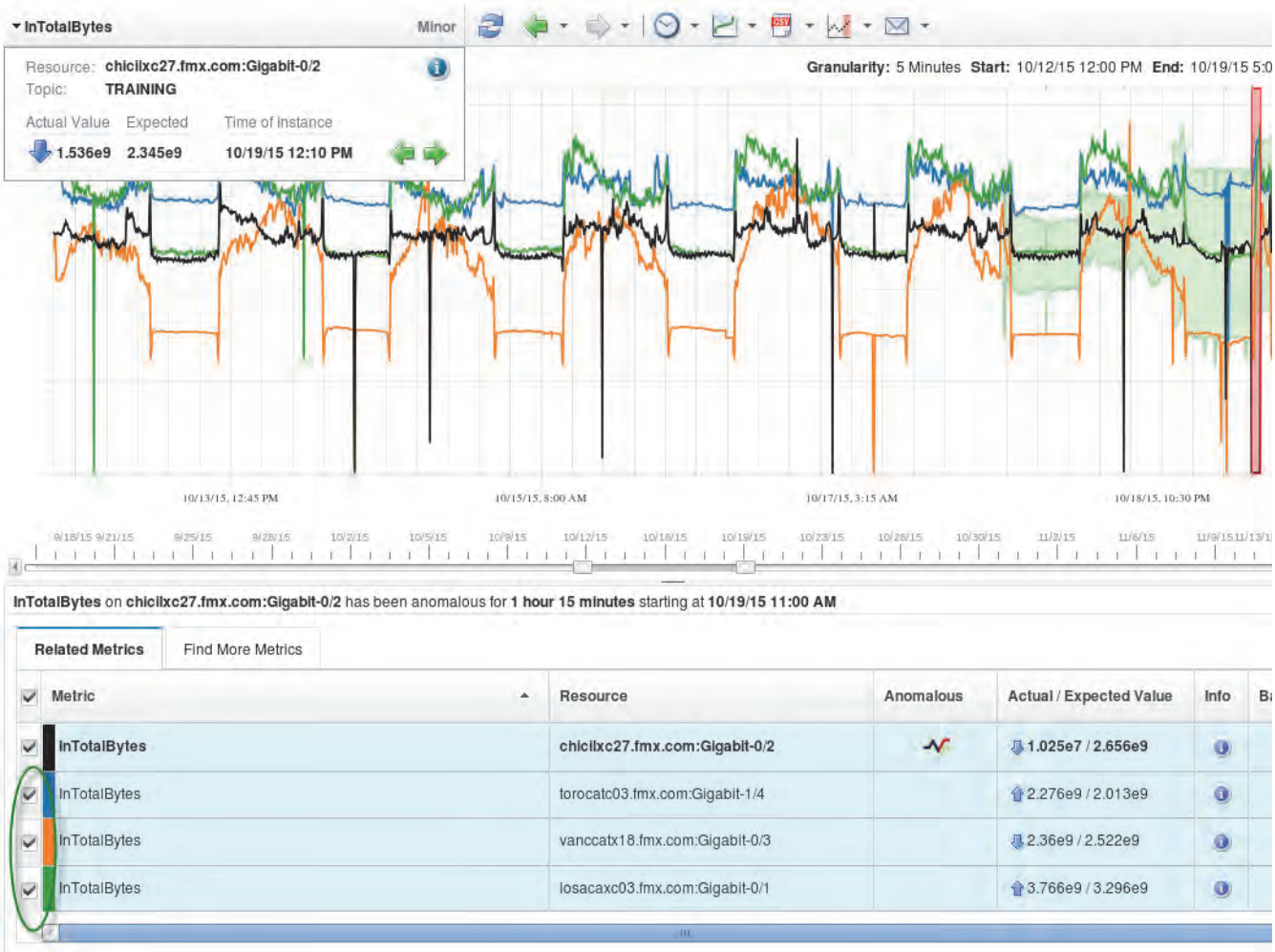
Note the three other metrics in the **Related Metrics** tab.



2. Hold your mouse pointer over the information icons for each of these three metrics and note they are correlated to the metric that you are displaying.

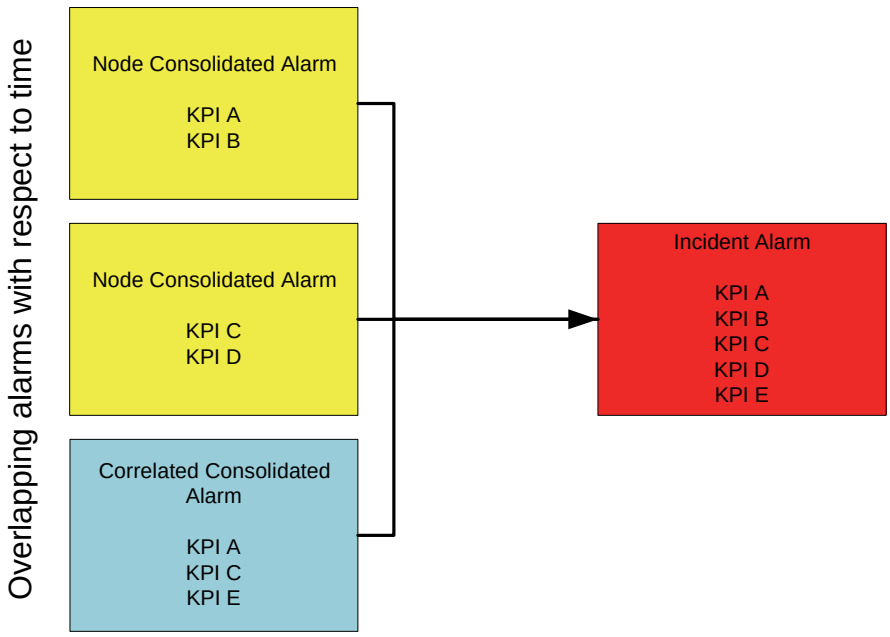


3. Add data associated to the three metrics to the graph. Can you see each KPI has similar features to the original KPI in the graph?

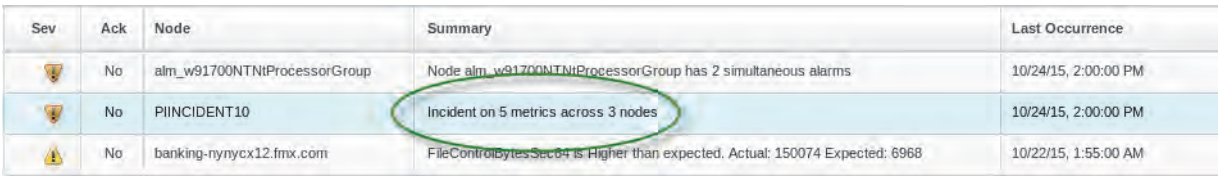


Exercise 10 Reviewing an incident

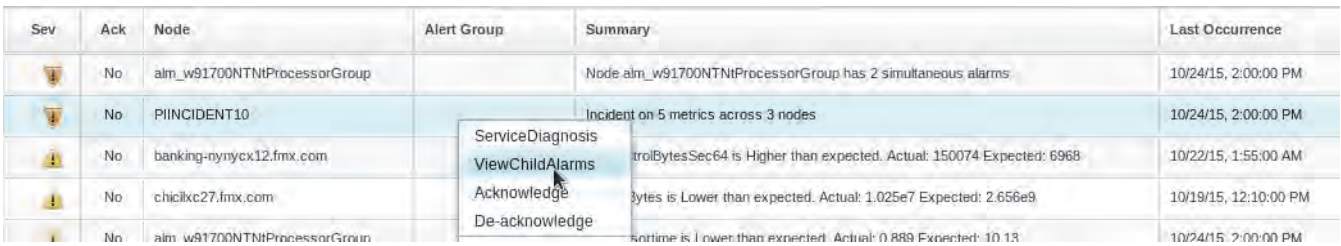
Where it makes sense, Predictive Insights attempts to consolidate alarms. Alarm consolidation occurs when multiple alarms occur on a single node or there is a strong relationship between alarms. In the following exercise you review an incident which is a consolidation of consolidated alarms. Incidents occur when there is two or more consolidated alarms that have overlapping KPIs. In this example, you have multiple alarms occurring on two nodes (creating two node consolidated alarms) and a third consolidated alarm that is occurring across three KPIs that are tightly correlated. All three of these alarms have common threads that look similar to the following.



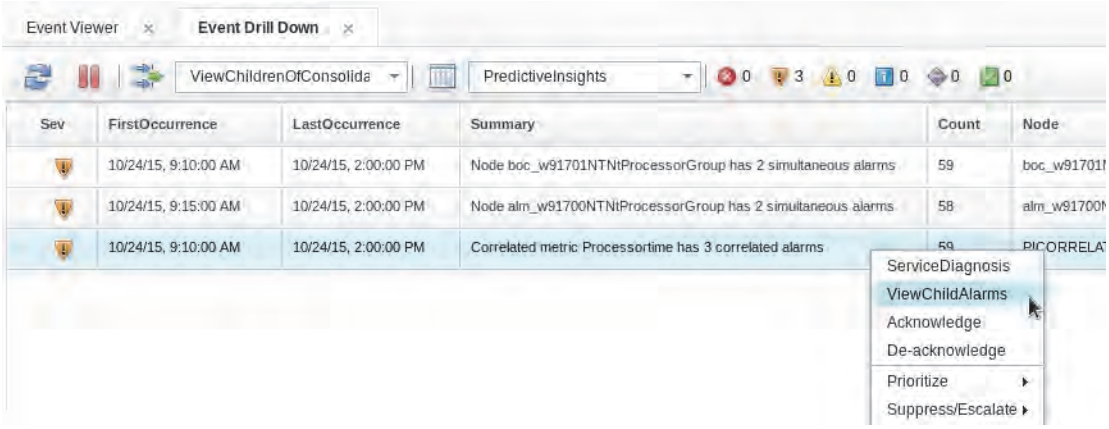
1. Return to the event viewer and search for the **incident** that last occurred on **24 Oct 24** at **2:00 p.m.** Note the summary of the alarm.



2. Since this is a consolidation, the alarm has children that can be viewed. Click right on the alarm and select **ViewChildAlarms**.



3. An incident is a consolidation of consolidated alarms. Each alarm displayed is a consolidation. Select any one of them, click right and select **ViewChildAlarms** to view the individual alarms that Predictive Insights raised.

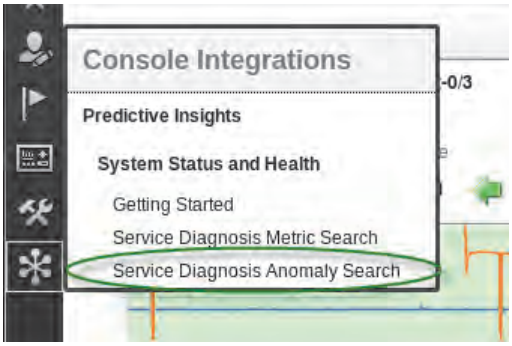


4. Return to the event viewer and select the incident and view it in **ServiceDiagnosis**. Note how all five metrics are displayed. Since the data is highly correlated, it may be easier to view the data in the absolute view.

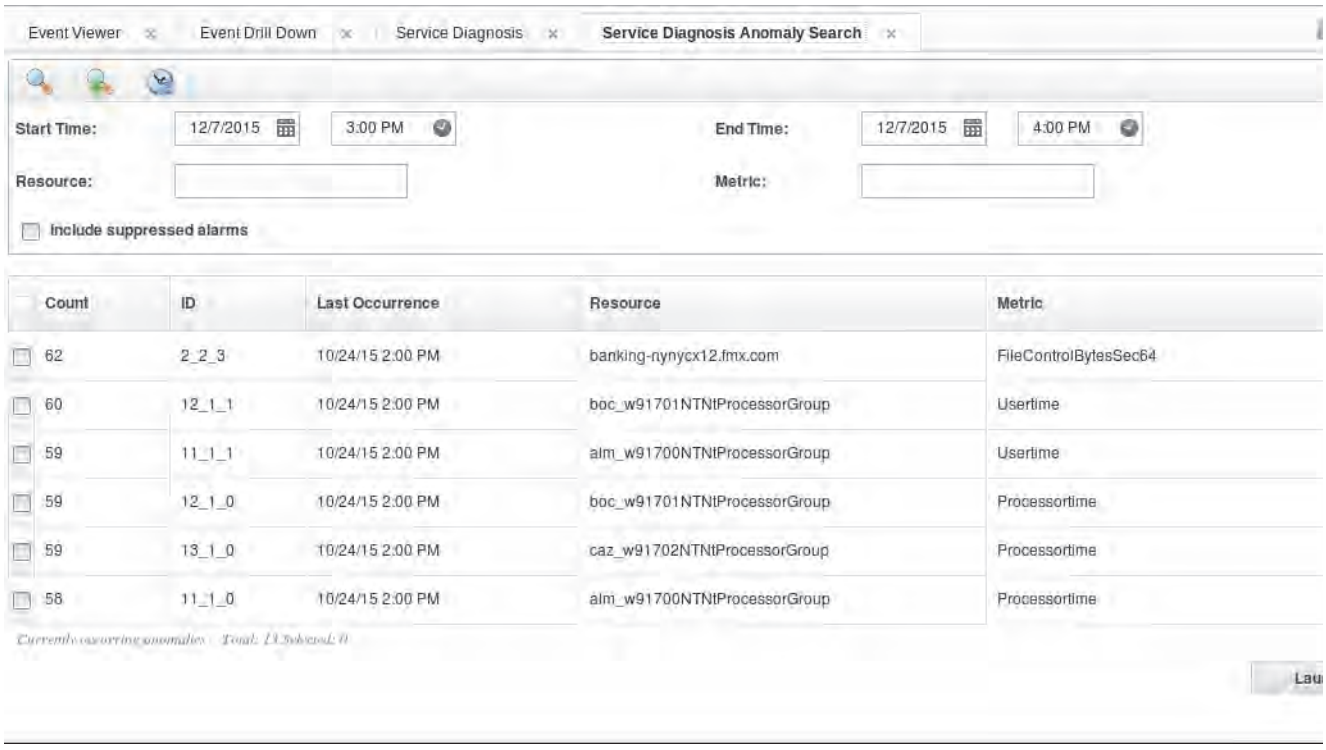
Exercise 11 Searching for anomalies

Reviewing historical anomalies might be necessary to help in troubleshooting existing problems. You can use the Service Diagnosis Anomaly Search tool to search for them. In this exercise, you learn about using this tool.

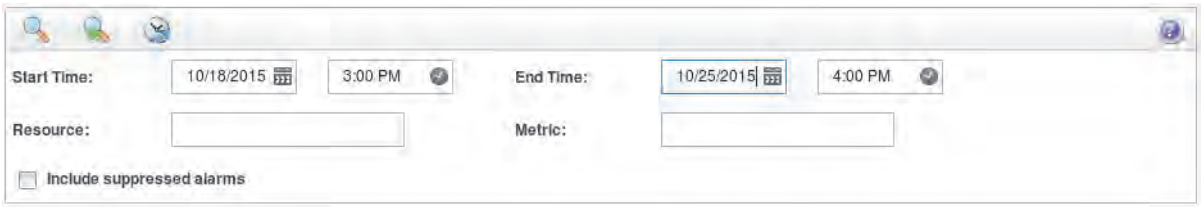
1. View the anomalies that were generated by the resource, **debit-meximxux11.fmx.com**.
   a. Open the Service Diagnosis Anomaly Search tool. Select the snowflake on the left side of the screen and then **Service Diagnosis Anomaly Search**.



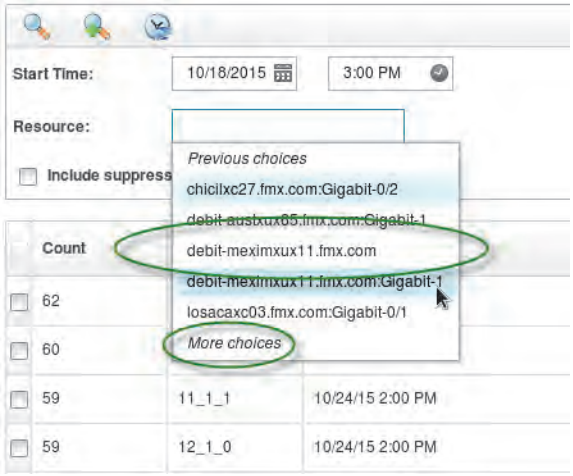
   The system displays the current anomalies, which are from historical data and are not truly current.



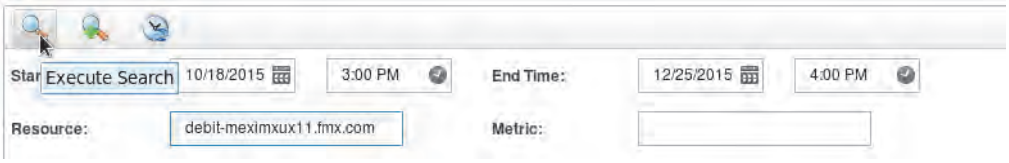
b. Set the **Start Time** to Oct 18, 2015 and **End Time** to Oct 25, 2014. Enter 10/18/2015 and 10/25/2015 in their respective fields. In this instance, the time can remain whatever is displayed.



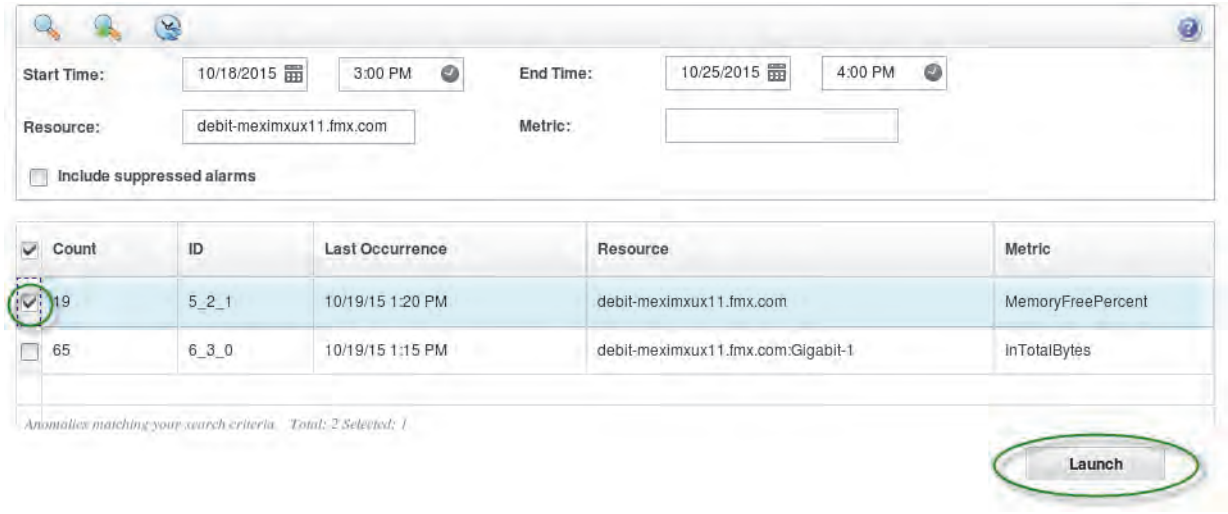
c. Set the resource to debit-meximxux11.fmx.com. Place your cursor in the **Resource** field, and select **debit-meximxux11.fmx.com** from the list of resources. You may have to select **More choices** to find it.



d. Select the magnifying glass to search for an anomaly.



e. Select one or more of the anomalies. Click **Launch** in the lower right of the window or double-click the anomaly to display its details.

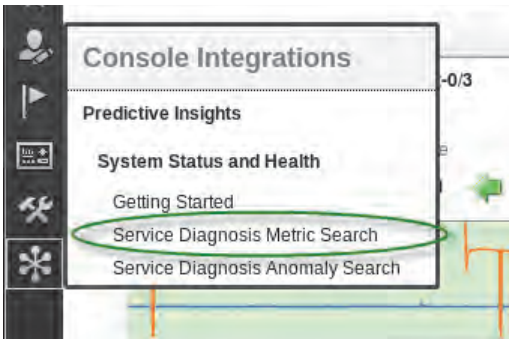


Exercise 12 Searching the metrics

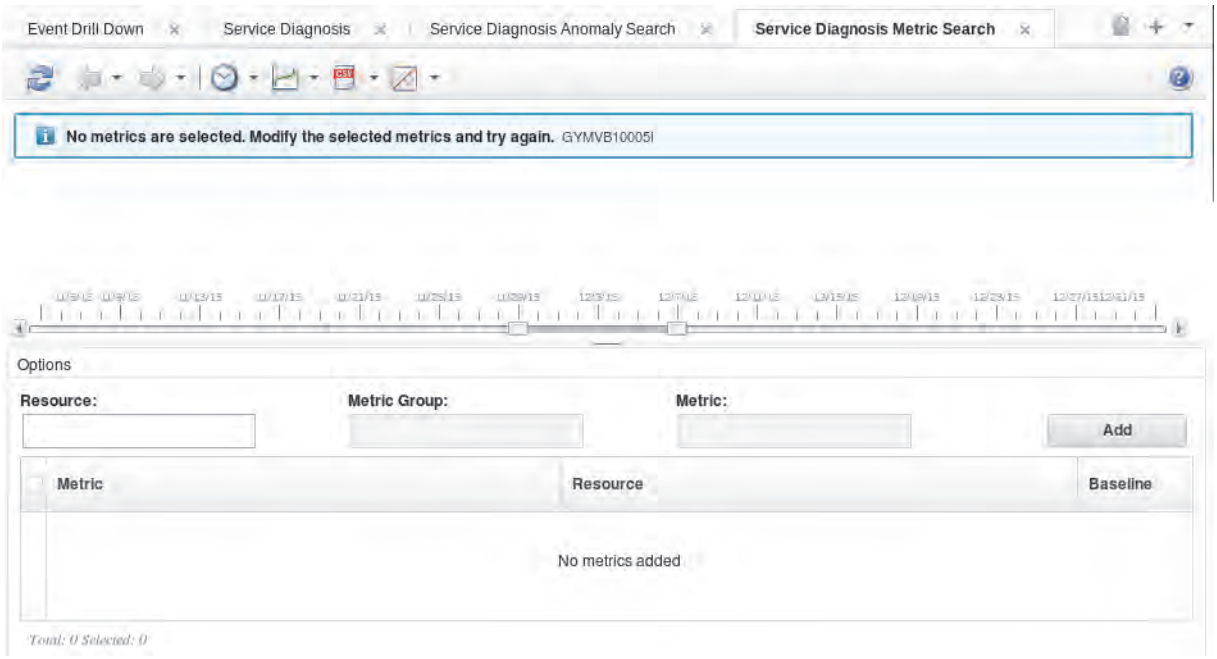
You might want to see the data that is associated to one or more metrics. You can use the Service Diagnosis Metric Search widget to do so. In this exercise, you work with this tool.

Complete the following steps:

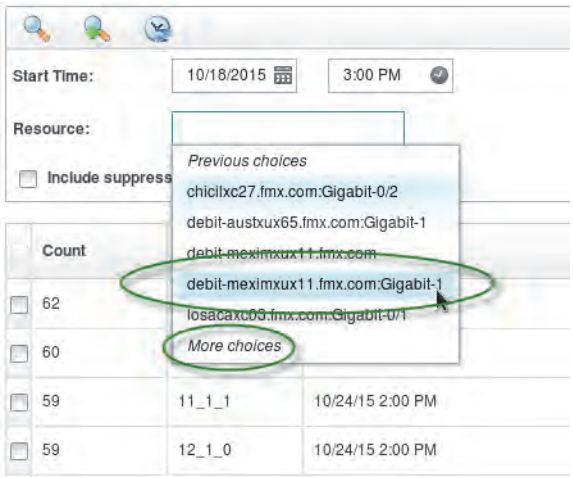
1. Navigate and select the Service Diagnosis Metric Search tool. Select the snowflake on the left side of the screen and **Service Diagnosis Metric Search**.



An empty search pane is displayed.

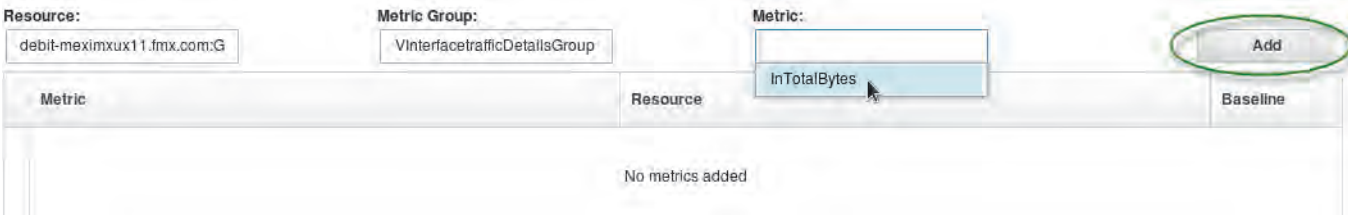


2. Select a set of resources and display some metrics.
   a. Click the **Resource** field, and select **debit-meximxux11.fmx.com:Gigabit-1** from the list.



   b. Click the **Metric Group** field, and select **VInterfacetrafficDetailsGroup.**

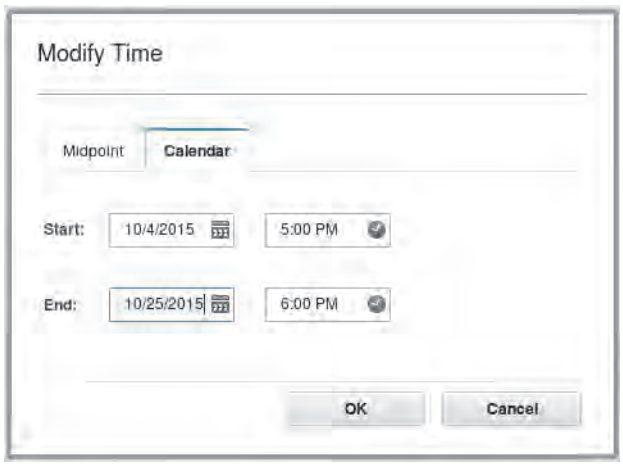
c. Click the **Metric** field and select **InTotalBytes.** Click the **Add** button to add this metric to the list.



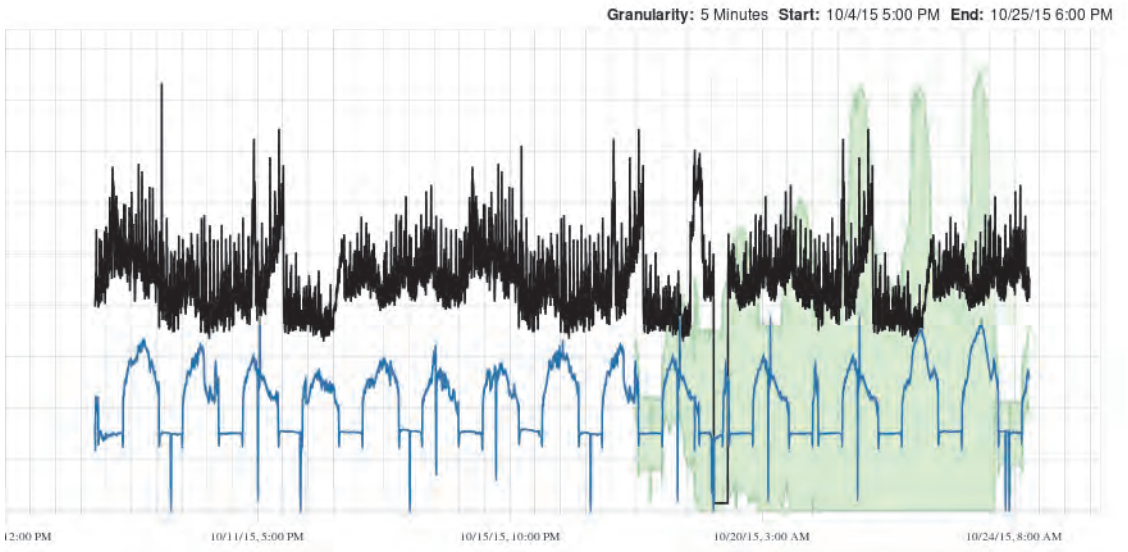
3. Repeat this process for the Resource **vanccatx18.fmx.xom:Gigabit-0/3** by using the same metric group and metric. You need to clear the **Resource** field to begin. Once you add this metric, select **vanccatx18.fmx.xom:Gigabit-0/3** to display its baseline. Select the check boxes to have the values displayed.

4. Set the time frame from Oct 4, 2015 to Oct 25, 2015. Click the clock in the **Metric Search** main menu.

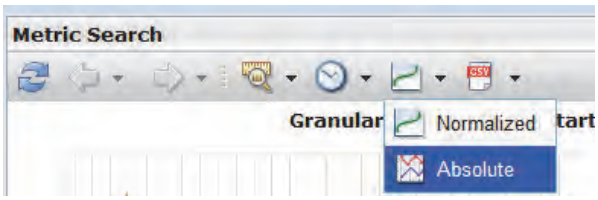
5. In the Modify Time window, click the **Calendar** tab. Select a **Start** date of 10/4/2015, and an **End** date of 10/24/2015. Leave the default times. Click **OK**.



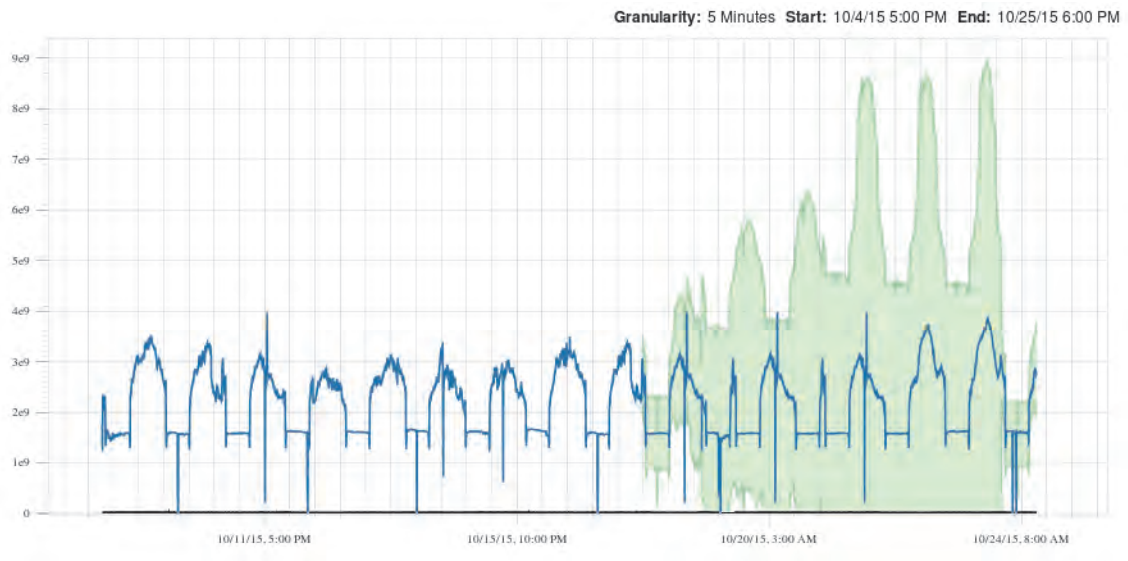
6. Display the data in both normalized and absolute mode. When multiple metrics are selected, the graph is normalized by default, and all metrics are easier to see. Note that no units are on the vertical axis.



7. Select the **Absolute** mode from the **Metric Search** menu.



Note the difference in the displayed data now that units are on the vertical axis. Note the changes in the chart when the **debit-meximxux11.fmx.com:Gigabit-1** is display in absolute mode.

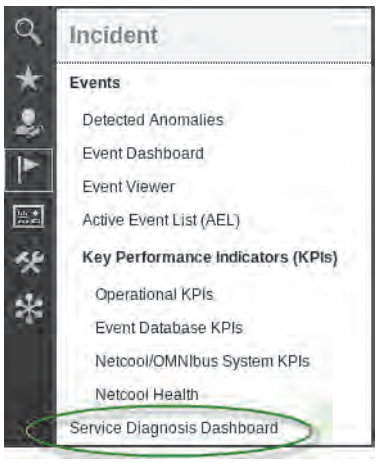


Exercise 13 Viewing the Service Diagnosis Dashboard

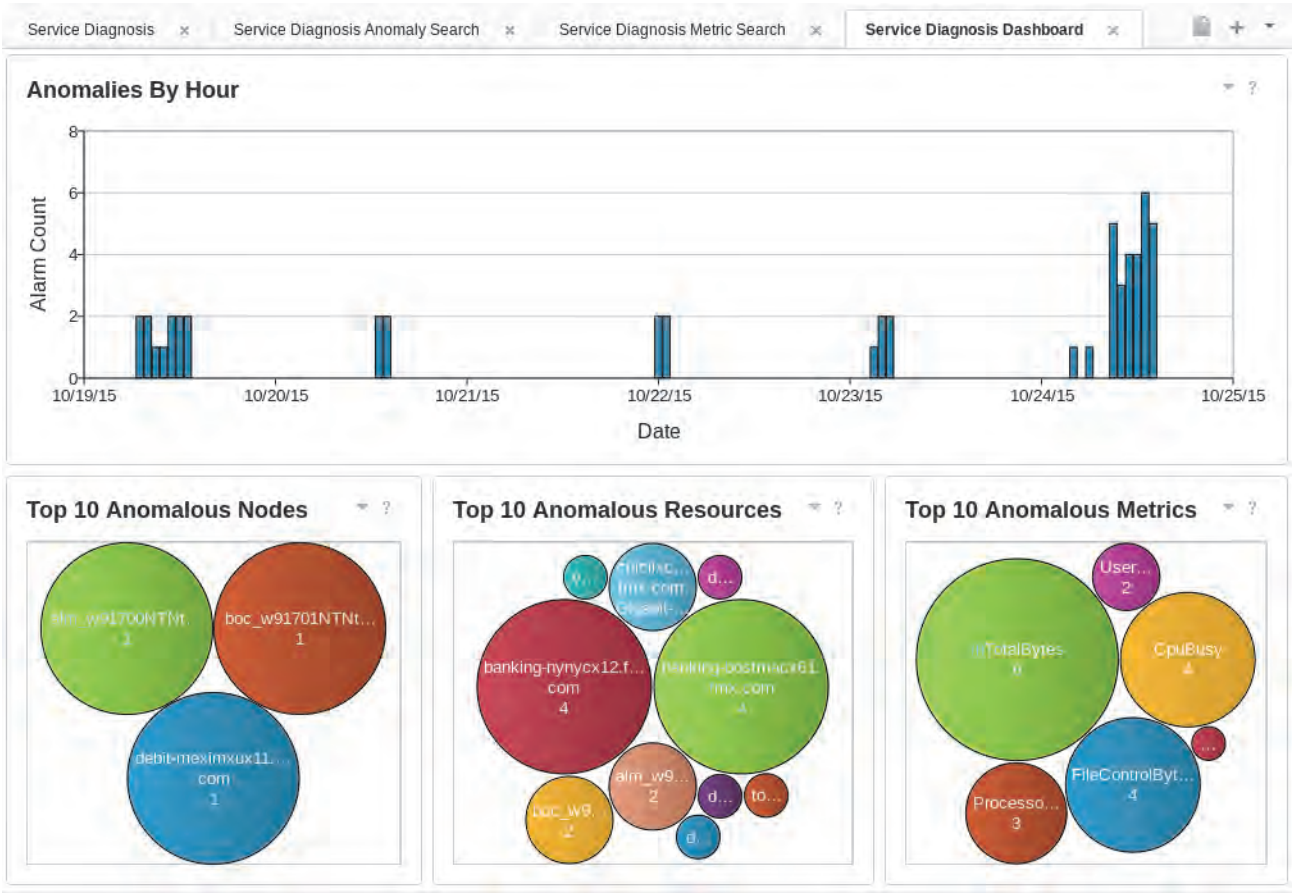
In this exercise, you view the Service Diagnosis Dashboard. This dashboard gives you an overall view of the anomalies that have recently occurred.

1. Click the **Incident** menu on the left side of the Dash interface.

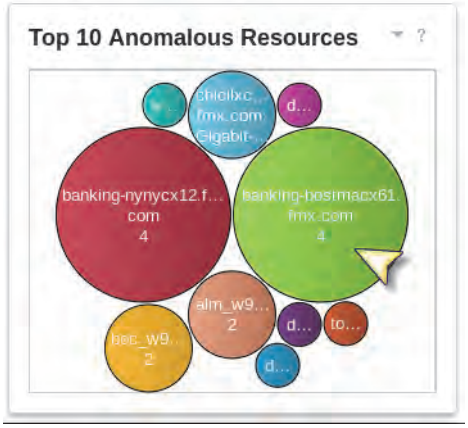
2. Select the Service Diagnosis Dashboard.

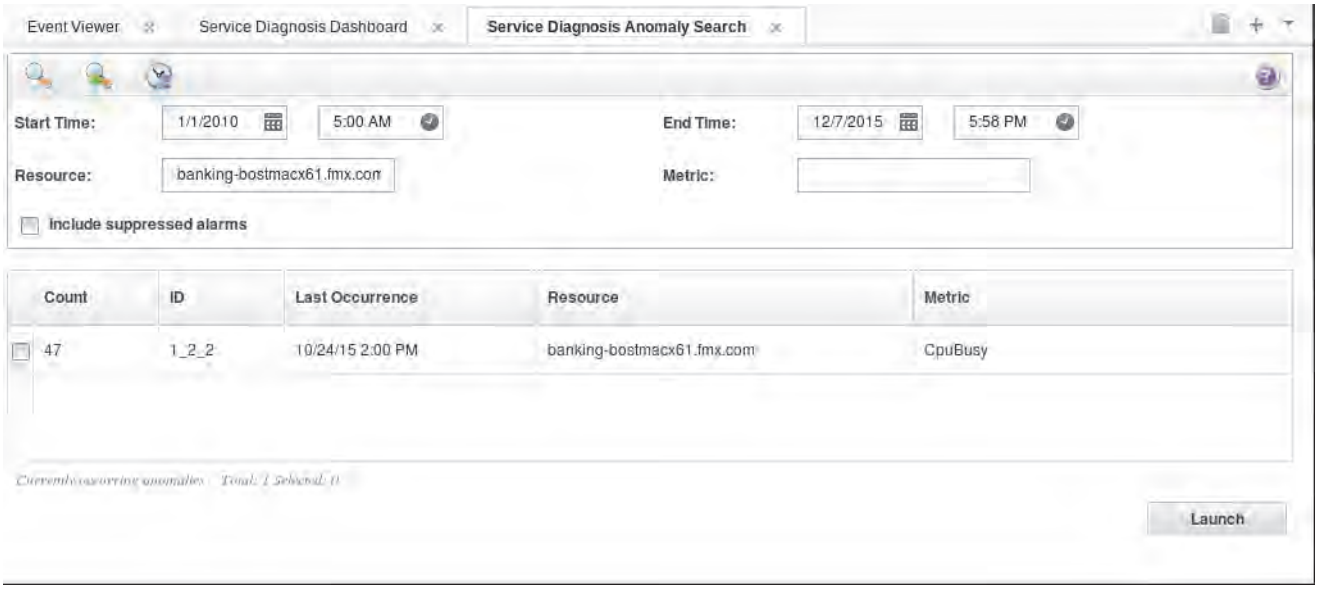


3. Take some time and review the interface. The **Top 10 Anomalous Nodes** references host names. A resource (like the **Top 10 Anomalous Resources**) can either be a host name or a subcomponent of a host name, like an interface.



4. Double click into one of the bubbles to see details.





Unit 6 Administration and troubleshooting exercises

This unit has no student exercises.

Unit 7 Advanced mediation techniques exercises

Simple logstash operations

In this set of exercises, your introduction to logstash includes building a simple configuration file to work with a log file that is a combination of useful metric data and log messages that must be removed. For testing purposes, you stream messages to standard input, parse and modify each message, and then sending the results to standard output. Because you are using standard input, you use the **head** command and pipe the data stream into **logstash** with a command similar to this one:

```
[scadmin@scapi]$ head -100 sample.log | logstash -f sample.conf
```

Note: With the **head -100 sample.log** command, you publish the first 100 lines of the **sample.log** file. The **logstash -f sample.conf** command starts logstash, loads the **sample.conf** file and then processes each of the 100 commands that have been piped to it.

In all of the following exercises, you need an editor to create the logstash conf files. These conf files tell logstash how to get, modify, and publish messages. Logstash uses many curly and square braces in its language. An editor that highlights matching braces is very helpful. The following examples use the GNU Emacs editor. The student can choose any of the other editors that are installed by default on the Linux operating system, for example, vi or gedit.

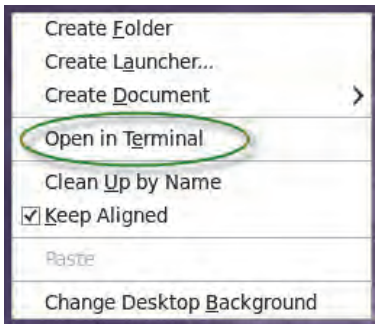
The conf files for these exercises can be found in the appendix.

Exercise 1. Creating simple conf file

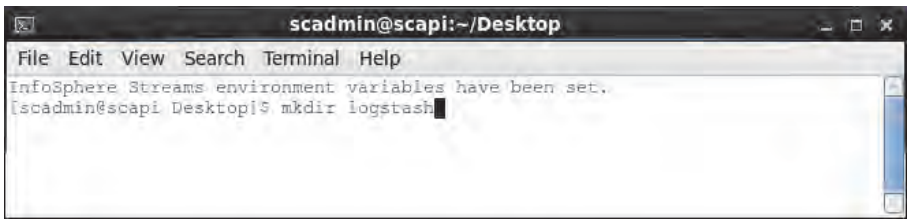
Here you create a simple logstash conf file and use it manipulate a flat file.

1. Log in to the Predictive Insights Linux server as the user **scadmin** with the password **object00**.

2. Open a terminal window.
   a. Right-click the scadmin desktop and select **Open in Terminal**.



3. Create a directory for your logstash conf files on the scadmin Desktop.
   a. In the terminal you just opened, enter this command:
      `mkdir /home/scadmin/Desktop/logstash`



4. In the newly created logstash directory, create a logstash conf file and name it **mixed-skinny-data.conf**.
   a. Change to the logstash directory by entering this command:
      `cd /home/scadmin/Desktop/logstash`

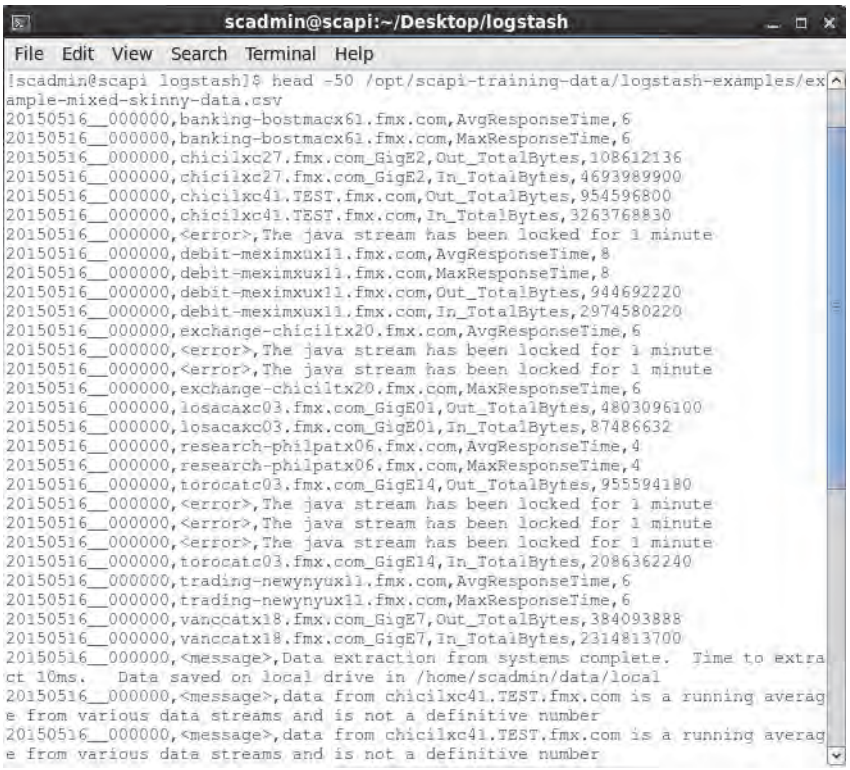
b. Create a new file called **mixed-skinny-data.conf** with this command:

```
touch mixed-skinny-data.conf
```

5. Review the format of the example data file located at this location: **/opt/scapi-training-data/logstash-examples/example-mixed-skinny-data.csv**

   a. In the terminal window, display the contents of the **example-mixed-skinny-data.csv** file with the **head** command.

   ```
   head -50
   /opt/scapi-training-data/logstash-examples/example-mixed-skinny-data.csv
   ```



   b. Take a moment and review the data in this file. In a production setting, you would work with a systems expert who could provide you details on what the log contains. Note the following items:

      i. Two types of messages; metric data and log messages

      ii. All messages have a time stamp

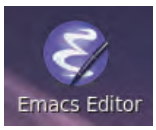
      iii. Log messages have a message type of either <message> or <error> header

iv. Metric data have a format of a host name, metric name, and a metric value (that is, the data is skinny)

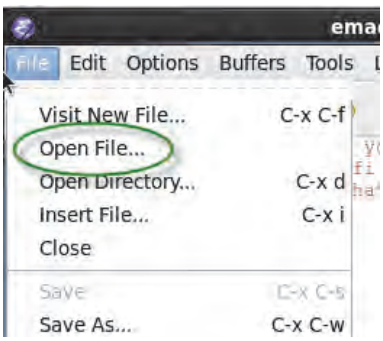
v. All items are comma delimited

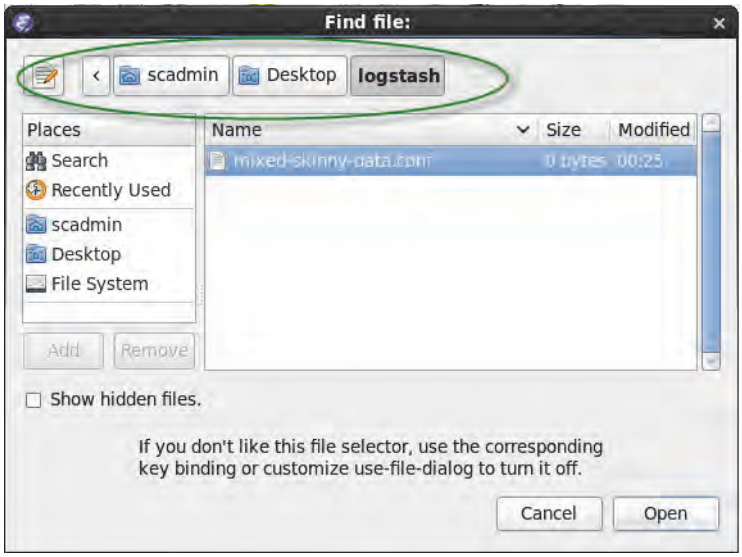
You should be able to see the skinny nature of this file once all the messages are removed.

6. Edit the **mixed-skinny-data.conf** file to take input and echo it.
   a. Open Emacs by double-clicking the **Emacs Editor** icon on the scadmin desktop.



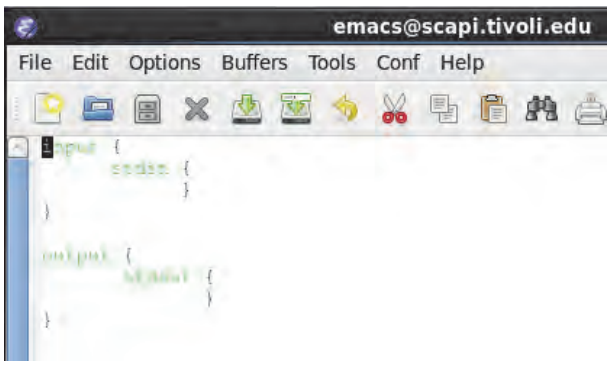
   b. Open the **mixed-skinny-data.conf** file in the Emacs editor by selecting **File** > **Open File** and navigating to **scadmin** > **Desktop** > **logstash.** Click **Open**.





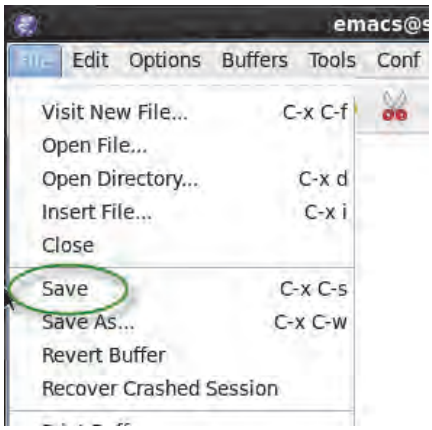
c. Enter the following text into the mixed-skinny-data.conf buffer. Use the tab key to help with useful indentation.

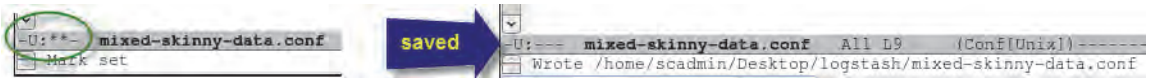
```
input {
   stdin {
   }
}

output {
   stdout {
   }
}
```



Note: **stdin** and **stdout** are logstash plug-ins that take data from the standard input device (the terminal screen) and publish it to standard out (the terminal screen).

d. Save the file by clicking **File > Save** or pressing Ctrl+x Ctrl+s on the keyboard. Note the lower portion of the editor changes when the file is saved.



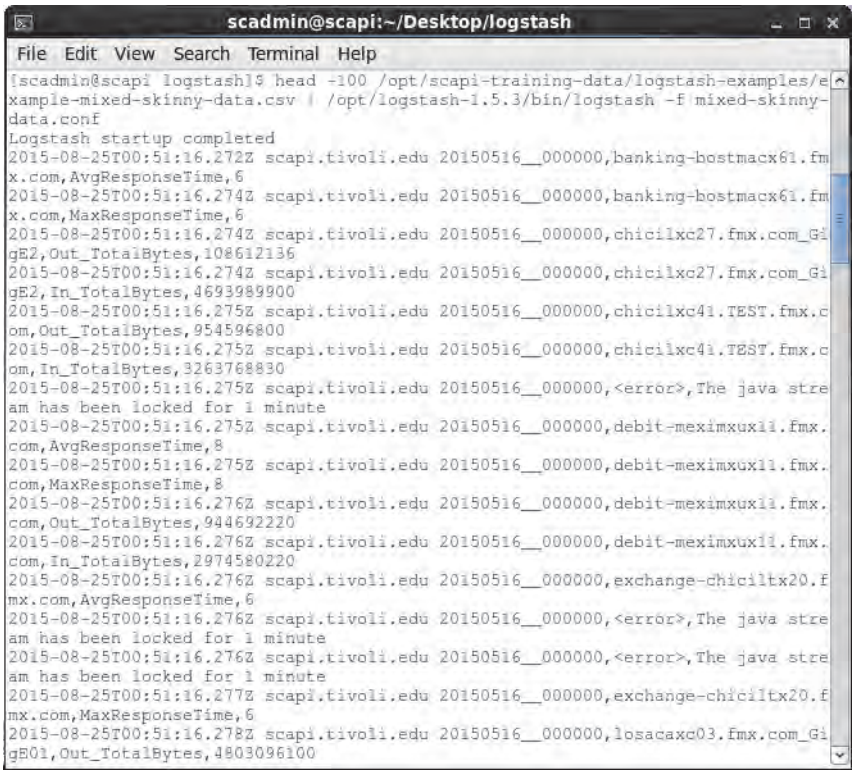


e. Test this CONF file by pushing the data in the **example-mixed-skinny-data.csv** file through it using standard input. You accomplish this process by using the head command to display the data in standard input and then piping it to the logstash command.

   i. In the open terminal, enter the following command:

   ```
   head -100
   /opt/scapi-training-data/logstash-examples/example-mixed-skinny-data.csv
   | /opt/logstash-2.1.0/bin/logstash -f mixed-skinny-data.conf
   ```

Note the output of the command. It might take several seconds for logstash to start and accept the input.



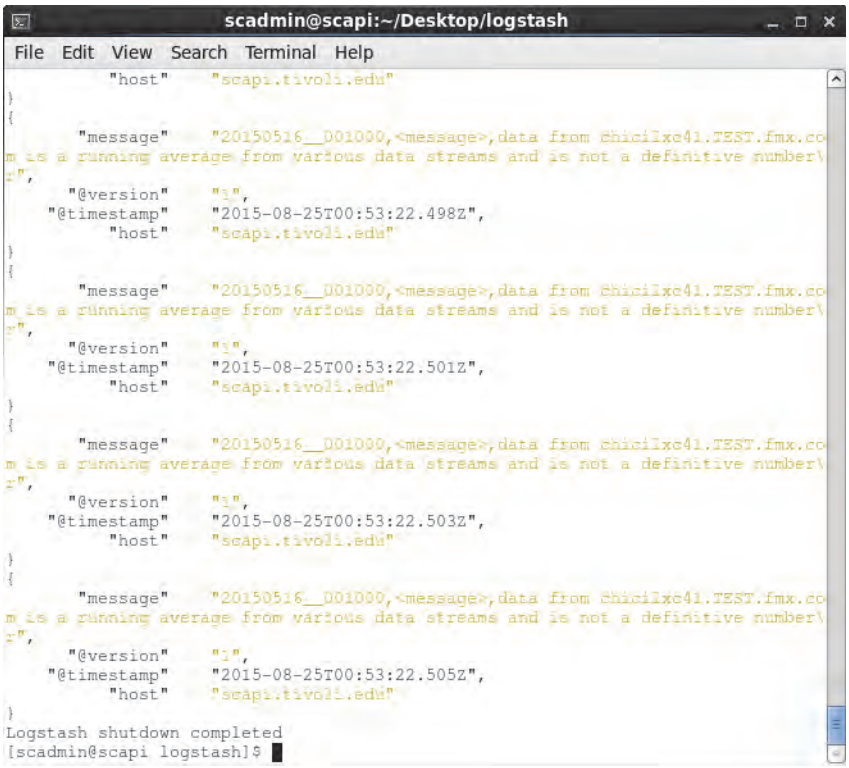
Note: Review the output. Two things were added to the output. There is a time stamp and the server name where the message was processed. Notice the difference between the two time stamps. One was included in the data file and the other was added during the processing.

You might also have noted that there is a delay after entering the command. Logstash gets compiled at runtime and then begins processing the command to standard input, thereby delaying the output.

7. View metadata surrounding messages in logstash using rubydebug codec.
   a. In the Emacs editor, add the **rubydebug** codec to the **stdout** plug-in.

```
output {
   stdout {
      codec => rubydebug {}
   }
}
```

b. Save the **mixed-skinny-data.conf** file and rerun the previous command to test this change. Use the up-arrow key in the terminal window to restore the previous command. Note the difference in the output. Another popular output codec would be **json**.



8. Add a tag to the message by using a filter.

   In this step, you add metadata to the message. You can add tags at any point during the processing of a message. They are saved in a array where you can use them for processing and debugging purposes, as shown in an upcoming exercise. In this example, you use the **mutate** filter to add a tag to each event that passes the filter. With this filter, you can perform general mutations on the fields in your messages Currently you have only four fields. You can rename, remove, replace, and modify fields in your events. Details on this filter can be found at this location:

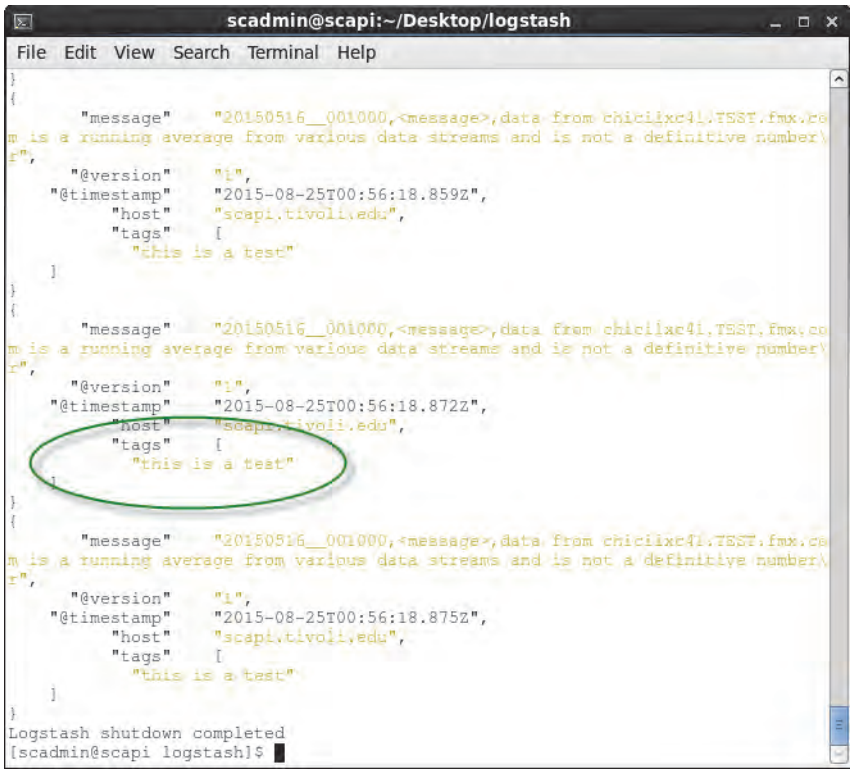
   `https://www.elastic.co/guide/en/logstash/current/plugins-filters-mutate.html`

   a. Add the following filter commands to the **mixed-skinny-data.conf** file. Note that the input, output, and filter areas of the CONF file are completely independent of each other with respect to curly braces, that is, they are not nested.

```
input {
   stdin {
   }
}

filter {
   mutate {
      add_tag => "this is a test"
```

```
    }
}

output {
    stdout {
        codec => rubydebug
    }
}
```

b. Save the **mixed-skinny-data.conf** file and rerun the previous command to test this change. Note the newly added field and the string that was added to it.



Exercise 2. Using IF commands

In this exercise you begin adding conditionals to the code to allow it to find useful and non-useful messages in the log. This logic is implemented via IF commands. Using IF-THEN-ELSE types of logic, you can begin adding useful functions to your configuration file. Here you use IF logic and the MUTATE function to tag messages that contain information that is not useful. The messages you consider non-useful are ones that have the <error> or <message> tags in their text.

For more information on logstash conditionals, refer to the following URL:

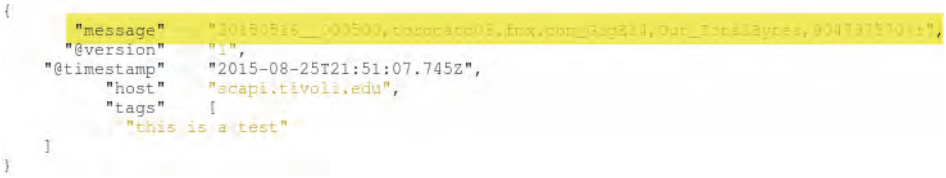
`https://www.elastic.co/guide/en/logstash/current/event-dependent-configuration.html#conditionals`

Complete the following steps.

1. Add the following **if** filter commands to the **mixed-skinny-data.conf** file.

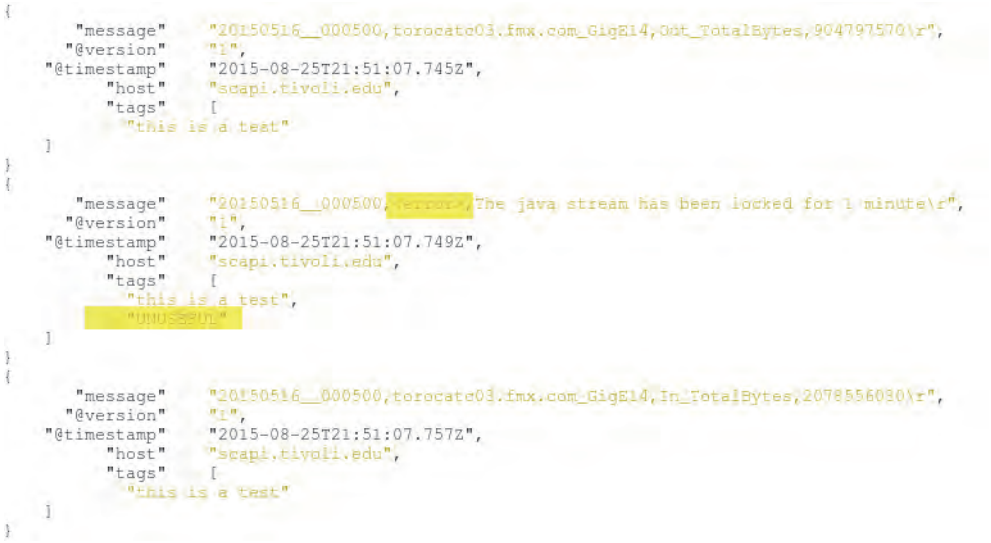
```
input {
   stdin {
   }
}
filter {
   mutate {
      add_tag => "this is a test"
   }
   if [message] =~ /.*<error>.*/ or [message] =~ /.*<message>.*/ {
      mutate {
        add_tag => "UNUSEFUL"
        }
   }
}
output {
   stdout {
      codec => rubydebug {}
   }
}
```

Note: Looking at the structure of the IF statement, you can see it is referencing the attribute **message**. This attribute belongs to the event that logstash has received as input. Refer to the earlier output.



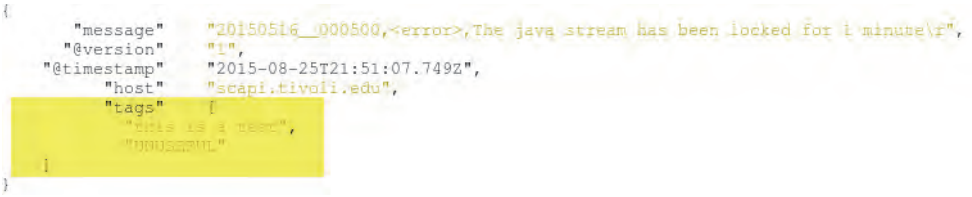
The IF structure is checking for the string **<error>** or **<message>** in the message itself. If it finds either one, it appends to the tags the word **UNUSEFUL**.

2. Save the **mixed-skinny-data.conf** file and rerun the previous command to test this change. Search through the output in the terminal and look for the log messages that are not metrics and see they were tagged as UNUSEFUL.



Exercise 3. Using the drop plug-in

Now that you have tagged which data is useful and which is not, you can further filter the results by deleting that which you are not interested in. For example, you can use the **drop** plug-in to delete messages that you have no interest in. In this case, you drop the messages that have **<error>** or **<message>** in them. You inspect the tags that are associated to the message and use “UNUSEFUL” as the key.



For more information on drop, refer to the following URL:

`https://www.elastic.co/guide/en/logstash/current/plugins-filters-drop.html`

Complete the following steps.

1. Add the following **if** filter commands to the **mixed-skinny-data.conf** file.

```
input {
   stdin {
   }
}
```

```
filter {
   mutate {
      add_tag => "this is a test"
   }
   if [message] =~ /.*<error>.*/ or [message] =~ /.*<message>.*/ {
      mutate {
        add_tag => "UNUSEFUL"
      }
   }
   if "UNUSEFUL" in [tags] {
      drop {}
   }
}
output {
   stdout {
      codec => rubydebug {}
   }
}
```

Note: This code is not the most efficient use of conditionals. The best option was not to tag a message as **UNUSEFUL** but to **drop** it immediately.

2. Save the **mixed-skinny-data.conf** file and rerun the command to test this change. Search through the output in the terminal. All the messages should be related to metrics only.

Exercise 4. Giving the messages structure

Now that you have only the messages that are important, you can give these messages structure. As you noticed earlier, the messages have the following format:

```
<time-stamp>,<hostname>,<metric-name>,<metric-value>
```

Use this format with the **csv** plug-in to separate this data into individual fields that allow use to better work with its information. Details on the **csv** plug-in can be found at the following URL.

```
https://www.elastic.co/guide/en/logstash/current/plugins-filters-csv.html
```

Complete the following steps.

1. Add the following **csv** filter commands to the **mixed-skinny-data.conf** file.

```
input {
   stdin {
   }
}
filter {
   mutate {
      add_tag => "this is a test"
   }
   if [message] =~ /.*<error>.*/ or [message] =~ /.*<message>.*/ {
      mutate {
        add_tag => "UNUSEFUL"
      }
   }
   if "UNUSEFUL" in [tags] {
      drop {}
   }
   csv {
      columns => ["timestamp","hostname","metric-name","metric-value"]
      add_tag => "added fields to CSV data"
   }
}
output {
   stdout {
      codec => rubydebug {}
   }
}
```

Note: See the tag that was added with **csv** plug-in. Most plug-ins allow the use of add_tag in their definitions. You can use these tags as debugging steps as you are building your messages. Such debugging can be useful if you are using complex conditional logic.

2. Save the **mixed-skinny-data.conf** file and rerun the command to test this change. All the messages should include the **timestamp**, **hostname**, **metric-name**, and **metric-value** fields.

Exercise 5. Sending data to a file

Logstash cannot push data directly into Predictive Insights. Your only alternatives are to push the data into a database or into a flat file that Predictive Insights can then read. Predictive Insights can push data directly into Log Analysis, but that task is not addressed in this lab. In this exercise, you push the metric data into a flat file that could potentially read by Predictive Insights.

There is a output plug-in that is also called **csv** which is designed to create a comma-delimited file. Details on the **csv** plug-in can be found at the following URL.

`https://www.elastic.co/guide/en/logstash/current/plugins-outputs-csv.html`

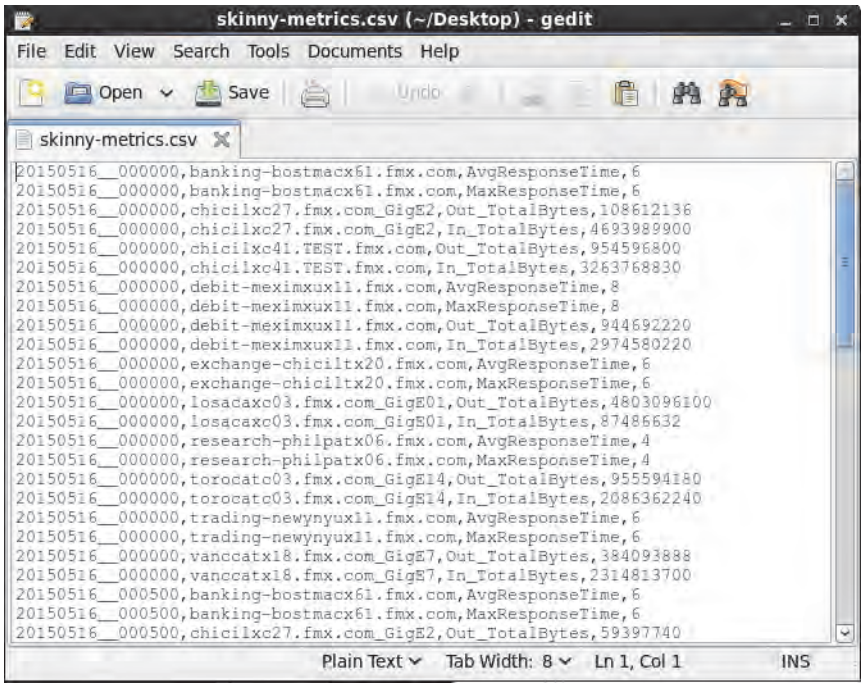
Complete the following steps.

1. Add the following **csv** filter commands to the **mixed-skinny-data.conf** file.

```
input {
   stdin {
   }
}
filter {
   mutate {
      add_tag => "this is a test"
   }
   if [message] =~ /.*<error>.*/ or [message] =~ /.*<message>.*/ {
      mutate {
        add_tag => "UNUSEFUL"
```

```
        }
    }
    if "UNUSEFUL" in [tags] {
        drop {}
    }
    csv {
        columns => ["timestamp","hostname","metric-name","metric-value"]
        add_tag => "added fields to CSV data"
    }
}
output {
    stdout {
        codec => rubydebug {}
    }
    csv {
        fields => ["timestamp","hostname","metric-name","metric-value"]
        path => "/home/scadmin/Desktop/skinny-metrics.csv"
    }
}
```

2. Save the **mixed-skinny-data.conf** file and rerun the command to test this change. A file should now be on the Desktop called **skinny-metrics.csv**. Click right on this new file and select **Open with gedit**. Review the file.



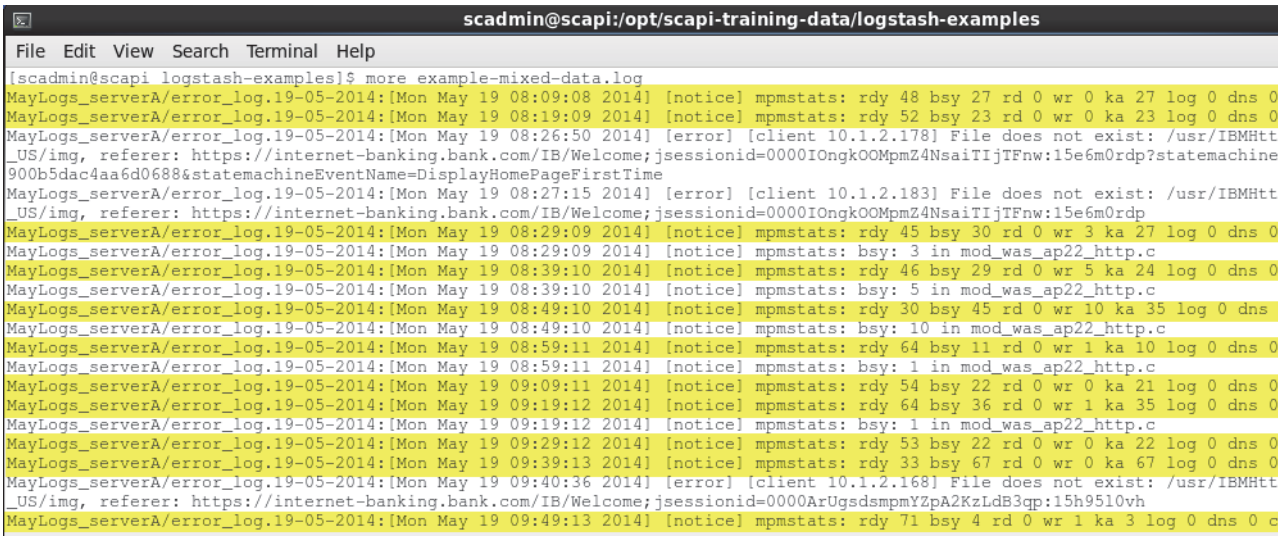
Important: The CSV file you created is not ready for consumption by Predictive Insights. There are two important elements missing. The column names have not been defined in the first line of the file and the file name does not include a start time and end time for the data it encapsulates. You are introduced to a custom plug-in in the next set of exercises to help solve this problem.

Using the grok plug-in

In the following set of exercises you are introduced to the **grok** plug-in. Grok could be the most powerful and useful of all the plug-ins that logstash offers. Using a combination of predefined regular expression patterns and your own, you can use the grok plug-in to add structure to unstructured data.

Similar to the previous exercise, you work with a log that has a mixture of both useful and nonuseful data for Predictive Insights. You filter out the nonuseful data. What is left is data that is useful but not comma delimited like the previous example. You can use grok to parse this data and associate fields to the data that needs to be tracked.

Here is the data in the log that you work with. The useful data is highlighted in yellow.



You are also introduced to a custom plug-in that was developed specifically for Predictive Insights. You use it to produce a CSV file that meets the Predictive Insights requirements for file naming and file format.

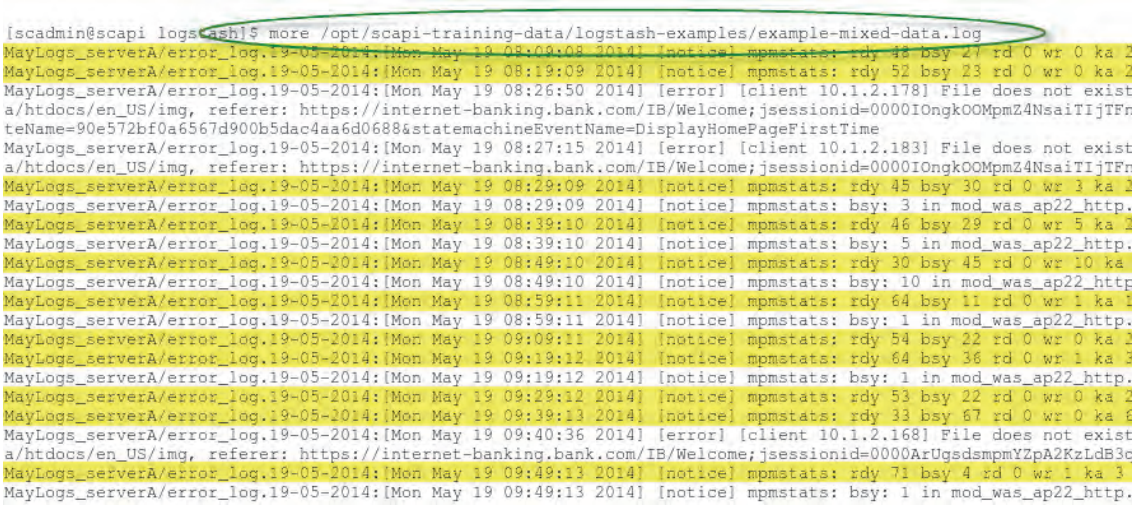
Exercise 6. Filtering the log file

In this exercise, you filter the log file in same manner that you did earlier with an IF statement and drop plug-in.

Complete the following steps.

1. View the format of the **example-mixed-data.log**.
   a. In the terminal window, view the data in the **/opt/scapi-training-data/logstash-examples/example-mixed-data.log** using the **more** command. The lines that contain useful performance data are highlighted in yellow.

      ```
      more /opt/scapi-training-data/logstash-examples/example-mixed-data.log
      ```



      Note the format of the message that is highlighted.

      ```
      MayLogs_serverA/error_log.19-05-2014:[Mon May 19 08:29:09 2014] [notice]
      mpmstats: rdy 45 bsy 30 rd 0 wr 3 ka 27 log 0 dns 0 cls 0
      ```

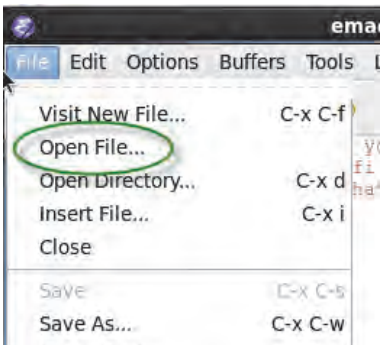
      The message includes the following items:

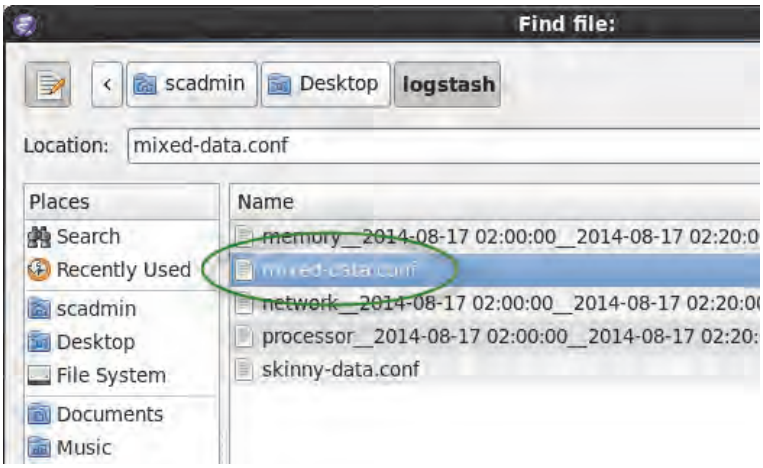
      - Host name: serverA
      - Time stamp: `19-05-2014 at 08:29:09`
      - Performance data: `rdy 45 bsy 30 rd 0 wr 3 ka 27 log 0 dns 0 cls 0`

2. Create a conf file called **mixed-data.conf**
   a. Like you did in the previous exercise, use the **touch** command in the terminal window to create an empty file in the /home/scadmin/Desktop/logstash directory:

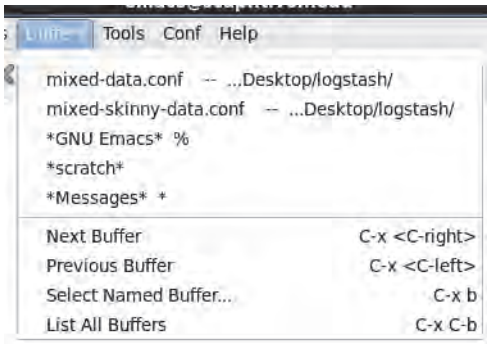
      ```
      cd /home/scadmin/Desktop/logstash
      touch mixed-data.conf
      ```

b. Open the **mixed-data.conf** file in Emacs clicking **File** > **Open File**.





Note: Opening the second file in Emacs creates a new buffer. Both files are now open in the editor, and you can switch between the two using the **Buffers** menu:



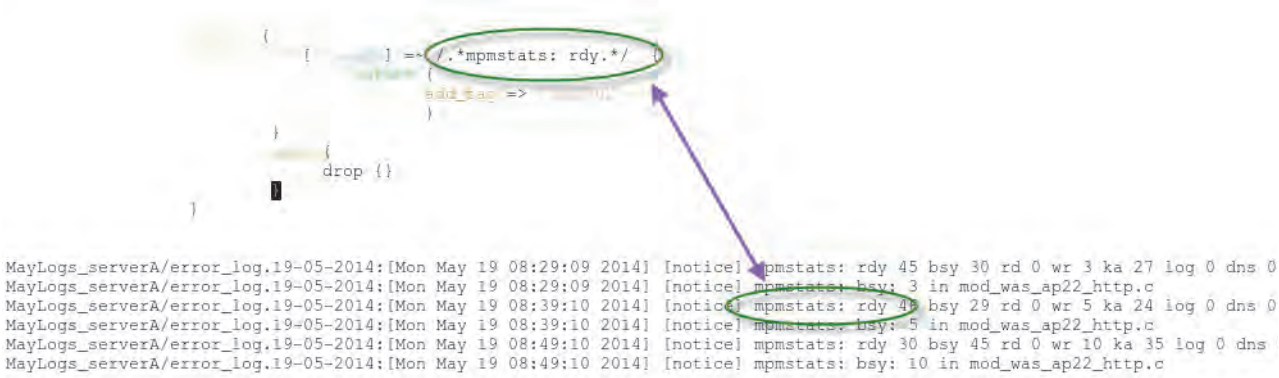
3. Create a filter that compares the correct message in the file. The first challenge to extracting the data from the file is making sure you process only the messages that are interesting. You can use and IF statement in the filter section to accomplish this task.
   a. As in the previous example, start with a simple framework for entering and publishing data. Enter the following text into the **mixed-data.conf** file.

```
input {
   stdin {
   }
}

filter {
}

output {
   stdout {
      codec => rubydebug
   }
}
```

   b. Enter an IF statement to extract only messages that have the string **mpmstats: rdy** in them. If that string is missing, drop the message to prevent further processing. Add the following code to the filter section.

```
input {
   stdin {
   }
}

filter {
   if [message] =~ /.*mpmstats: rdy.*/  {
      mutate {
      add_tag => "USEFUL"
      }
   }
   else {
      drop {}
   }
}

output {
   stdout {
      codec => rubydebug
   }
}
```

Note: Here you see the regular expression in the IF command matching the string in the message.



c. Save the file in Emacs.

d. Test the filter you made with the following command in the terminal window. Note the change in the configuration and example log file names.

```
head -100 /opt/scapi-training-data/logstash-examples/example-mixed-data.log
| /opt/logstash-2.1.0/bin/logstash -f mixed-data.conf
```

Exercise 7. Parsing data and adding structure

You now must parse useful information from the message. The **grok** plug-in is useful in adding structure to the messages. Using a grok debugger is helpful in this case. One is available at https://grokdebug.herokuapp.com.

For details on how to use grok, you can reference the following URL:

`https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html`

You can find a useful YouTube video about the grok debugger used in the subsequent exercise here:

`https://www.youtube.com/watch?v=YIKm6WUgFTY`

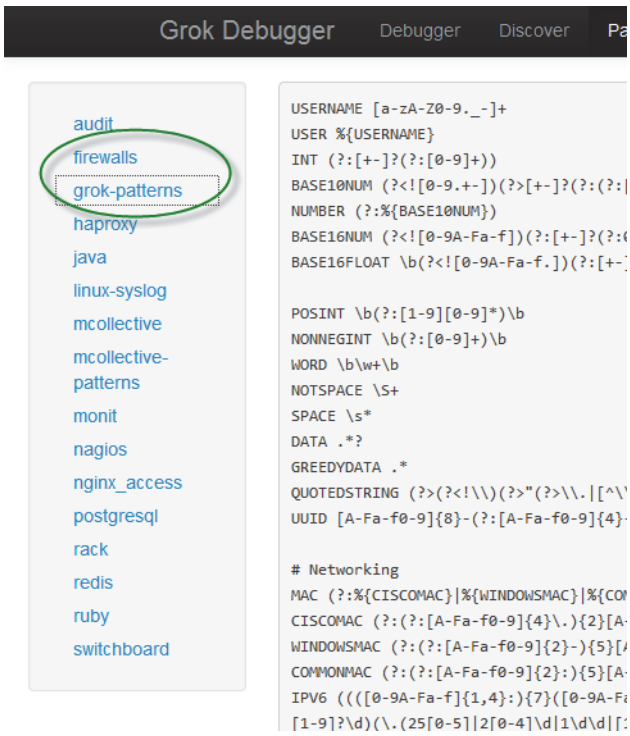
To help in understanding the grok method of parsing, take a simple set of text.

`55.3.244.1 GET /index.html 3.44`

The syntax for a grok pattern is **%{SYNTAX:SEMANTIC}.**

The **SYNTAX** is the name of the pattern that matches the text you are interested in. There are many predefined syntaxes you can use. For example, 3.44 could be matched by the **NUMBER** pattern and 55.3.244.1 can be matched by the **IP** pattern. The syntax is how you match and ultimately references a regular expression. Refer to the following URL to see the patterns that come standard with the grok plug-in. Review the grok-patterns first because they are the most primitive subset of patterns. Then look at the **firewall** patterns to see how complex and specific they can get.

`https://grokdebug.herokuapp.com/patterns#`



The **SEMANTIC** is the identifier (or variable) that you give to the piece of text being matched. For example, 3.44 could be the duration of an event. So you could call it **duration**. Further, a string 55.3.244.1 might identify the *client* making a request. The two grok patterns for these data items would then be as follows:

```
%{NUMBER:duration} %{IP:client}
```

To full parse the message, use this pattern:

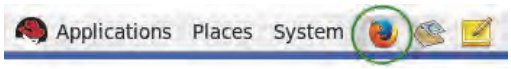
```
55.3.244.1 GET /index.html 3.44
```

You could use the following set of grok patterns (IP and URIPATHPARAM are predefined patterns.)

```
%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:duration}
```

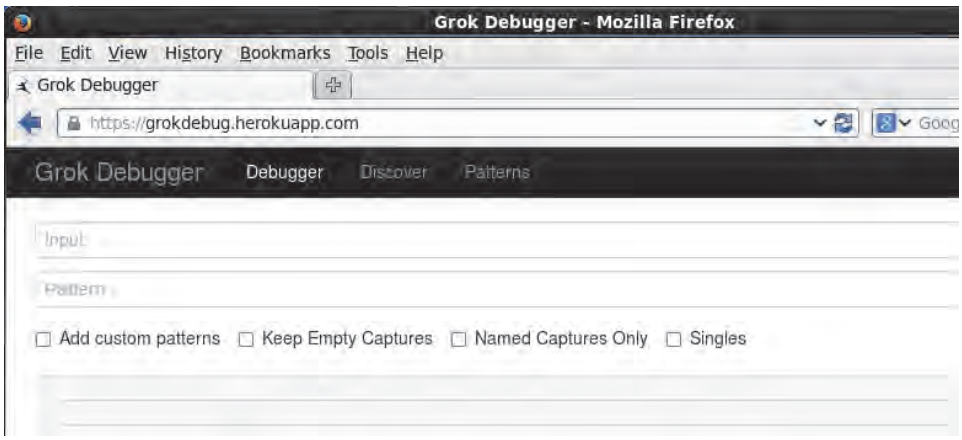
Complete the following exercise.

1. Create a grok statement.
   a. Open the Firefox browser inside the Predictive Insights VM. Click the Firefox icon on the upper menu bar.



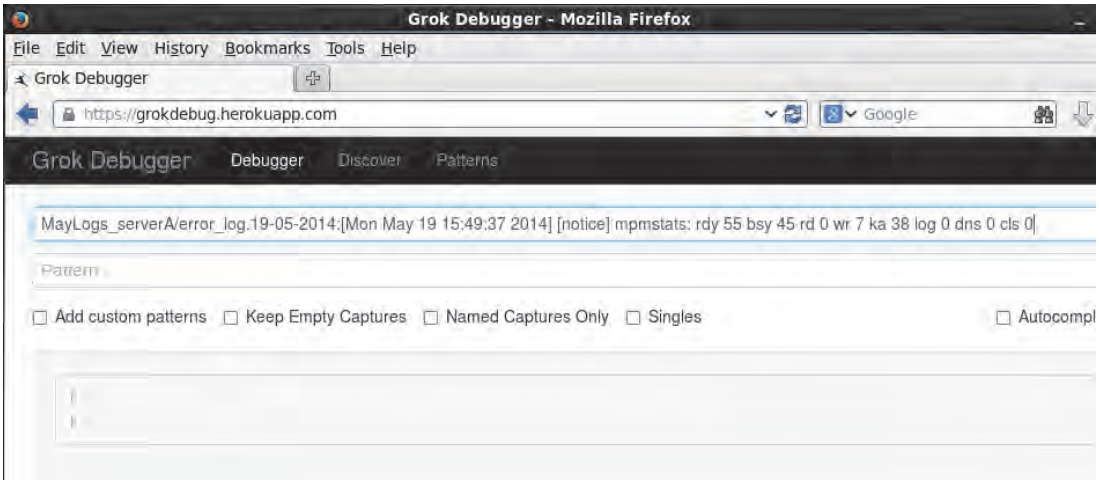
Important: You might not be to access the Internet with the VM that hosts Predictive Insights. If that is the case, you open a browser on your local desktop.

   b. When the browser opens, go to the following URL:
   https://grokdebug.herokuapp.com



c. To start the process of creating the grok statement, type or cut and paste an example message into the **Input** field. Enter this log statement into the **Input** field.

```
MayLogs_serverA/error_log.19-05-2014:[Mon May 19 15:49:37 2014] [notice]
mpmstats: rdy 55 bsy 45 rd 0 wr 7 ka 38 log 0 dns 0 cls 0
```



d. Build the grok pattern. Remember that the message must be structured. Extract the following items that are in red.

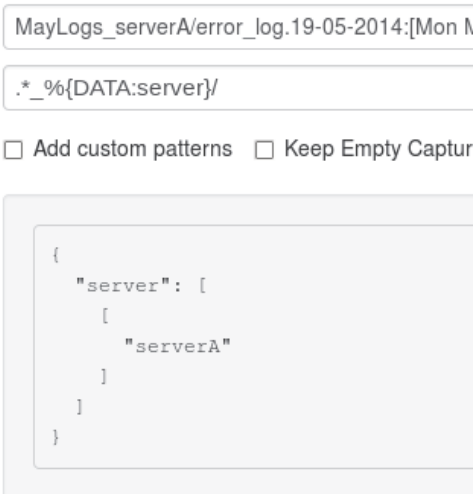
```
MayLogs_serverA/error_log.19-05-2014:[Mon May 19 15:49:37 2014] [notice]
mpmstats: rdy 55 bsy 45 rd 0 wr 7 ka 38 log 0 dns 0 cls 0
```

- Host name: serverA
- Date: 19-05-2014 (note European data format)
- Time: 15:49:37
- Performance metrics: rdy 55 bsy 45 rd 0 wr 7 ka 38 log 0 dns 0 cls 0

e. Enter the following text into the **Pattern** field to extract the host name. It compares any character as many times as necessary until it reaches the underscore. From there, you use

a grok predefined pattern for data. Anything between the underscore and forward slash is captured and associated to the field name server.

```
.*_%{DATA:server}/
```



f. You now must ignore all the characters until you reach the period. Then you must capture the message date. Add the following code to the pattern. This expression compares all characters until it finds a period. Because the period is a special character in regular expressions, you escape it with a backslash. You use another predefined search pattern called DATE, which extracts the date from the message. DATE is made up of a number other patterns that can extract the day, month, and year from the fields.

```
.*_%{DATA:server}/.*\.%{DATE:date}
```

g. To capture the time stamp, you skip over the colon, the square bracket (the square bracket is also a special character and must be escaped), and the day, month, and date and use the

TIME pattern to extract the server time. Here you use the predefined pattern WORD and whitespace to determine the three fields that you want to ignore. If you forget to use the whitespace or have more than one whitespace between patterns, you get a warning of No Matches. Scroll down the screen to see how the time stamp is further broken down by hour, minute, and seconds.

```
.*_%{DATA:server}/.*\.%{DATE:date}:\[%{WORD} %{WORD} %{WORD} %{TIME:time}
```

h. To keep the pattern simpler, use the following code to extract all the metrics from this message in one field for post processing in a subsequent call in the conf file. Add the following to the grok pattern. Note that the **Named Capture Only** has been selected on the web page to reduce the amount of output shown.

```
.*_%{DATA:server}/.*\.%{DATE:date}:\[%{WORD} %{WORD} %{WORD}
%{TIME:time}.*mpmstats: %{GREEDYDATA:metrics}
```

2. Place this pattern in a grok statement in your filter.
   a. Add the following code to the filter section in the **mixed-data.conf** file. You also add an additional field that defines the time stamp. This time stamp is a combination of both the **date** and **time** fields that are parsed from the message.

```
filter {
   if [message] =~ /.*mpmstats: rdy.*/  {
      mutate {
      add_tag => "USEFUL"
      }
   }
   else {
      drop {}
   }
   grok {
       match => ["message",".*_%{DATA:server}/.*\.%{DATE:date}:\[%{WORD}
%{WORD} %{WORD} %{TIME:time}.*mpmstats: %{GREEDYDATA:metrics}"]
      add_field => {"timestamp" => "%{date} %{time}"}
   }
}
```

3. You must now parse out the data in the **metrics** field you created in the previous step using an additional grok statement. You could do this step in one grok statement, but it would be long. The metrics field should have data with the following format.

```
rdy 55 bsy 45 rd 0 wr 7 ka 38 log 0 dns 0 cls 0
```

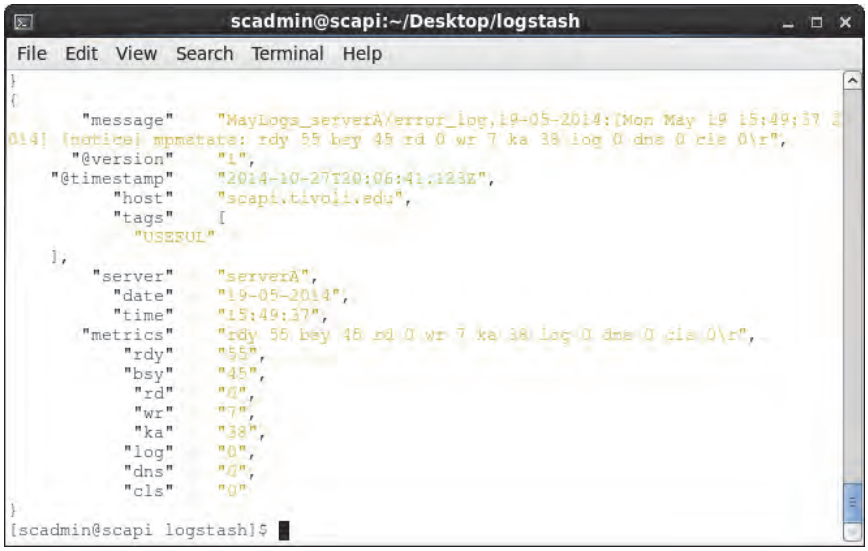
   a. Create another grok statement with the following format. Note how the pattern uses %{WORD} %{DATA:value-name} to structure the data.

```
filter {
   if [message] =~ /.*mpmstats: rdy.*/  {
      mutate {
      add_tag => "USEFUL"
      }
   }
   else {
      drop {}
   }
   grok {
       match => ["message",".*_%{DATA:server}/.*\.%{DATE:date}:\[%{WORD}
%{WORD} %{WORD} %{TIME:time}.*mpmstats: %{GREEDYDATA:metrics}"]
      add_field => {"timestamp" => "%{date} %{time}"}
   }
```

```
    grok {
        match => ["metrics","%{WORD} %{INT:rdy} %{WORD} %{INT:bsy} %{WORD}
%{INT:rd} %{WORD} %{INT:wr} %{WORD} %{INT:ka} %{WORD} %{INT:log} %{WORD}
%{INT:dns} %{WORD} %{INT:cls}"]
    }
}
```

b. Save the file in Emacs.

c. Test your new grok commands by rerunning logstash.



Exercise 8. Publishing data with scacsv

In this exercise, you use a custom plug-in that was developed specifically for Predictive Insights. When Predictive Insights reads from CSV files, the file must meet two formatting requirements.

- The first line in the file must have names that describe each column in the CSV file. These names must be the same between all files that are capturing the metric data.
- The file name must include a start time and end time that represents the timespan the data in the file covers. The time stamps must allow parsing by Java SimpleDataFormat rules, and the format must be consistent between file names.

As was shown earlier, the standard **csv** output plug-in does not support either of these requirements. To help, an IBM developer created a logstash plug-in called **scacsv**. This plug-in is currently being hosted on GitHub. You can access them at the following URL. Scrolling down on that webpage provides documentation on how to use this plug-in.

`https://github.com/IBM-ITOAdev/logstash-output-scacsv`

The plug-in was preloaded by the lab author and is hosted on the virtual machine in the following location:

```
/opt/scaLogstash/logstash/outputs/scacsv.rb
```

There are some key features that this plug-in uses to ensure it creates a correctly formatted file in a timely manner. These fields are the ones that you need to be most concerned with:

- **fields**: The name of the fields you want to include in the CSV file.
- **path**: The location and name of a temporary file that stores the CSV data.
- **group**: The final name give to the file that is created when file is closed and dated.
- file_interval_width: Setting this enables files to be closed on specified boundaries. This option is useful to break incoming stream up on PI preferred boundaries. Allowed values are MINUTE, HOUR, DAY. If set to HOUR, then all incoming data for a particular hour would be put in a file. When new data in the next hour arrives, the previous file is closed and a new one opened.
- **time_field**: The name of the field that defines the time stamp.
- **time_field_format**: The Java SimpleDateFormat of the **timestamp** field.
- **timestamp_output_format**: The Java SimpleDateFormat of the time stamps used in file name.

To ensure that logstash uses this plug-in, you define a **pluginpath** option when starting the server.

1. Use the **scascv** plug-in to create an output file in the correct Predictive Insights format.
   a. Add the following lines to the output section of the **mixed-data.conf** file.

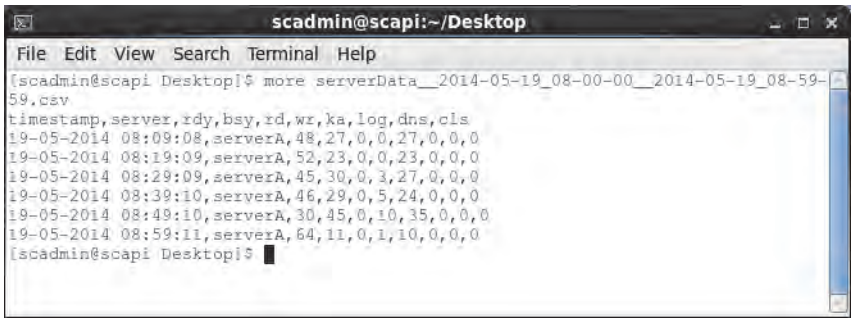
```
output {
   stdout {
      codec => rubydebug
   }
   scacsv {
      fields =>
["timestamp","server","rdy","bsy","rd","wr","ka","log","dns","cls"]
      path => "/home/scadmin/Desktop/server.csv"
      group => "serverData"
      file_interval_width => "HOUR"
      time_field => "timestamp"
      time_field_format => "dd-MM-yyyy HH:mm:ss"
      timestamp_output_format => "yyyy-MM-dd_HH-mm-ss"
   }
}
```

   b. Save the file and test the output commands. You must include the **pluginpath** option in your command:

```
head -100 /opt/scapi-training-data/logstash-examples/example-mixed-data.log
| /opt/logstash-2.1.0/bin/logstash -f mixed-data.conf -w 1 --pluginpath
/opt/scaLogstash
```

Note: The **-w 1** option was added to the logstash command to ensure only one log processing server was started. This addition avoids problems with multithreading the data that is being piped into the logstash command.

c. Review one of the new **serverData__<start time>__<end time>.csv** file that is on the **~/Desktop**. Note the naming format of the file and the format of the data within the file.



d. After you are done reviewing the output data, you can clean up the desktop with the command:

```
rm ~/Desktop/server*
```

Merging files

In this set of exercises, the necessary data for Predictive Insights is captured in a series of report files that were generated by an application. Each report file represents one host in the solution. These files must be merged together into one CSV file for Predictive Insights to use.

One of the challenges in this effort is that you do not know how many files to merge and the fully qualified domain name (FQDN) of the host is in the header to the file and not in the file name itself. Your challenge is that logstash cannot store a value from a previous message and use it for a subsequent one. Each message moving through the logstash conf file is essentially treated individually. The conf file either has hash files included with name-value pairs, or it relies on external mapping files to address these cache problems.

In this exercise, you create two conf files, one to create an external hash table by extracting the fully qualified host name from each report file and a second conf file that extracts the data from each report file and associates it to the FQDN.

You also learn about the limitations of logstash as a relatively new tool set. You must postprocess your output file so that the time stamps are correctly ordered. Currently, logstash does not have a plug-in to support this kind of sorting.

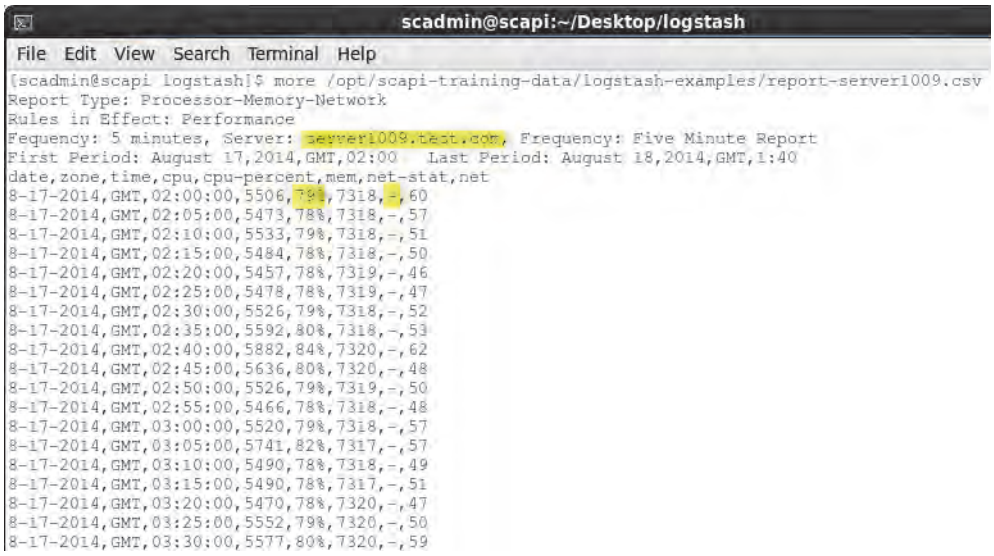
Exercise 9. Reviewing the report files

The report files are individual files that produce metric data for one server. For Predictive Insights to process these files you either have to create individual data sources for each file or merge the data into a single CSV file.

1. View the format of the report files:
   a. View the log file **/opt/scapi-training-data/logstash-examples/report-server1009.csv**, taking special note of the FQDN in the header.

      ```
      more /opt/scapi-training-data/logstash-examples/report-server1009.csv
      ```

Note that the percent sign is used with the **cpu-percent** value and that there is a dash for the **net-stat** value.

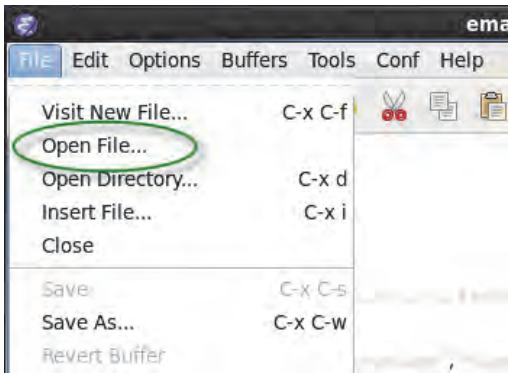


Exercise 10. Extracting FQDN from report files

1. Create a hash file where you store in a cache the FQDN names discovered in each file and associate them to their file names.
   a. Create a conf file called **extract-hosts.conf**:

      ```
      touch extract-hosts.conf
      ```

   b. Open **extract-hosts.conf** in Emacs.



   c. Instead of using the **stdin** plug-in for entering data, in this example, you have logstash read a set of files. Use the following **file** plug-in to read each of the report files.

      ```
      input {
         file {
            path => "/opt/scapi-training-data/logstash-examples/report*.csv"
      ```

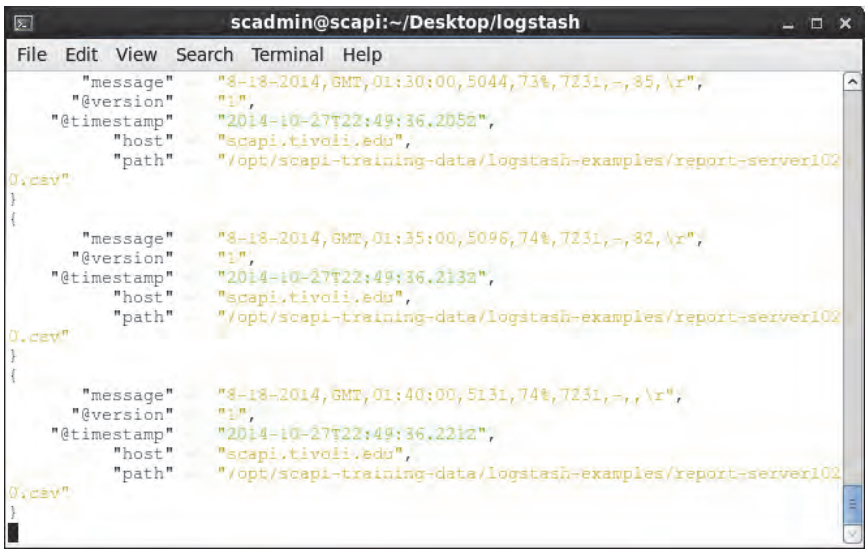
```
        start_position => "beginning"
    }
}

filter {
}

output {
    stdout {
        codec => rubydebug {}
    }
}
```

The option of **start_position** for the **file** plug-in tells logstash to read from the beginning of the file. You can find details of the **file** plug-in at the following location.

`https://www.elastic.co/guide/en/logstash/current/plugins-inputs-file.html`

d. Save this file and test it. You do not have to use the head command for this run because logstash is reading from a file. Because you are reading a file, logstash does not exit by itself. It continues reading these files and waiting for more input until you press Ctrl+c to exit.

`/opt/logstash-2.1.0/bin/logstash -f extract-hosts.conf`



Note that the path field that contains the file the message came from. This field is important and is used in an upcoming step.

You now search through each of these files to find the FQDN of the hosts.

2. As in the previous exercise, use an IF statement to find the specific line and then use grok to extract the data. Add the following to the filter section of your **extract-hosts.conf**

   a. Look at the header in the file and find the items to match on to find the message that has the FQDN. Here, you can match on **Frequency** and **Server** and use that to extract that specific message. If you do not find these attributes in the message, drop it.

   ```
   Report Type: Processor-Memory-Network
   Rules in Effect: Performance
   Frequency: 5 minutes, Server: server1009.test.com, Frequency: Five Minute
   Report
   First Period: August 17,2014,GMT,02:00   Last Period: August 18,2014,GMT,1:40
   date,zone,time,cpu,cpu-percent,mem,net-stat,net
   8-17-2014,GMT,02:00:00,5506,79%,7318,-,60
   8-17-2014,GMT,02:05:00,5473,78%,7318,-,57
   8-17-2014,GMT,02:10:00,5533,79%,7318,-,51
   8-17-2014,GMT,02:15:00,5484,78%,7318,-,50
   ```

   b. Add the following IF statement to your extract-hosts.conf file's filter section.

   ```
   filter {
      if [message]  =~ /^(?=.*Frequency)(?=.*Server)/ {
      }
      else {
         drop {}
      }
   }
   ```

   c. Create a grok message to extract the FQDN from the message. Add the following code to your filter section to find the message and parse it for the FQDN.

   ```
   filter {
      if [message]  =~ /^(?=.*Frequency)(?=.*Server)/ {
         grok {
         match => ["message","%{DATA} Server: %{DATA:FQDNServer},"]
         }
      }
      else {
         drop {}
      }
   }
   ```

   d. Save the **extract-hosts.conf** file and test it. You must remove the logstash file reader location files. If you do not remove these files, the logstash file reader resumes reading the

files from where it stopped in its previous execution. Press Ctrl+c to exit logstash after you read the output.

```
rm ~/.sincedb*
/opt/logstash-2.1.0/bin/logstash -f extract-hosts.conf
```


```
{
       "message"    "Fequency: 5 minutes, Server: server1020.test.com, Frequency
: Five Minute Report\r",
      "@version"    "1",
    "@timestamp"    "2014-10-28T16:09:41.932Z",
          "host"    "scapi.tivoli.edu",
          "path"    "/opt/scapi-training-data/logstash-examples/report-server102
0.csv",
    "FQDNServer"    "server1020.test.com"
}
```


3. For the hash file to work, you must equate the extracted FQDN with the report file name. Because the path is included as a field with the message, you can extract the file name from that field.
   a. Add another grok statement to extract the file name from the path field. Add the following statement to the filter section:

```
filter {
   if [message]  =~ /^(?=.*Frequency)(?=.*Server)/ {
      grok {
      match => ["message","%{DATA} Server: %{DATA:FQDNServer},"]
      }
      grok {
      match => ["path","%{GREEDYDATA}/%{GREEDYDATA:filename}.csv"]
      }
      }
   else
      drop {}
      }
}
```

   b. Save the **extract-hosts.conf** file and test it. Remember to remove the logstash file reader location files.

```
rm ~/.sincedb*
/opt/logstash-2.1.0/bin/logstash -f extract-hosts.conf
```


```
{
       "message"    "Fequency: 5 minutes, Server: server1020.test.com, Frequency
: Five Minute Report\r",
      "@version"    "1",
    "@timestamp"    "2014-10-28T16:24:13.298Z",
          "host"    "scapi.tivoli.edu",
          "path"    "/opt/scapi-training-data/logstash-examples/report-server102
0.csv",
    "FQDNServer"    "server1020.test.com",
      "filename"    "report-server1020"
}
```

4. Send the file name and FQDN to a mapping file with the **file** plug-in. You can find details of the file plug-in for the output section at this location.

   `https://www.elastic.co/guide/en/logstash/current/plugins-outputs-file.html`

   a. Add the following lines to the *output* section of the **extract-data.conf** file. Note the flush_interval is used to write to the output file every time there is a corresponding message, as opposed to waiting a certain period of time.

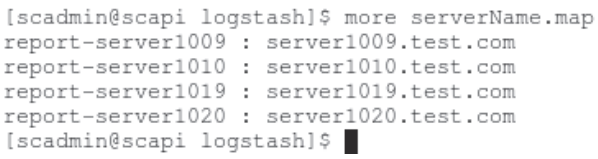
```
output {
   stdout {
      codec => rubydebug {}
   }
   file {
      path => "/home/scadmin/Desktop/serverName.map"
      message_format => "%{filename} : %{FQDNServer}"
      flush_interval => 0
   }
}
```

   b. Save and test the conf file:

```
rm ~/.sincedb*
/opt/logstash-2.1.0/bin/logstash -f extract-hosts.conf
```

   c. Review the **serverName.map** file that was created:

```
more /home/scadmin/Desktop/serverName.map
```



Exercise 11. Merging report files and adding FQDN

Now that you have a mapping file, you can create a conf file called **merge-reports.conf** that reads the reports, extracts the necessary performance metrics, and associates them to the correct FQDN. It also removes percent signs and dashes where they are not needed.

1. Create a new **merge-reports.conf** file.

   a. Issue the following command:

```
touch /home/scadmin/Desktop/logstash/merge-reports.conf
```

   b. Open the file in Emacs.

c. As with the **extract-hosts.conf** file, have **merge-reports.conf** file read all the files in a director that match **report-*.csv**. Add the following code to the *input* section of the conf file.

```
input {
   file {
      path => "/opt/scapi-training-data/logstash-examples/*.csv"
      start_position => "beginning"
   }
}

filter {

}

output {
   stdout {
      codec => rubydebug {}
   }
}
```

2. Find the necessary data in the file with an IF statement and label each of the fields.

   a. In the filter section, add an IF statement that finds the messages associated with performance data. Select messages that have **-2014** in them. Drop the others.

```
filter {
   if [message] =~ /.*-2014.*/ {

   }
   else {
      drop {}
   }
}
```

   b. Use the **csv** plug-in to name each field that is separated by a comma. Here is what the selected messages would look like:

```
8-17-2014,GMT,02:00:00,5506,79%,7318,-,60
8-17-2014,GMT,02:05:00,5473,78%,7318,-,57
8-17-2014,GMT,02:10:00,5533,79%,7318,-,51
8-17-2014,GMT,02:15:00,5484,78%,7318,-,50
```

   c. Enter the following text into the filter section. Note the addition of the time stamp, as in the previous exercise.

```
filter {
   if [message] =~ /.*-2014.*/ {
      csv {
```

```
        columns =>
["date","timezone","time","processor","process-percent","memory","net-stat",
"network"]
        add_field => {"timestamp" => "%{date} %{time}"}
        add_tag => "added fields to CSV data"
        }
    }
    else {
        drop {}
    }
}
```

3. Determine which FQDN to associate to the message, based on looking at the path:

Important: Determine the file name that the message was received from. Use the path field that is with the message and extract the file name. Add the following text to the *filter* section.

```
filter {
    if [message] =~ /.*-2014.*/ {
        csv {
        columns =>
["date","timezone","time","processor","process-percent","memory","net-stat",
"network"]
        add_field => {"timestamp" => "%{date} %{time}"}
        add_tag => "added fields to CSV data"
        }
        grok {
        match => ["path","%{GREEDYDATA}/%{GREEDYDATA:filename}.csv"]
        }
    }
    else {
        drop {}
        }
}
```

You can use the **translate** plug-in to work with the mapping file that you created in the earlier steps. You can find details on this plug-in at this location.

`https://www.elastic.co/guide/en/logstash/current/plugins-filters-translate.html`

4. Add the following code to your *filter* section. The **destination** argument means that a field called **FQDNServer** is created and the results of the matching are added to it.

```
filter {
    if [message] =~ /.*-2014.*/ {
```

```
        csv {
        columns =>
["date","timezone","time","processor","process-percent","memory","net-stat",
"network"]
        add_field => {"timestamp" => "%{date} %{time}"}
        add_tag => "added fields to CSV data"
        }
        grok {
        match => ["path","%{GREEDYDATA}/%{GREEDYDATA:filename}.csv"]
        }
        translate {
        dictionary_path => "/home/scadmin/Desktop/serverName.map"
        field => "filename"
        destination => "FQDNServer"
        add_tag => "figured out server name"
        }
    }
    else {
        drop {}
        }
}
```

5. Remove the percent sign and dash sign from the appropriate data columns.
   a. Use the **mutate** plug-in to alter the **process-percent** and **net-stat** fields that were created by the **csv** plug-in. Add the following code to the *filter* section.

```
filter {
    if [message] =~ /.*-2014.*/ {
        csv {
        columns =>
["date","timezone","time","processor","process-percent","memory","net-stat",
"network"]
        add_field => {"timestamp" => "%{date} %{time}"}
        add_tag => "added fields to CSV data"
        }
        grok {
        match => ["path","%{GREEDYDATA}/%{GREEDYDATA:filename}.csv"]
        }
        translate {
        dictionary_path => "/home/scadmin/Desktop/serverName.map"
        field => "filename"
        destination => "FQDNServer"
        add_tag => "figured out server name"
        }
        mutate {
        gsub => ["process-percent", "%",""]
```

```
        gsub => ["net-stat", "-",""]
        add_tag => "removed % and -"
        }
    }
    else {
        drop {}
        }
}
```

b. Save the m**erge-reports.conf** file and test it. There is at least 1 minute between starting and the messages showing up in standard output.

```
rm ~/.sincedb*
/opt/logstash-2.1.0/bin/logstash -f merge-reports.conf --pluginpath
/opt/scaLogstash/
```

Note: The **translate** plug-in is not included with the default logstash installation. It was downloaded by the author and placed in the /**opt/scaLogstash/logstash/filter** directory. The previous steps had you add the **translate** plug-in and hence, the use of the **pluginpath** option.

6. Publish the data to a file using the **scacsv** plug-in.
   a. Add the following text to the output section of the **report-merge.conf** file.

```
output {
    stdout {
        codec => rubydebug {}
    }
    scacsv {
```

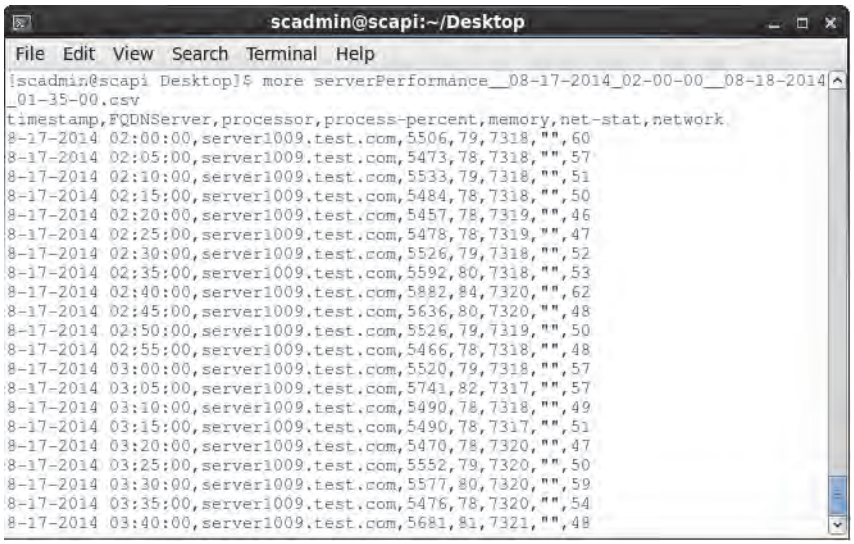
```
        fields =>
["timestamp","FQDNServer","processor","process-percent","memory","net-stat",
"network"]
        path => "/home/scadmin/Desktop/servertemp.csv"
        group => "serverPerformance"
        time_field => "timestamp"
        time_field_format => "MM-dd-yyyy HH:mm:ss"
        timestamp_output_format => "yyyy-MM-dd_HH-mm-ss"
    }
}
```

b. Save the file and test it. The newest conf file uses **scacsv**, which requires the plug-in path to be defined. Because there is a timeout associated with scacsv, it takes 1 minute for the final file to be displayed on the desktop. This conf file takes longer to run than the others that you have worked with; so have patience when running it. Use Ctrl+c to exit.

```
rm ~/.sincedb*
/opt/logstash-2.1.0/bin/logstash -f merge-reports.conf --pluginpath
/opt/scaLogstash/
```

c. Review this output file:
serverPerformance__2014-08-17_02-05-00__2014-08-18_01-35-00.csv

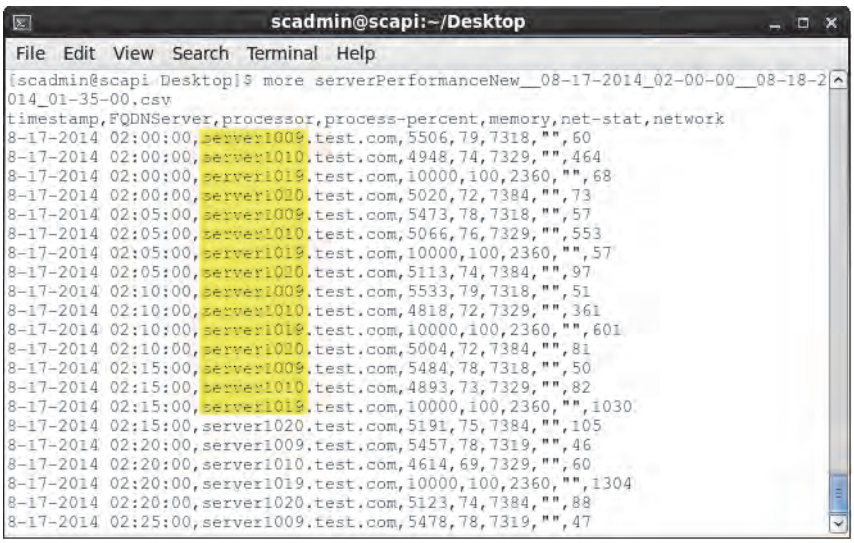
Notice that each file was read and published sequentially. This means that the data is not in chronological order. Logstash currently does not have a usable sorting function to fix the chronology of the file. However, Linux does.



7. Use the Linux **sort** command to fix the file.
   a. Use the following Linux command to sort the file by the time stamps. You do not want to sort the first line of the file because it is a header file. However, you do want to sort all other lines.

      ```
      cd /home/scadmin/Desktop
      (head -n 1 serverPerformance__2014-08-17_02-05-00__2014-08-18_01-35-00.csv && tail -n +2 serverPerformance__2014-08-17_02-05-00__2014-08-18_01-35-00.csv | sort) > servertempNew.csv
      ```

   b. Review the new file that was created to see the results of this sort.



Using logstash to connect to databases

In this set of examples, you use logstash to connect to a database and extract data from it. Your first reaction might be to ask the question “Doesn’t the Predictive Insights mediation client already connect to databases for extraction?” You are correct in asking this question and, indeed, the mediation client is designed to make JDBC calls to databases for extracting data. However, the mediation client allows only simple extractions from a single data table. If the data table uses foreign keys for resource names and other attributes, the mediation client does not allow you to join tables to resolve these keys.

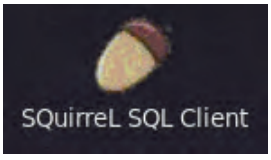
A solution to this problem can be the creation of a view in the database to resolve these keys and then for the mediation client to make calls to this view. However, that may not be feasible in some instances due to customer issues.

If the creation of a database view is not feasible, then you need to create an SQL statement that does the join for you. Because the mediation client currently does not support this feature, logstash becomes a viable option.

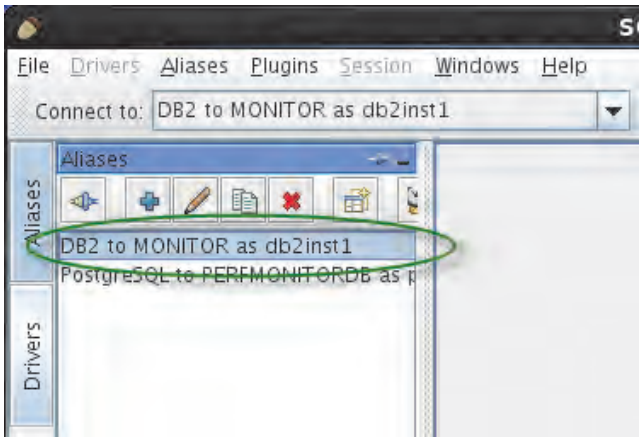
Exercise 12. Reviewing data tables

The following exercise has you review the tables that are part of monitoring database that is based on the Solarwinds Orion schema. Here network resources and their network interfaces are stored in separate tables. You use the JDBC database viewer SQuirreL to look at the tables that are hosted in the DB2 database.

1. As the **db2inst1** user, review the data in the MONITOR database that is in a DB2 database. You use SQuirreL as a client tool to look at the data tables.
   a. Connect to the MONITOR tables in DB2 with the SQuirreL SQL client utility. On the Linux desktop, double-click the acorn icon to start SQuirreL.SQL client.



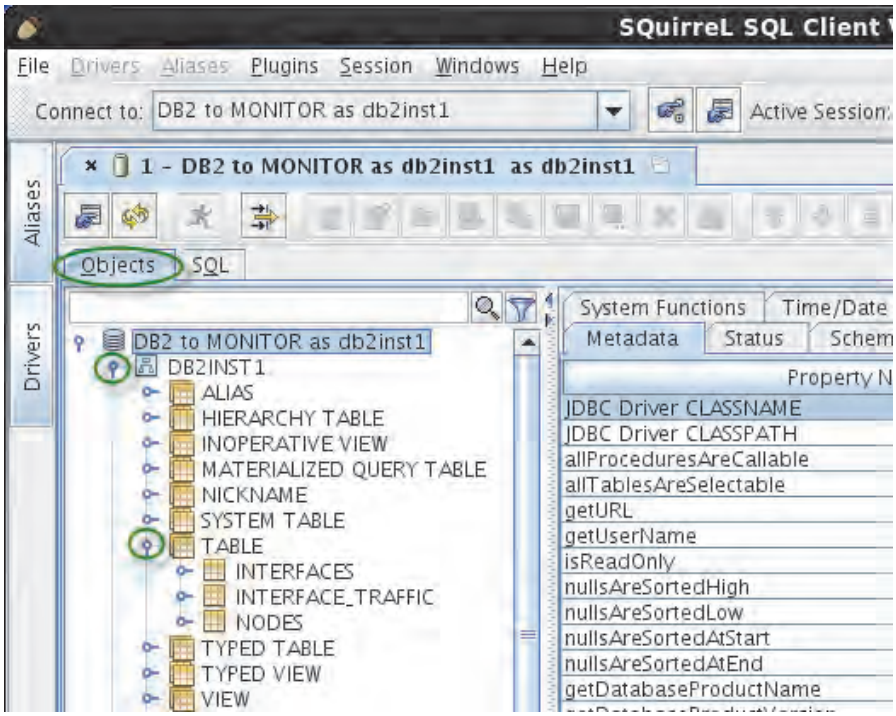
b. In the left window, double-click the **DB2 to MONITOR as db2inst1** alias.



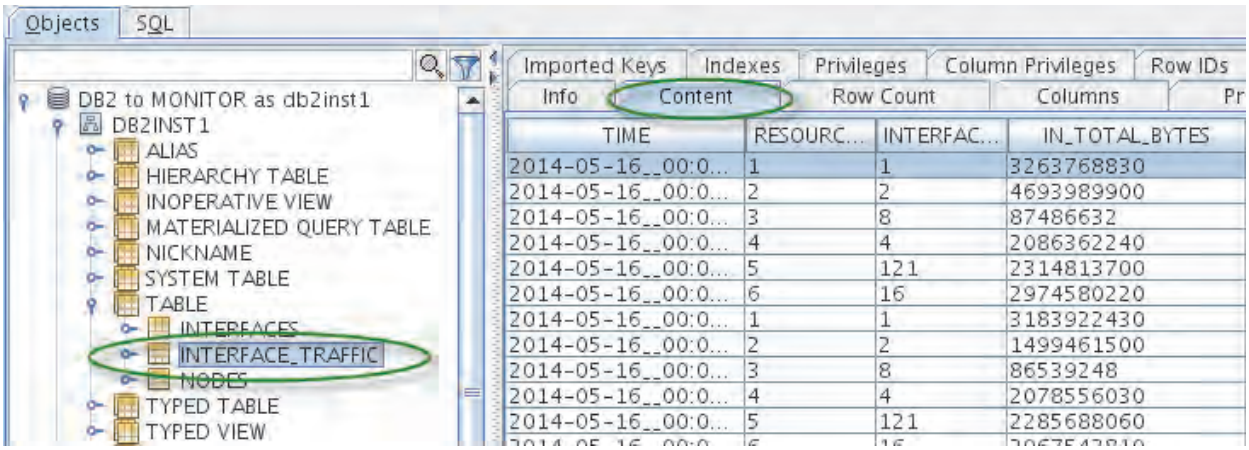
You are automatically connected to the database.

Note: Occasionally, you might receive a time-out error with your first connection to the database. Try connecting again. The problem usually resolves itself.

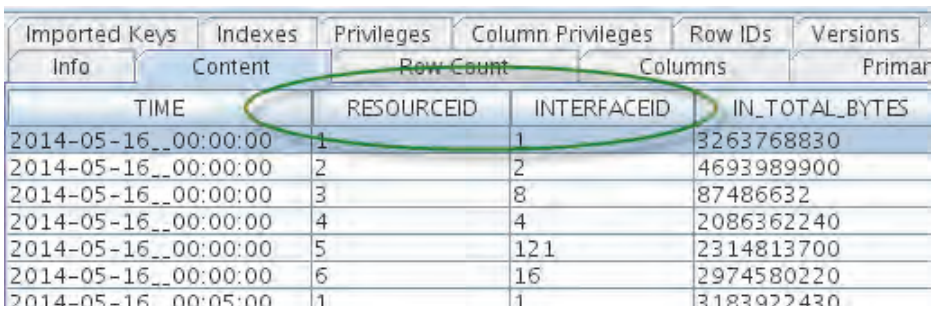
c. On the **Objects** tab, expand **DB2INST1**, and expand **TABLE.**



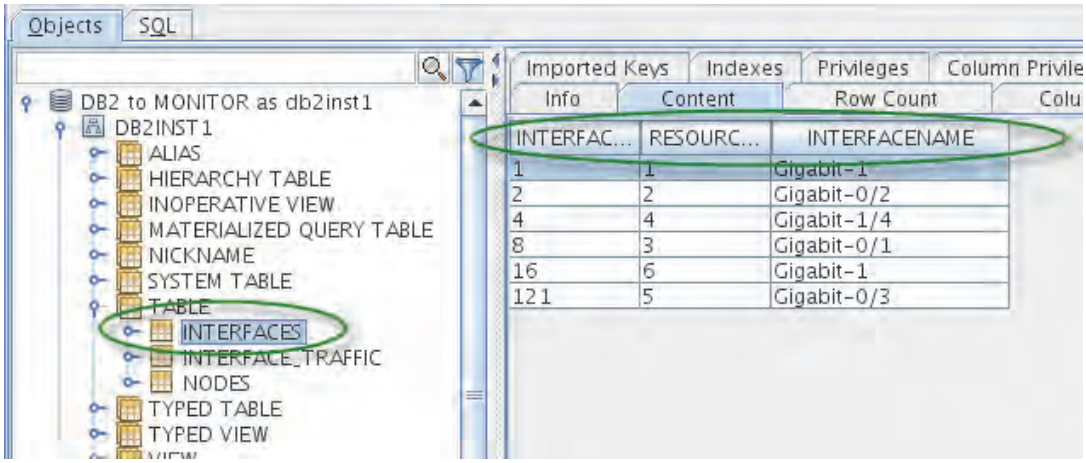
d. Display the INTERFACE_TRAFFIC table and note the foreign keys. Select the **INTERFACE_TRAFFIC** table, and click the **Content** tab.



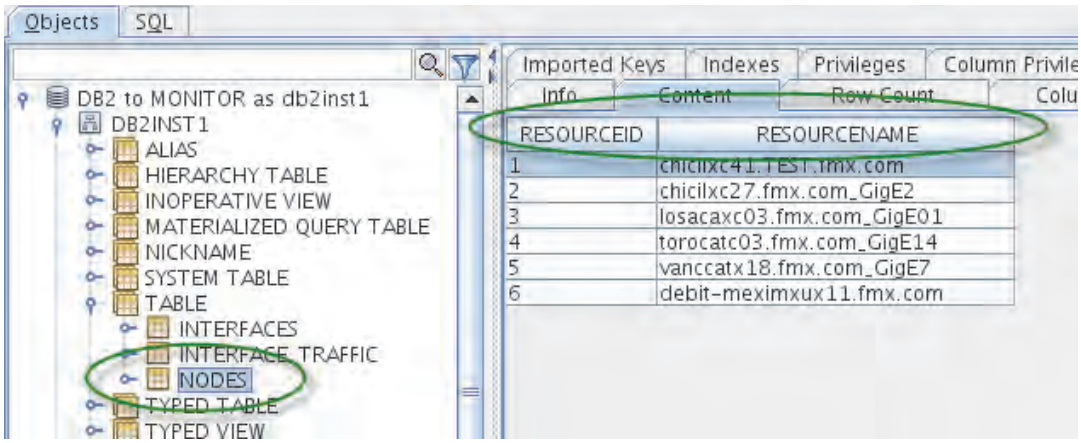
Note that the **RESOURCEID** and **INTERFACEID** uses foreign keys to reference the host name and interface of a device.



e. Select the **INTERFACES** tables and note the data that is on the **Content** tab. See how the **INTERFACEID** resolves to an **INTERFACENAME**. Note the **RESOURCEID** that is associated with this table.



f. Select the **NODES** table and note the data that is on the **Content** tab. You can associate **RESOURCEID** with a **RESOURCENAME**.



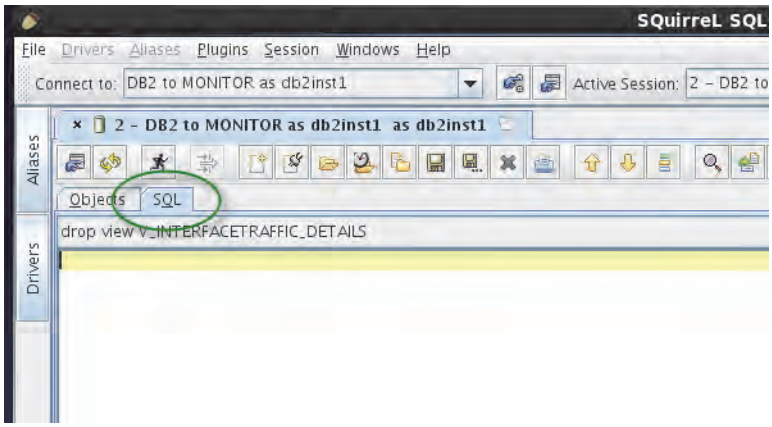
g. Keep the SQuirreL SQL client window open.

Exercise 13 Testing a database join

In this exercise, you build and test a database join command using the SQuirreL database client. This test ensures that your SQL command works before attempting to use it in the logstash configuration files. Working with SQL using the SQL client is much more convenient that attempting to test SQL inside of a logstash file.

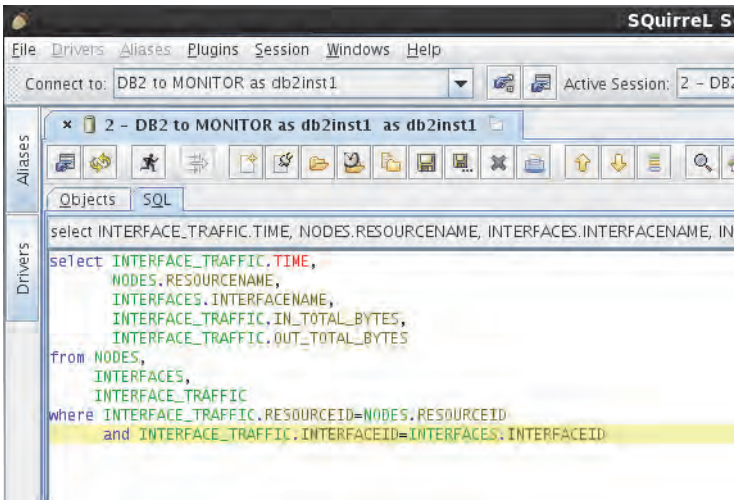
Complete the following steps.

1. Click the SQL tab inside the SQuirreL client.

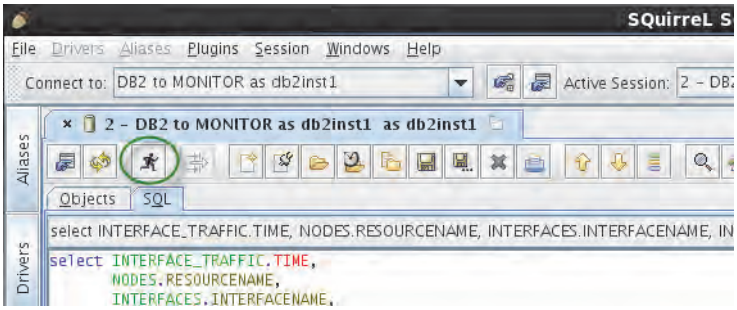


2. Enter the following SQL in the editing field and click run.
   a. Copy and paste the following SQL into the SQuirreL SQL editor:

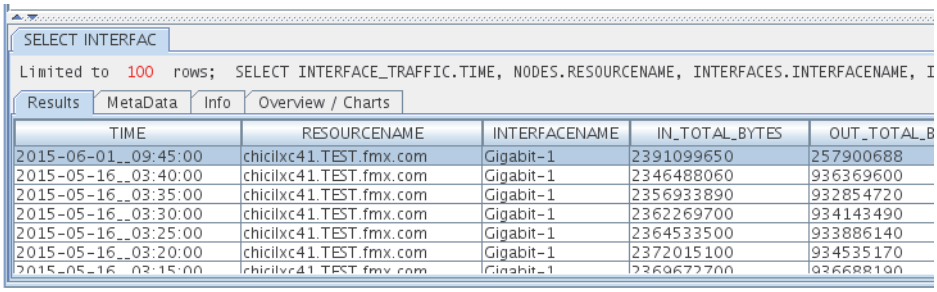
```
select INTERFACE_TRAFFIC.TIME, NODES.RESOURCENAME, INTERFACES.INTERFACENAME,
INTERFACE_TRAFFIC.IN_TOTAL_BYTES from NODES, INTERFACES, INTERFACE_TRAFFIC
where INTERFACE_TRAFFIC.RESOURCEID=NODES.RESOURCEID and
INTERFACE_TRAFFIC.INTERFACEID=INTERFACES.INTERFACEID
```



b. Run the SQL by clicking the run icon.



c. Review the results. Are the foreign keys resolved?



TIME	RESOURCENAME	INTERFACENAME	IN_TOTAL_BYTES	OUT_TOTAL_B
2015-06-01_09:45:00	chicilxc41.TEST.fmx.com	Gigabit-1	2391099650	257900688
2015-05-16_03:40:00	chicilxc41.TEST.fmx.com	Gigabit-1	2346488060	936369600
2015-05-16_03:35:00	chicilxc41.TEST.fmx.com	Gigabit-1	2356933890	932854720
2015-05-16_03:30:00	chicilxc41.TEST.fmx.com	Gigabit-1	2362269700	934143490
2015-05-16_03:25:00	chicilxc41.TEST.fmx.com	Gigabit-1	2364533500	933886140
2015-05-16_03:20:00	chicilxc41.TEST.fmx.com	Gigabit-1	2372015100	934535170
2015-05-16_03:15:00	chicilxc41.TEST.fmx.com	Gigabit-1	2369672700	936688190

Exercise 14. Building a logstash conf file

Now that you have a tested query, you can begin building a logstash configuration file that uses the custom plug-in **genjdbc** that was developed by an IBM engineer to make the appropriate call to the data base and collect data as logstash messages that are processed in the same manner as the messages that you worked with earlier.

Before stepping through the exercise, you should review and understand the **genjdbc** plug-in. You can find these details at the following URL. Scrolling down on that webpage provides documentation on how to use this plug-in.

https://github.com/IBM-ITOAdev/logstash-input-genjdbc

As was shown in the installation exercises for those who had Internet access, you can install this plug-in using the **plugin** command that is included with logstash. The plug-in is also hosted on the virtual machine in the following location:

/opt/scaLogstash/logstash/inputs/genjdbc.rb

There are some key features that this plug-in uses:

- **jdbcHost**: A string that defines the host name or IP address of the database host.
- **jdbcPort**: A string that defines the port number that the database is listening on for JDBC connections.
- **jdbcDBName**: A string that defines the database name to which you want to connect.

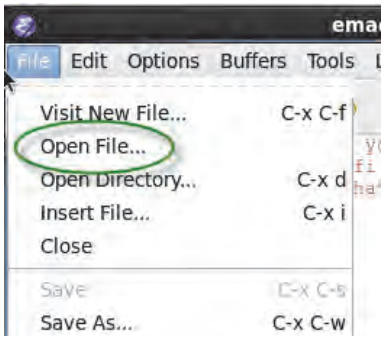
- **jdbcTargetDB**: A string that defines the database vendor. The currently supported databases are **postgresql**, **oracle**, **db2**, **mysql**, **derby**, and **mssql**. Review the **genjdbc.rb** file to see how to extend this list if necessary.
- **jdbcDriverPath**: A string that defines the location of the JDBC jar file that has the appropriate drivers to make a JDBC connection. These jar files are not included with genjdbc. You must download them from the database vendor.
- **jdbcUser**: A string that defines the user name you want to connect to the database as.
- **jdbcPassword**: A string that defines the password that you want to use to connect to the database as the jdbcUser.
- **jdbcSQLQuery**: A string that defines the SQL query that you are sending to the database.
- **jdbcURL**: An optional string that overrides the URL that genjdbc uses to connect to the database.
- **jdbcTimeField**: An optional string that defines the name of the table column that contains time stamp information
- **jdbcPollInterval**: An optional number of seconds to wait between query loops. A query loop runs the query, waits for a response, and processes the response by emitting events. After these steps are completed, the plug-in waits the specified amount of time before starting again and invoking the query.
- **jdbcCollectionStartTime**: An optional string expressed as a time stamp (for example, '2015-01-01 00:00:00.000'). Because it is used in direct comparison operations against the specified **jdbcTimeField**, the precise format depends on the database and the nature of the time column referenced.

This plug-in makes periodic calls to the database by using the jdbcSQLQuery that you give it and adjusting the jdbcCollectionStartTime using the jdbcPollInterval. It is specifically designed to retrieve metric data and so it makes those periodic calls. The jdbcPollInterval should align with the Predictive Insights aggregation interval.

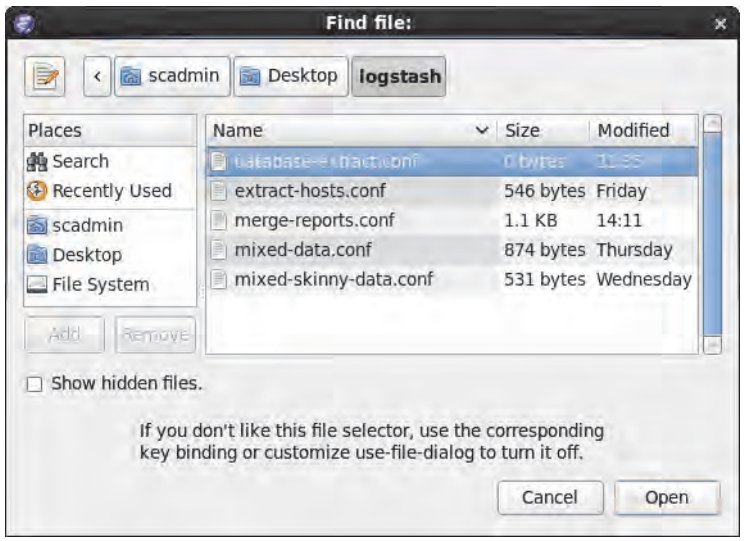
Complete the following steps.

1. In the **/home/scadmin/Desktop/logstash** directory, create the file **database-extract.conf**:
   a. Open a terminal session and change directory to **/home/scadmin/Desktop/logstash**:
      ```
      cd /home/scadmin/Desktop/logstash
      ```
   b. Create a new file with the **touch** command:
      ```
      touch database-extract.conf
      ```

2. Create a simple conf file so that you can test the genjdbc plug-in:
   a. Open the new file in your favorite editor. In the Emacs editor, by select **File** > **Open File** and navigating to **scadmin** > **Desktop** > **logstash.** Click **Open**.



   b. Select the newly created file.



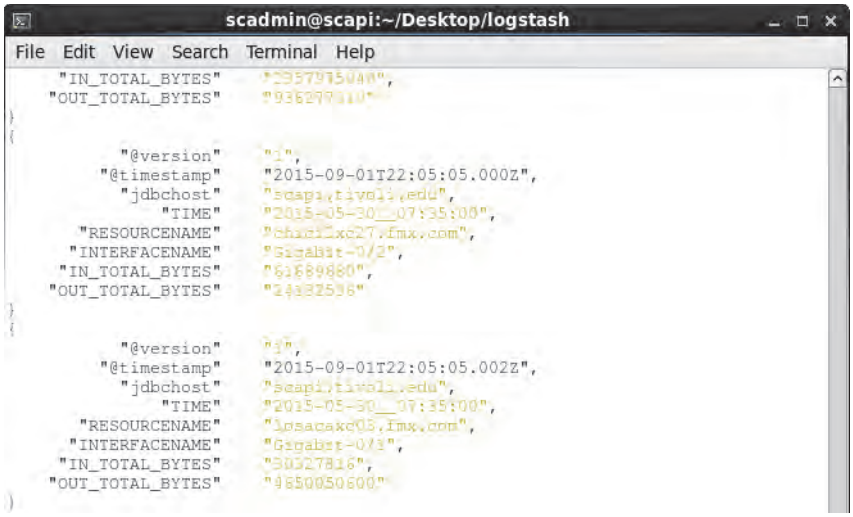
   c. In the blank file, add the following code to configure this configuration file

```
input {
   genjdbc {
    jdbcHost       => 'scapi.tivoli.edu'
    jdbcPort       => '50000'
    jdbcTargetDB   => 'db2'
    jdbcDBName     => 'MONITOR'
    jdbcUser       => 'db2inst1'
    jdbcPassword   => 'object00'
    jdbcDriverPath => '/opt/ibm/db2/V10.5/java/db2jcc4.jar'
    jdbcSQLQuery   => 'select INTERFACE_TRAFFIC.TIME, NODES.RESOURCENAME,
INTERFACES.INTERFACENAME, INTERFACE_TRAFFIC.IN_TOTAL_BYTES from NODES, INTERFACES,
INTERFACE_TRAFFIC where INTERFACE_TRAFFIC.RESOURCEID=NODES.RESOURCEID AND
INTERFACE_TRAFFIC.INTERFACEID=INTERFACES.INTERFACEID'
     jdbcTimeField  => 'INTERFACE_TRAFFIC.TIME'
     jdbcCollectionStartTime => '2015-10-23__00:00:00'
     }
}
```

```
filter {
}

output {
    stdout {
     codec => rubydebug {}
    }
}
```

d. Save and test this file with the following command

```
/opt/logstash-2.1.0/bin/logstash -f database-extract.conf --pluginpath
/opt/scaLogstash/
```



3. Export the database data to a CSV file.

   a. Add the following lines to the output section of your conf file.

```
input {
    genjdbc {
        jdbcHost       => 'scapi.tivoli.edu'
        jdbcPort       => '50000'
        jdbcTargetDB   => 'db2'
        jdbcDBName     => 'MONITOR'
        jdbcUser       => 'db2inst1'
        jdbcPassword   => 'object00'
        jdbcDriverPath => '/opt/ibm/db2/V10.5/java/db2jcc4.jar'
        jdbcSQLQuery   => 'select INTERFACE_TRAFFIC.TIME, NODES.RESOURCENAME,
INTERFACES.INTERFACENAME, INTERFACE_TRAFFIC.IN_TOTAL_BYTES, from NODES, INTERFACES,
INTERFACE_TRAFFIC where INTERFACE_TRAFFIC.RESOURCEID=NODES.RESOURCEID AND
INTERFACE_TRAFFIC.INTERFACEID=INTERFACES.INTERFACEID'
         jdbcTimeField  => 'INTERFACE_TRAFFIC.TIME'
         jdbcCollectionStartTime => '2015-10-23__00:00:00'
         }
}
```

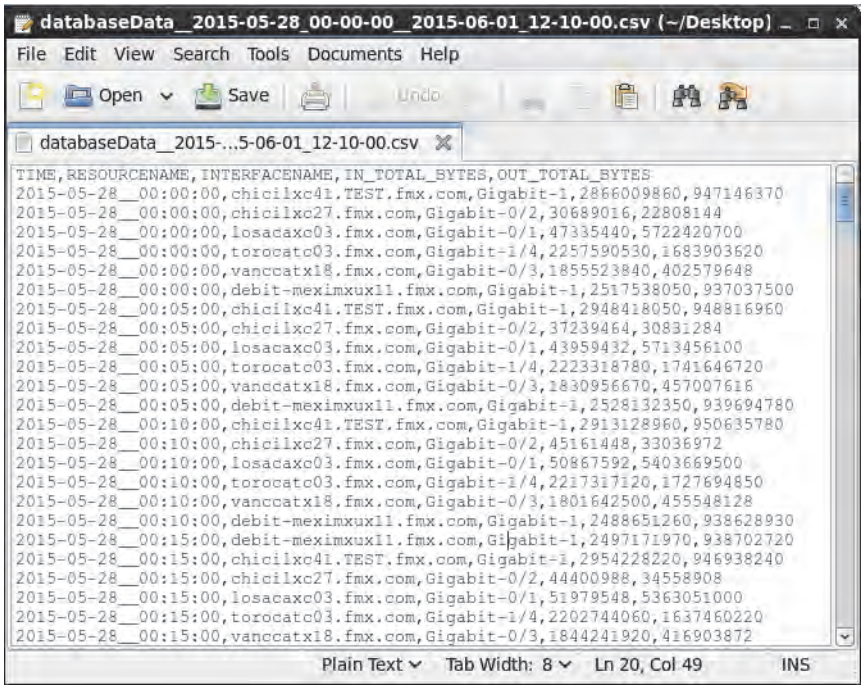
```
filter {
}

output {
   stdout {
      codec => rubydebug {}
   }
      scacsv {
      fields =>
["TIME","RESOURCENAME","INTERFACENAME","IN_TOTAL_BYTES","OUT_TOTAL_BYTES"]
      path => "/home/scadmin/Desktop/data.csv"
      group => "databaseData"
      file_interval_width => "HOUR"
      time_field => "TIME"
      time_field_format => "yyyy-MM-dd__HH:mm:ss"
      timestamp_output_format => "yyyy-MM-dd_HH-mm-ss"
   }
}
```

b. Save and test this file with the following command

   `/opt/logstash-2.1.0/bin/logstash -f database-extract.conf --pluginpath /opt/scaLogstash/`

c. After the output to the terminal stops, you can use Ctrl+c to stop logstash.

d. Review one of the **databaseData__<start-time>__<end-time>.csv** files on the Desktop.



Appendix A Installation of DB2

1. As root, create **/scapi-install-directory**.
2. Move installation file into that directory with SSH/SFTP.
3. Unpack the installation.
4. Ran the server_r/db2prereqcheck.
5. Add the following packages and libraries
   a. 32 bit versions of

      pam-1.1.1-17.el6
      libstdc++-4.4.7-4.el6
      libibcm-1.0.5-3.el6.i686
      libcxgb3-1.3.1-1.el6.i686
      libibverbs-1.1.7-1.el6
      libibverbs-devel-1.1.7-1.el6
      librdmacm-1.0.17-1.el6
      libibmad-1.3.9-1.el6
      libibumad-1.3.8-1.el6
      libmlx4-1.0.5-4.el6.1
      libmthca-1.0.6-3.el6

   b. 64 bit versions of

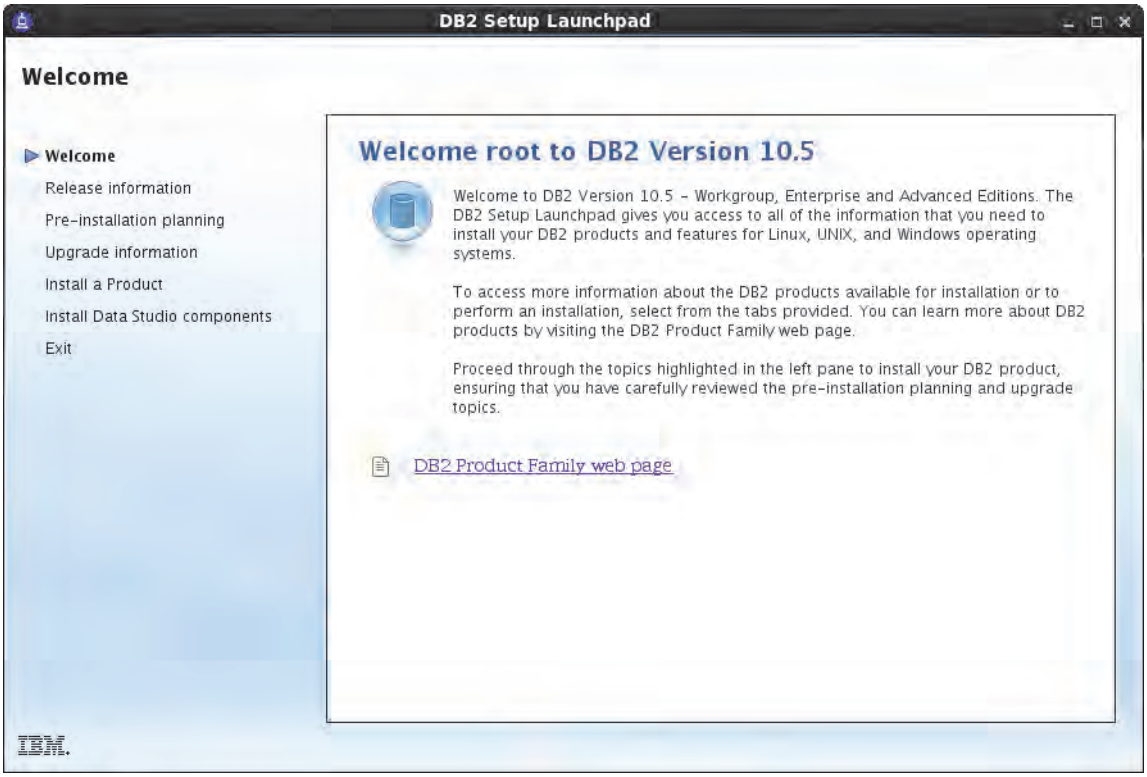
      libibcm-1.0.5-3.el6.x86_64
      dapl
      ibsim-0.5-7.el6
      ibutils-1.5.7-8.el6.x86_64
      libcxgb3-1.3.1-1.el6.x86_64
      libibverbs-devel-1.1.7-1.el6
      libibverbs-utils-1.1.7-1.el6
      libibmad-1.3.9-1.el6
      libibumad-1.3.8-1.el6
      libipathverbs-1.2-4.el6
      libmlx4-1.0.5-4.el6.1
      libmthca-1.0.6-3.el6
      libnes-1.1.3-1.el6
      rdma-3.10-3.el6 (noarch)
      sg3_utils-1.28-5.el6
      sg3_utils-libs-1.28-5.el6

6. Rerun the **server_r/db2prereqcheck command** until most of the checks passed.

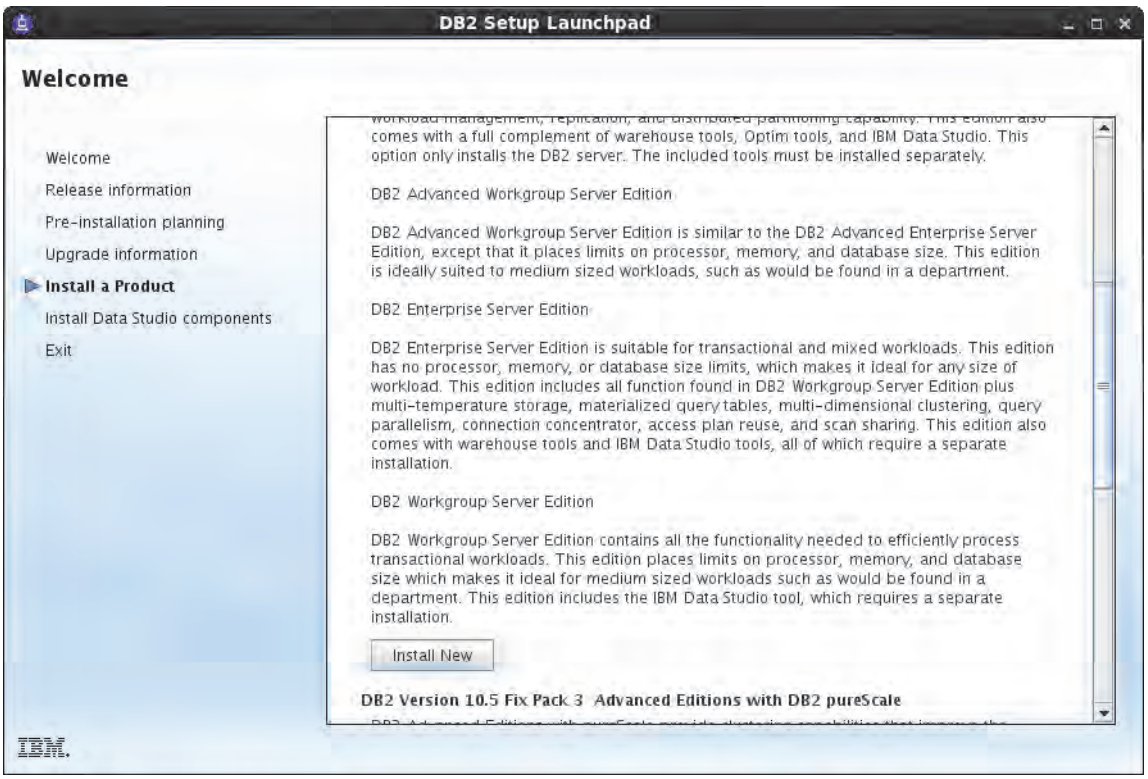
   Ignored the following errors:

   ```
   DBT3588W  The db2prereqcheck utility was unable to validate the configuration of
   the log_mtts_per_seg parameter on the following host machine:
   "scapi.tivoli.edu". Reason code: "1".
   DBT3566E  The db2prereqcheck utility detected that the service named "rdma" is
   not enabled on host "scapi.tivoli.edu".
   ```

7. Run db2setup.

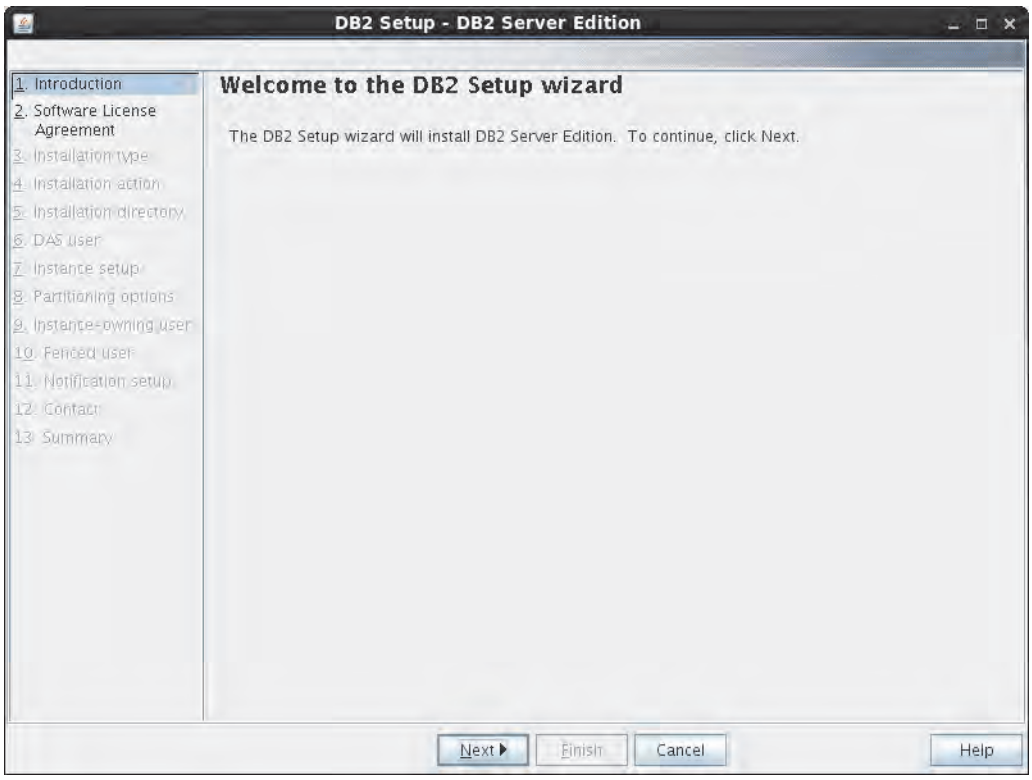


8. Install a new DB2 Workgroup Server Edition.

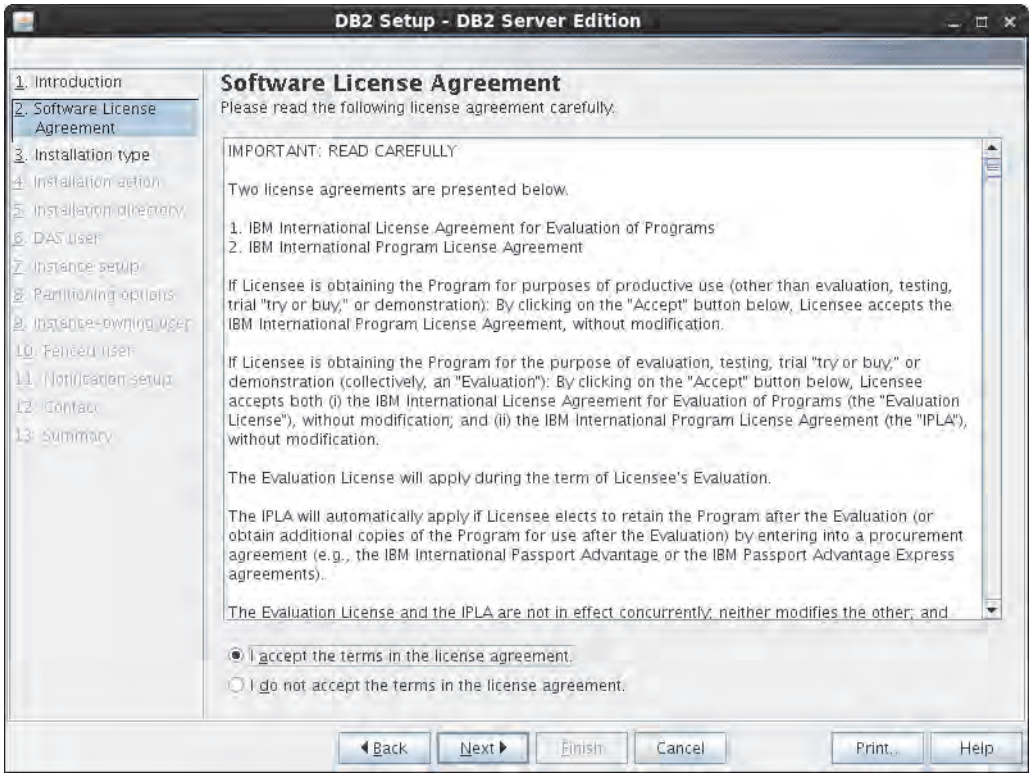


The installer opens.

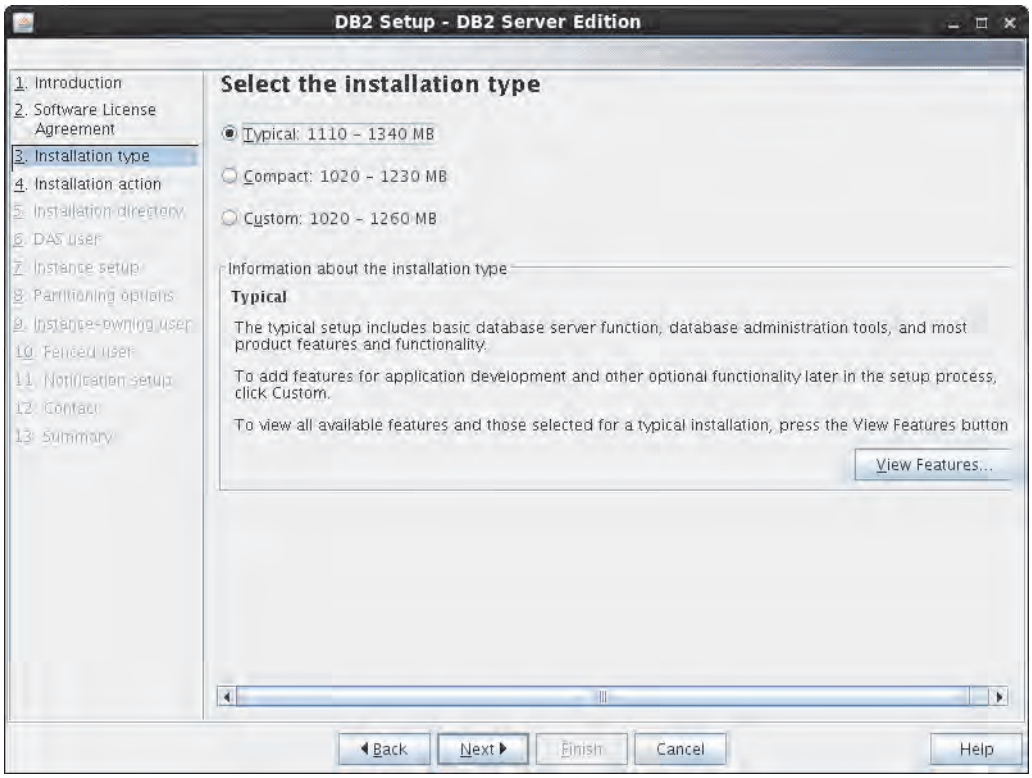
9. Click **Next**.



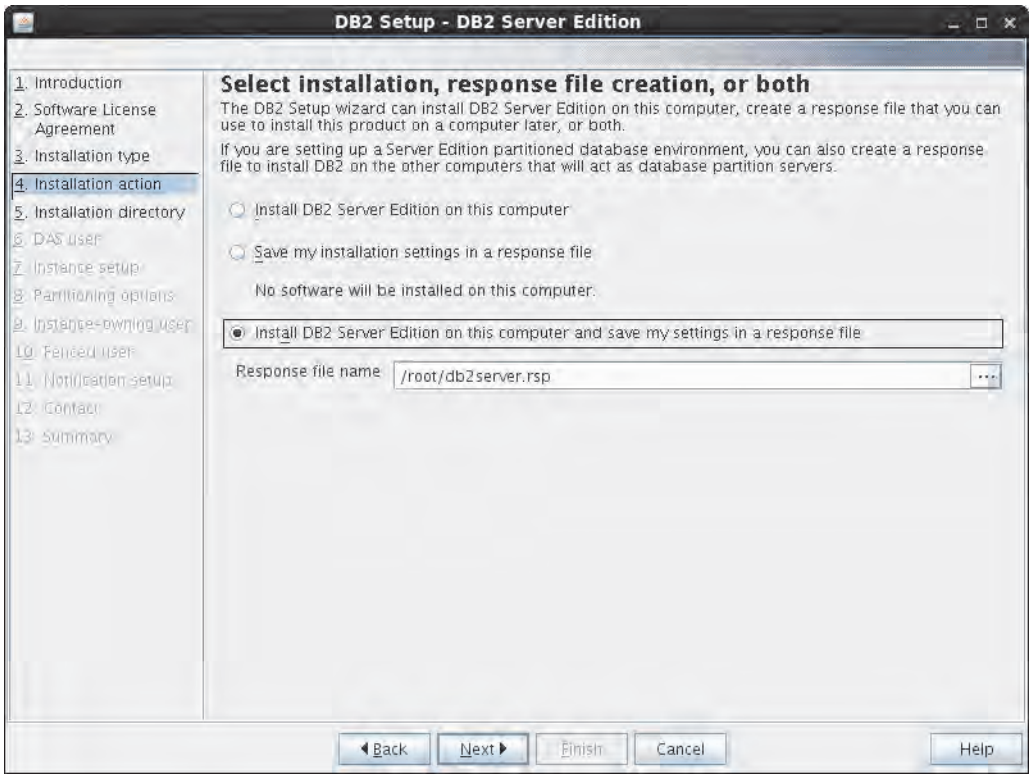
10. Accept the license agreement and click **Next**.



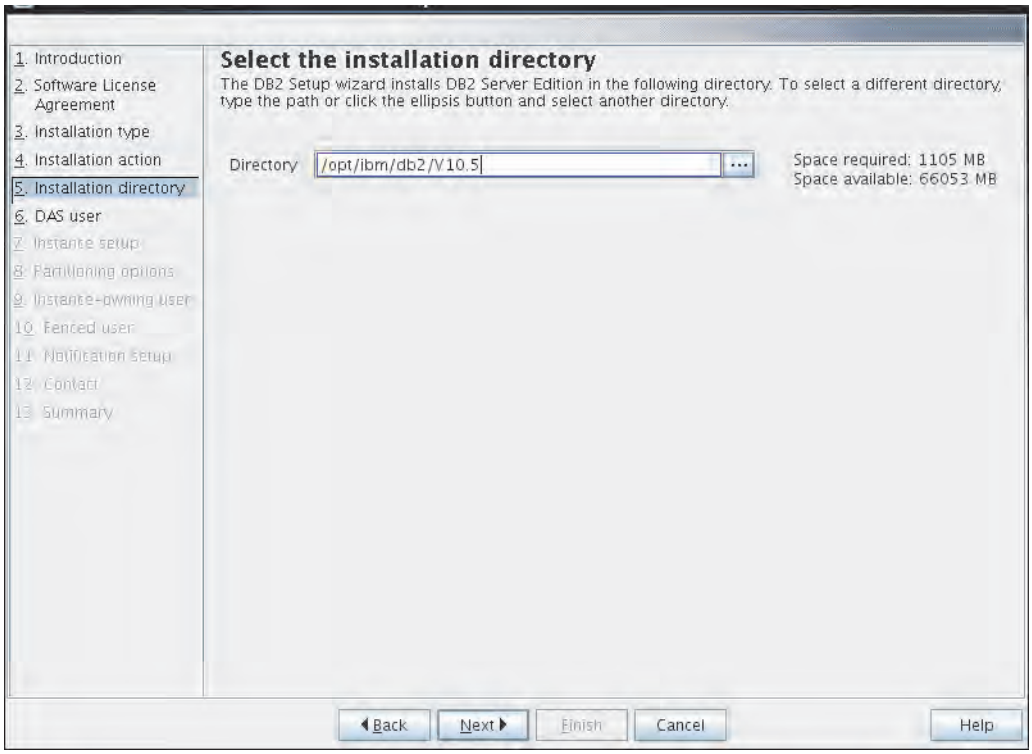
11. Select the typical installation. Click **Next**.



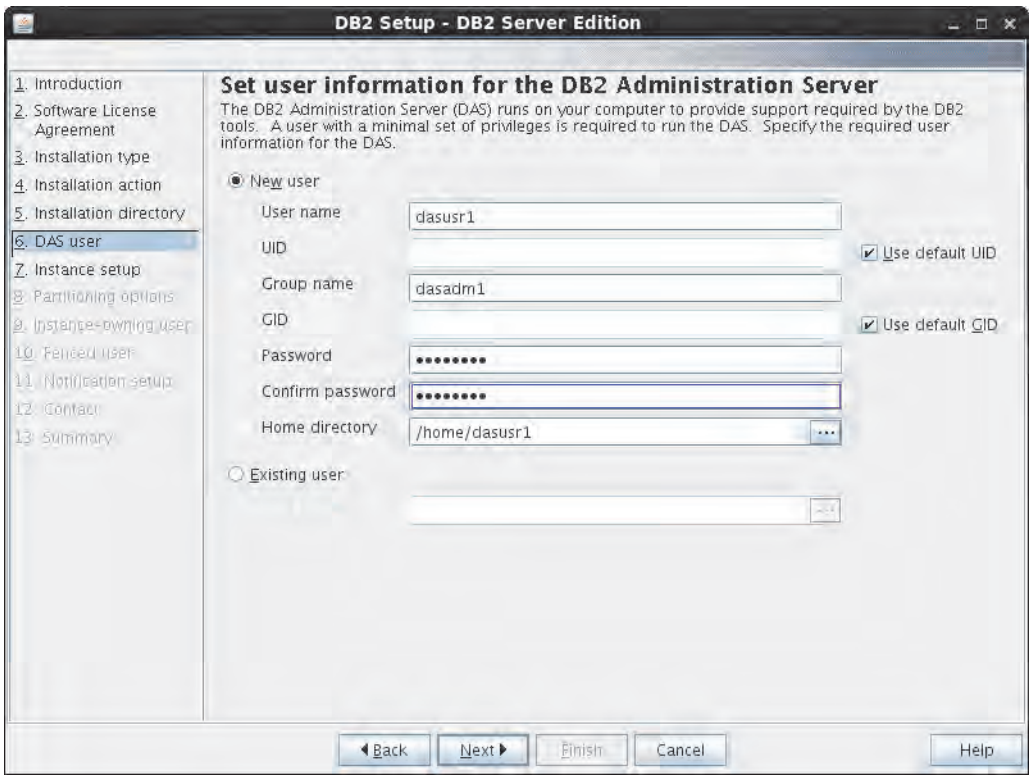
12. Select Install DB2 Server Edition on this computer and save your settings in a response file. Click **Next**.



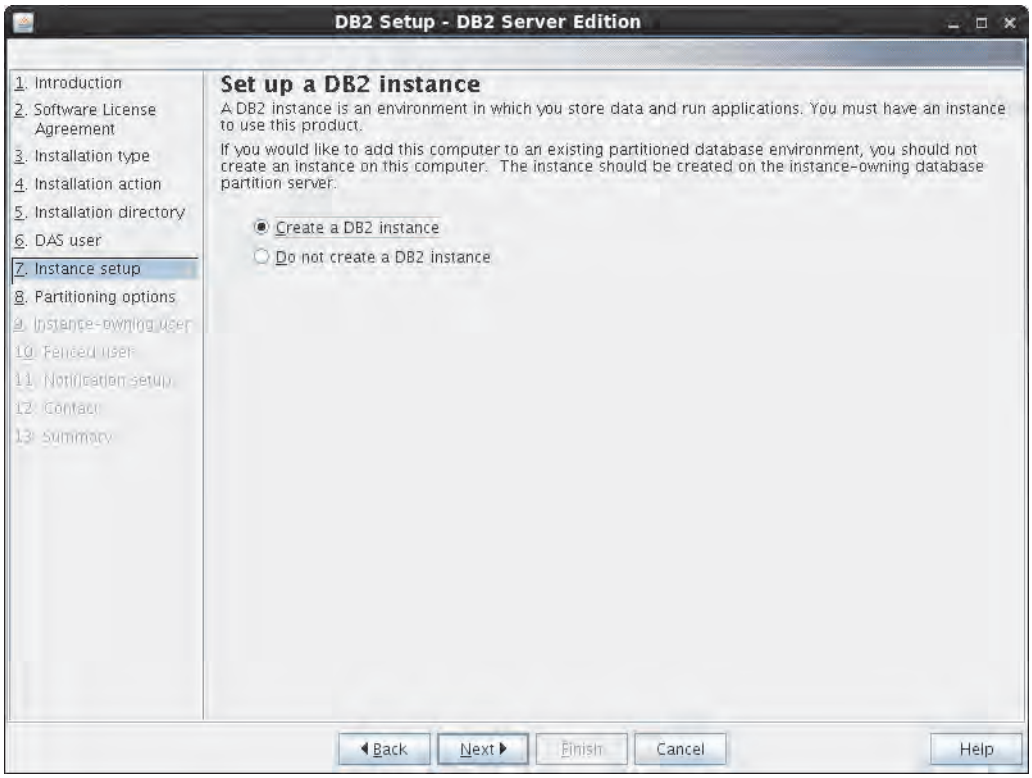
13. Use the default installation directory. Click **Next**.



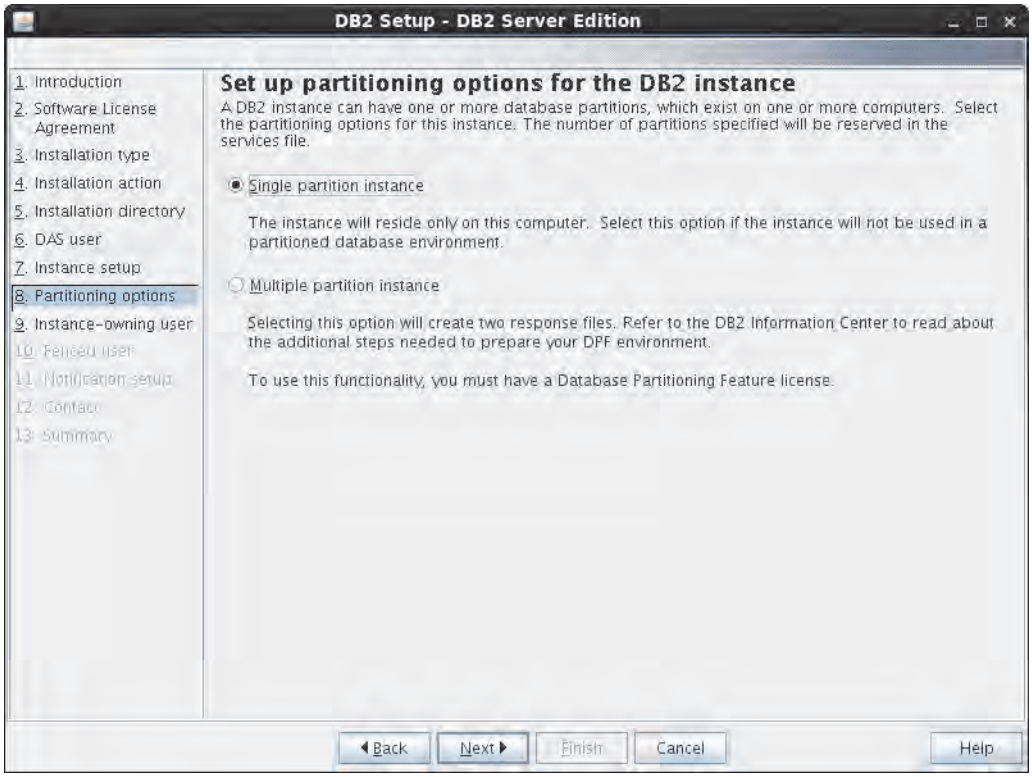
14. Define password for **dausr1** to be **object00**. Click **Next**.



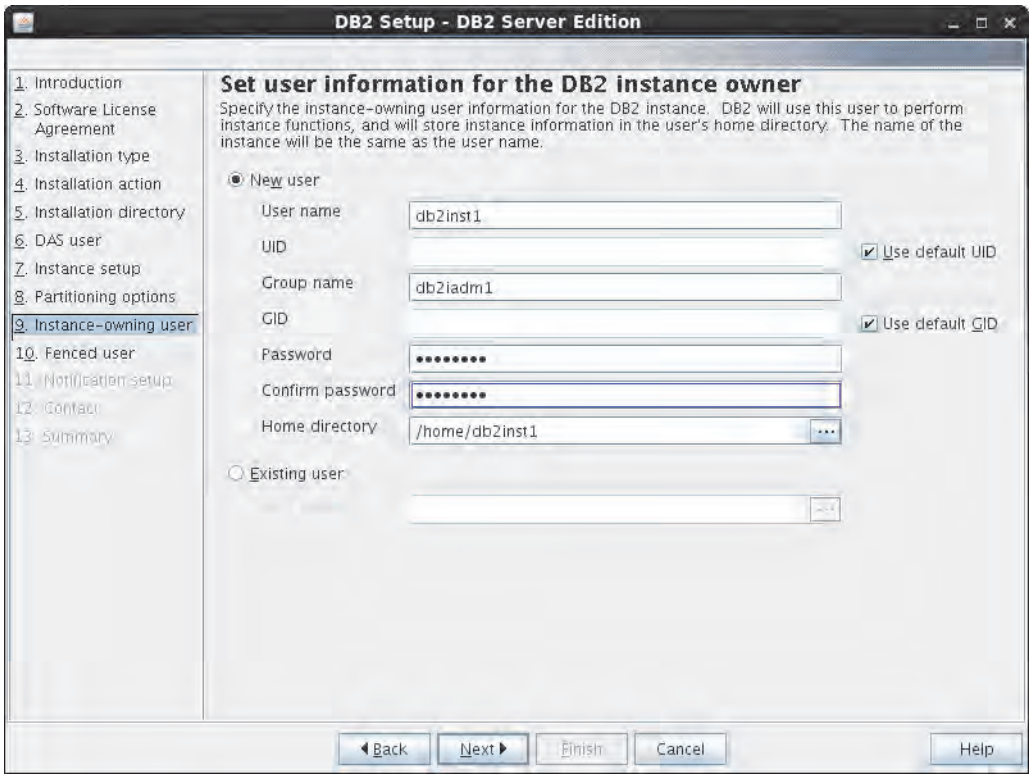
15. Select **Create a DB2 instance**. Click **Next**.



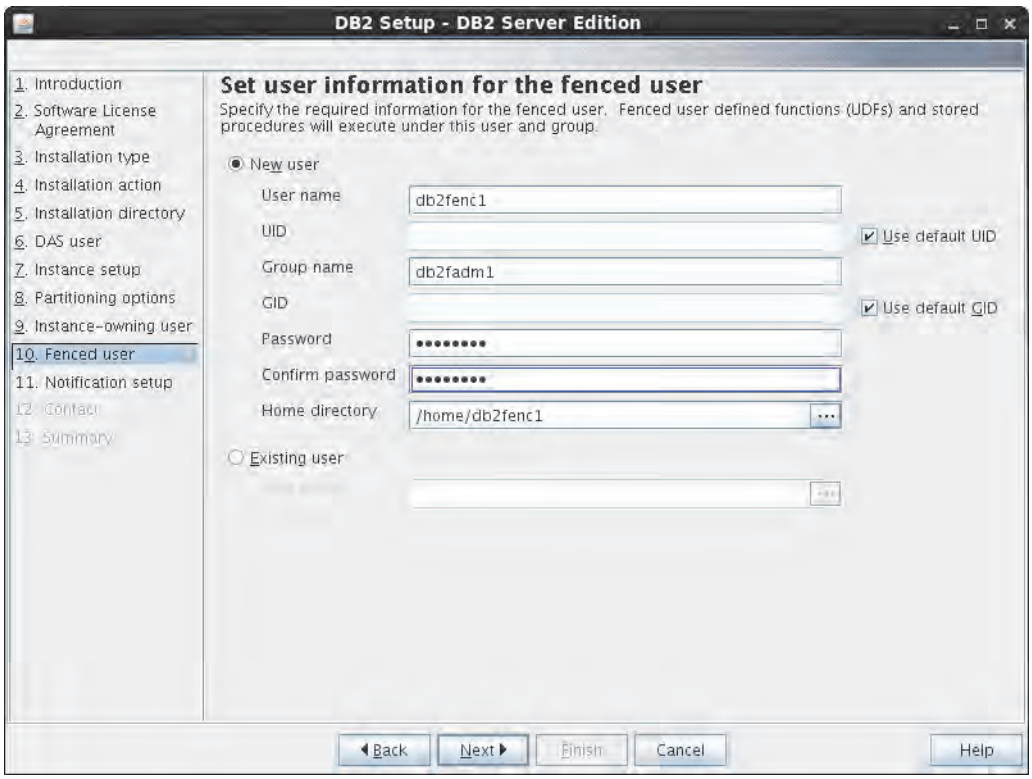
16. Select **Single partition instance**. Click **Next**.



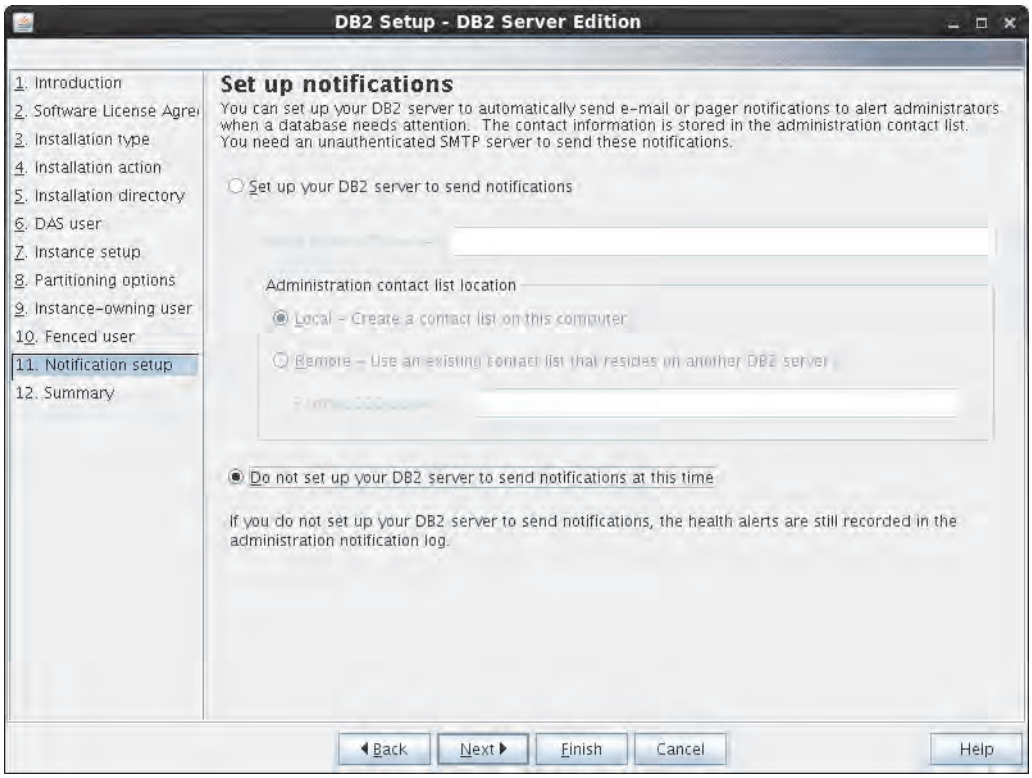
17. Define the password for the **db2inst1** user as **object00**. Click **Next**.



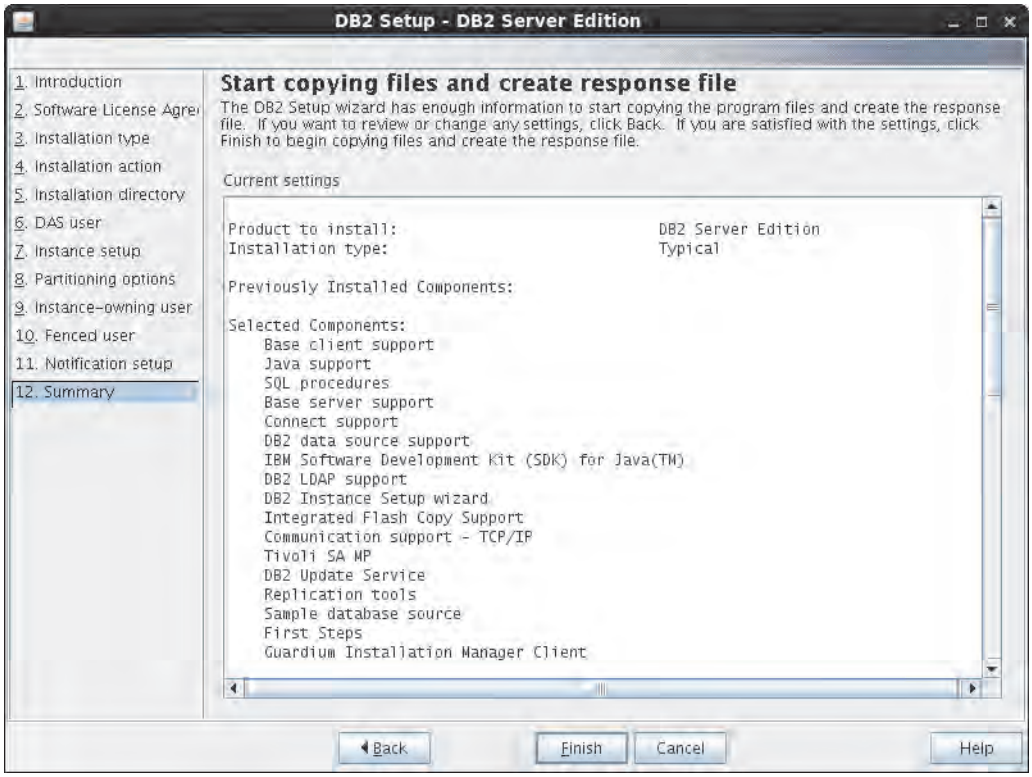
18. Define the password for the **db2fenc1** user as **object00**. Click **Next**.



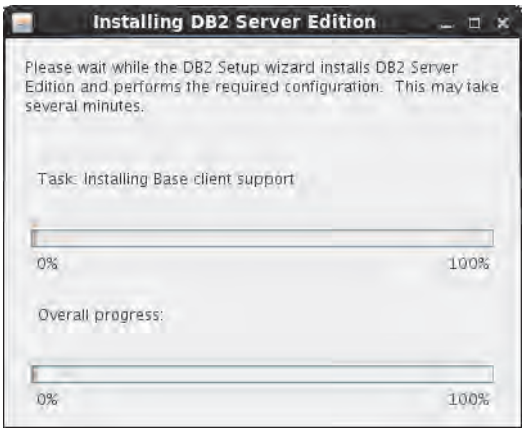
19. Select **Do not set up your DB2 server to send notifications at this time**. Click **Next**.



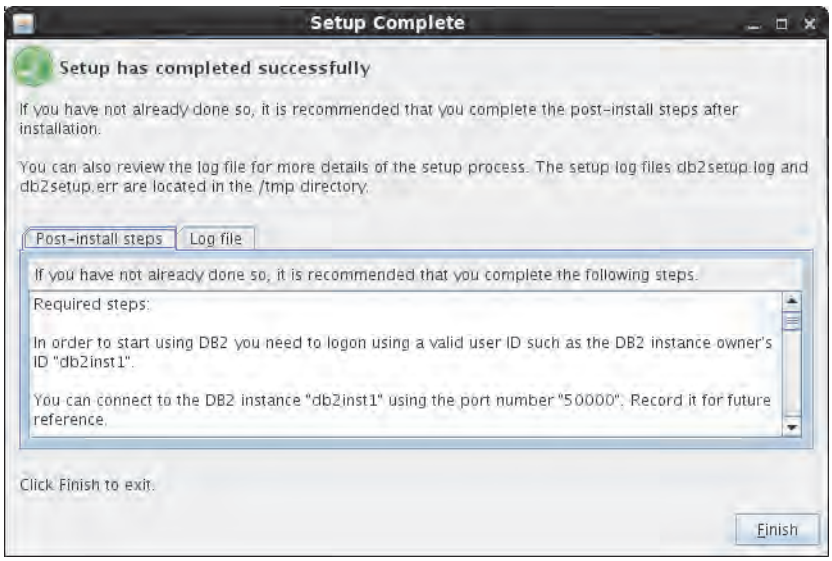
20. Review summary. Click **Finish**.



Installation begins.



21. Click **Finish** to exit the installer.



22. Enable the DB2 monitoring daemon by running the following command as **root**:

```
/opt/ibm/db2/V10.5/bin/db2fmcu -u -p /opt/ibm/db2/V10.5/bin/db2fmcd
```

23. Source the **db2inst1 .bashrc** file so that **root** has access to DB2 functions.

```
. /home/db2inst1/.bashrc
```

24. Run the following command to ensure that the DB2 Administrative Server command is removed because it causes errors in the **/var/log/messages** log:

```
/opt/ibm/db2/V10.5/instance/dasdrop system
DBI1070I Program dasdrop completed successfully.
```

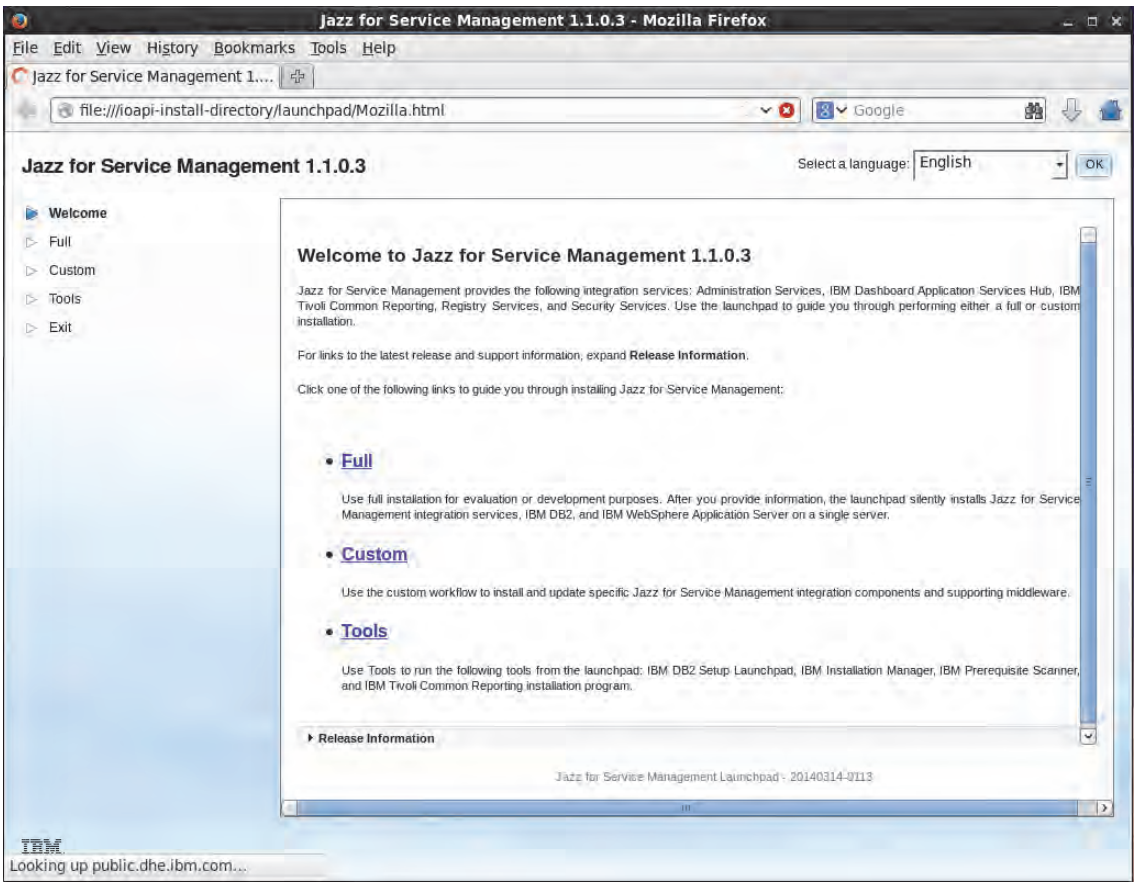
Appendix B Installing and patching Jazz for Service Management

1. Log in as **scadmin**.
2. Download and extract the required software The software required for the Jazz for Service Management integration with IBM Operations Analytics Predictive Insights is as follows:

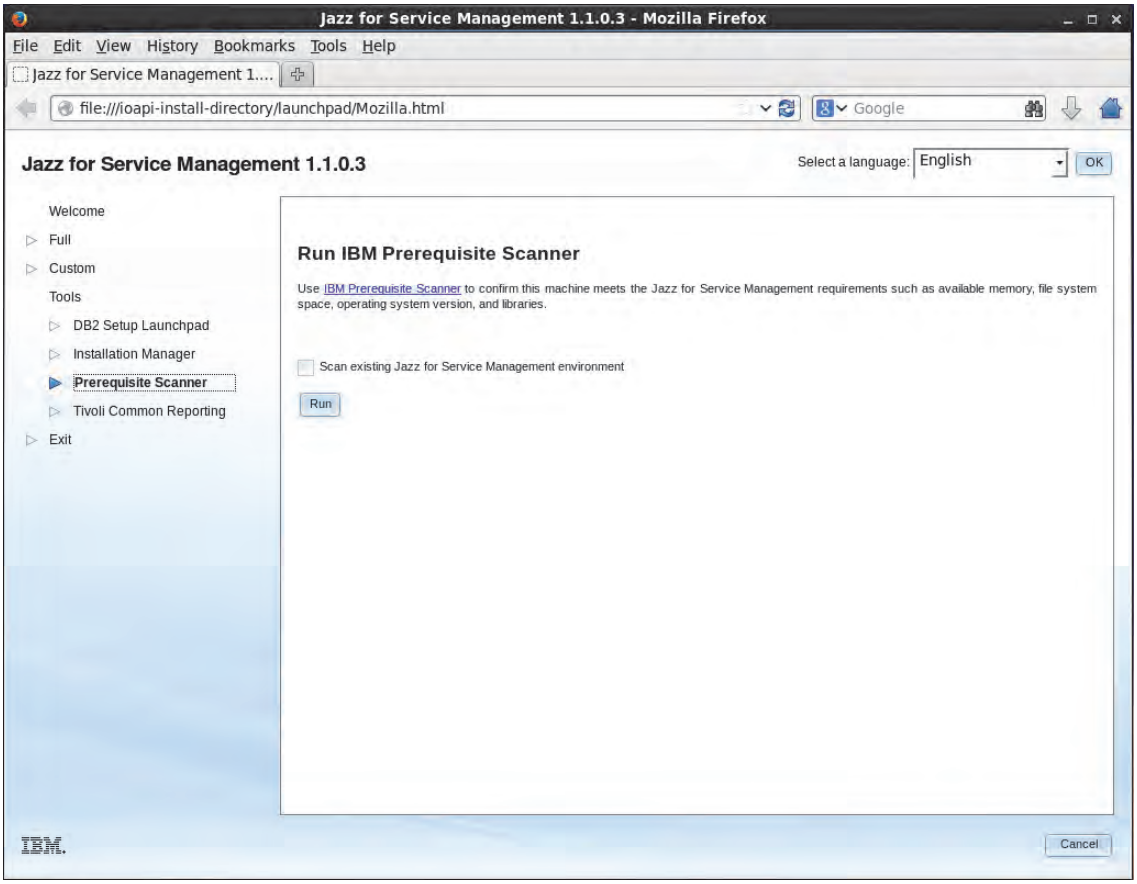
   JAZZ_FOR_SM_1.1.0.3_FOR_LNX.zip

   WAS_V8.5.0.1_FOR_JAZZSM_LINUX_ML.zip
3. Extract both files to a directory on the server where you intend to install Jazz for Service Management.
4. Run the prerequisite scanner. Navigate to the directory where you extracted the Jazz for Service Management zip file. Run the command to start launchpad, for example:

   `./launchpad.sh`



5. Click **Tools > Prerequisite Scanner** and click the **Run** button.

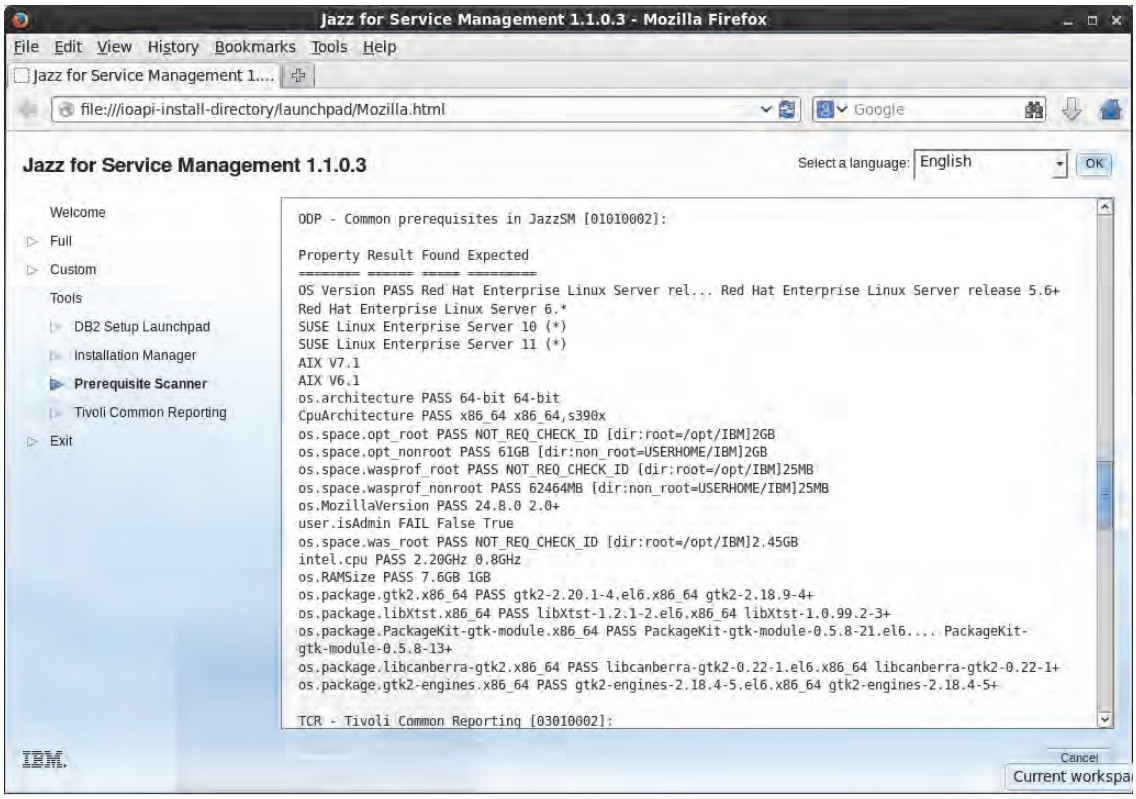


6. Review the results of the scan. These are the two sections of the prerequisite scanner output that you must check for failures:

   DSH: Dashboard Application Services Hub in JazzSM (ignore intel.cpu FAIL 2.20GHz 2.4GHz)

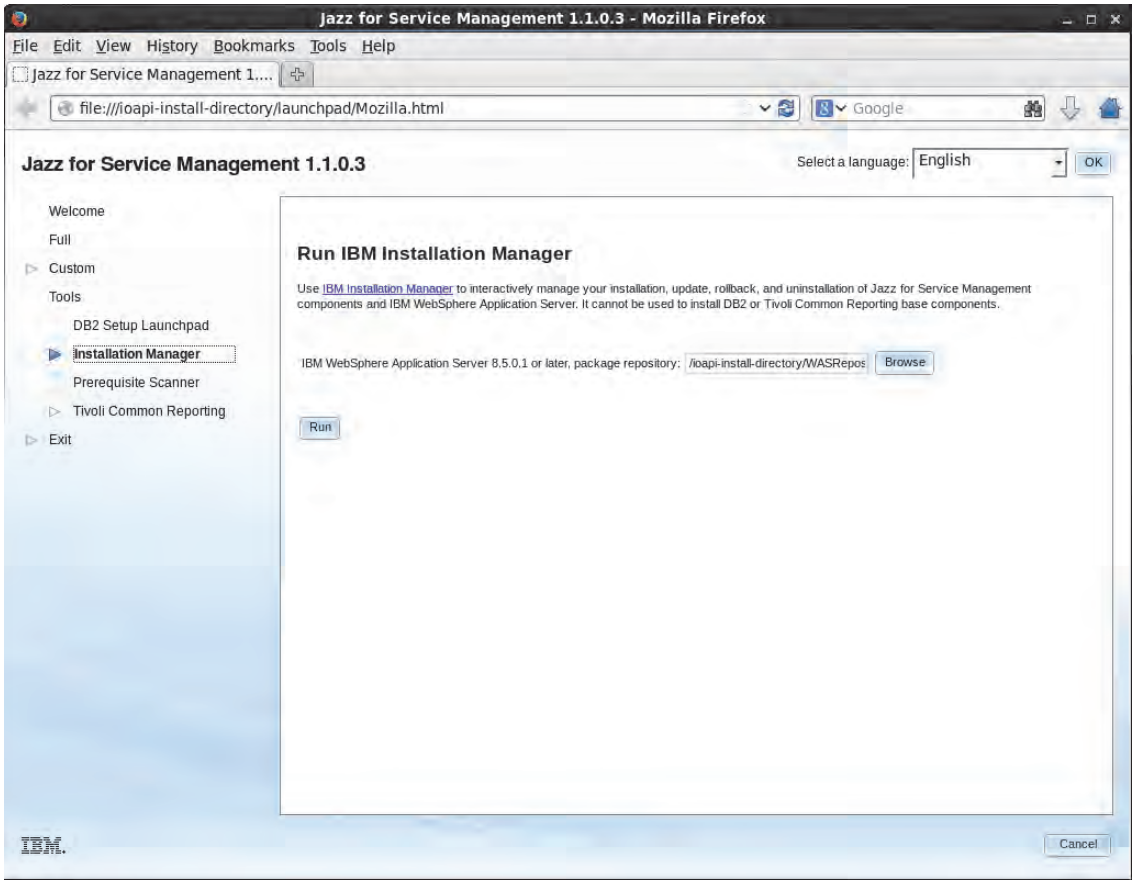
   ODP: Common prerequisites in JazzSM (ignore user.isAdmin FAIL False True)

TCR is not installed as part of Predictive Insights and the packages associated to that software can be ignored for this effort.



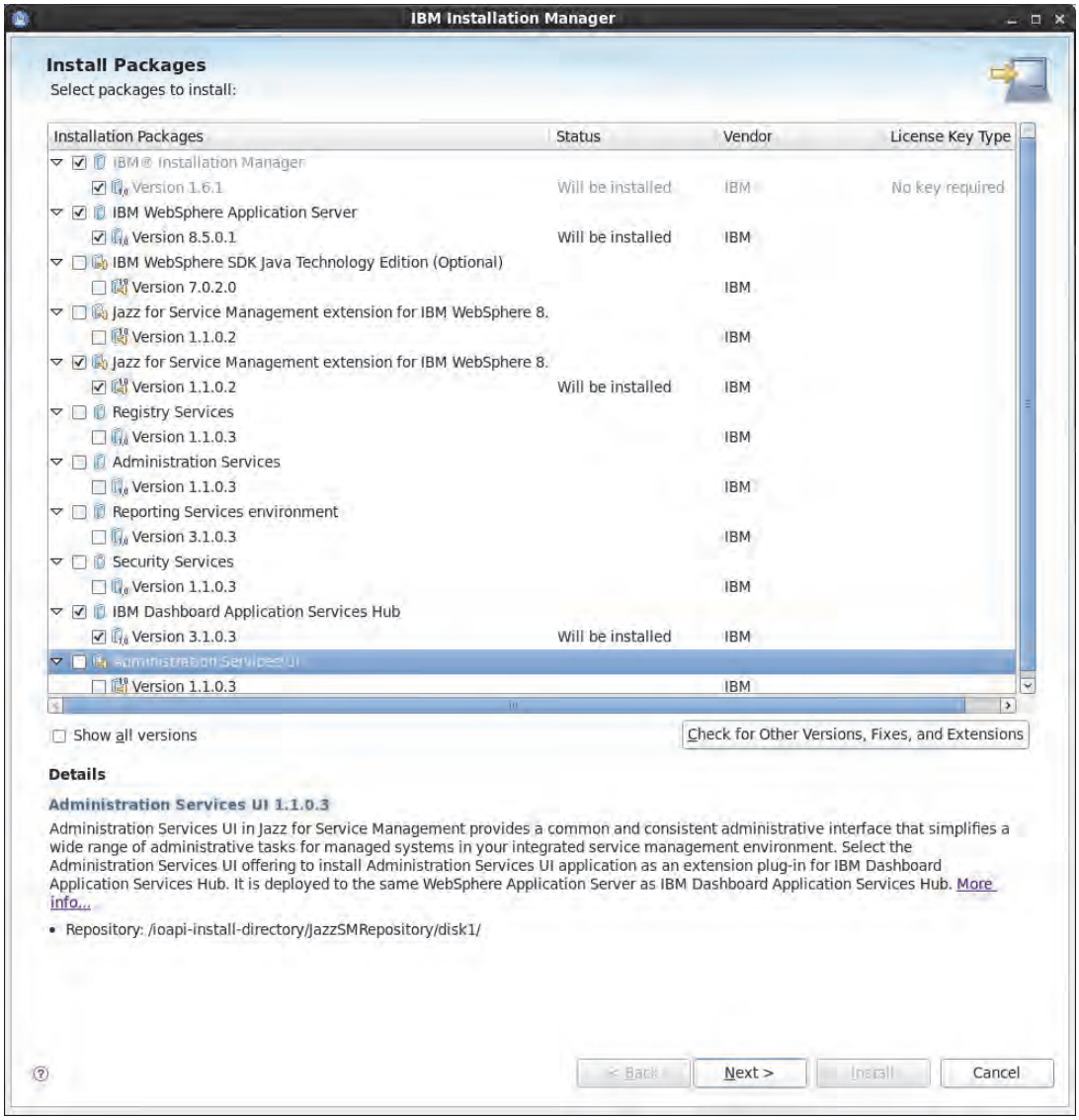
7. Select **Tools > Installation Manager**. Make sure the package repository point to the following location. Click **Run**.

   /ioapi-install-directory/WASRepository/disk1/diskTag.inf

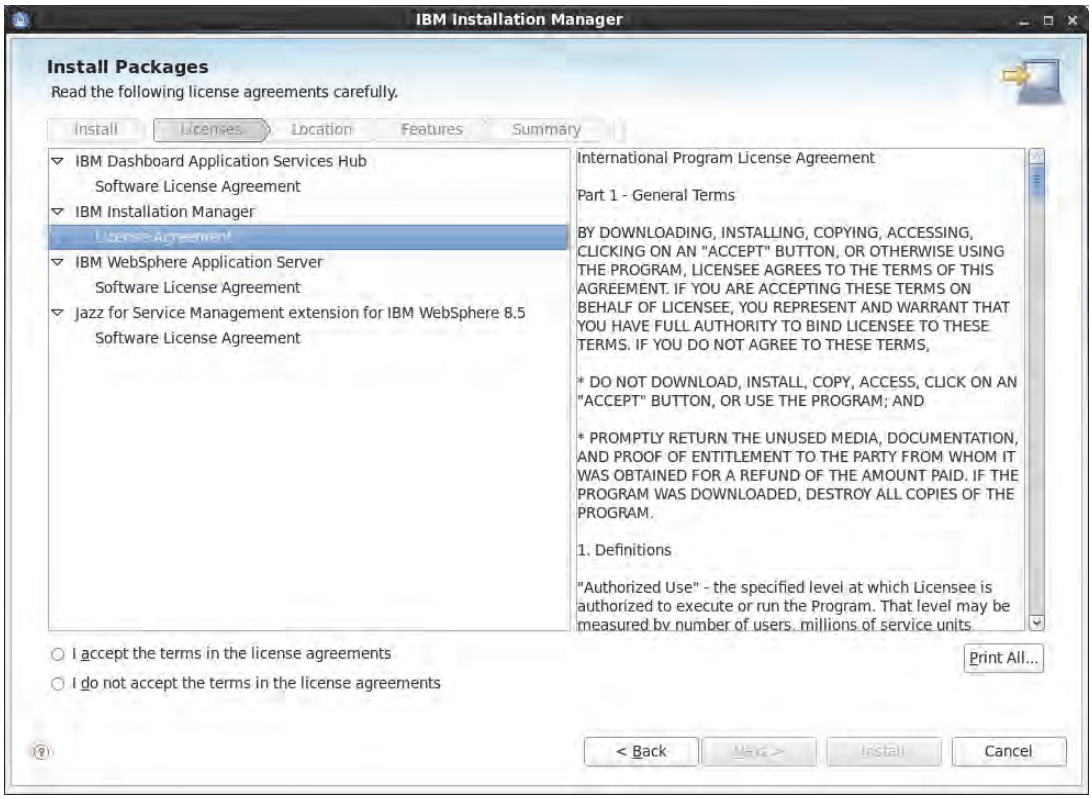


8. Ensure that only the following packages are selected. Click **Next**.

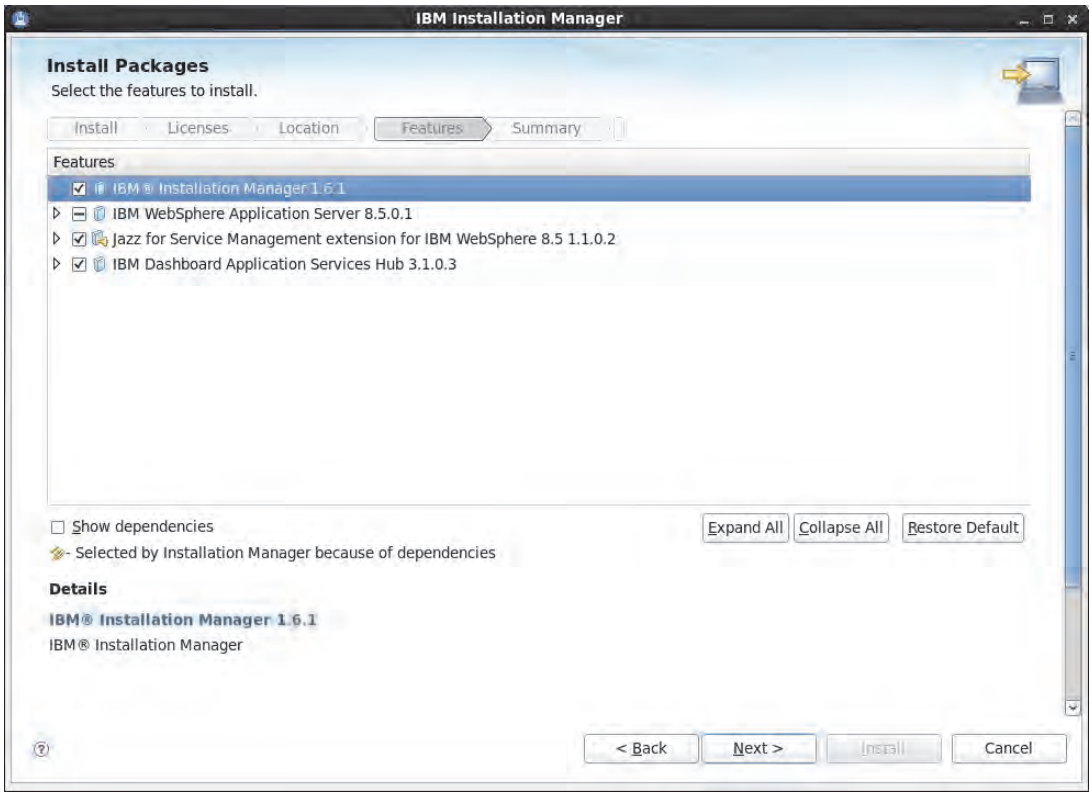
   IBM Installation Manager
   IBM WebSphere Application Server
   Jazz for Service Management extensions for IBM WebSphere
   IBM Dashboard Application Service Hub



9. Accept license agreement and click **Next**.

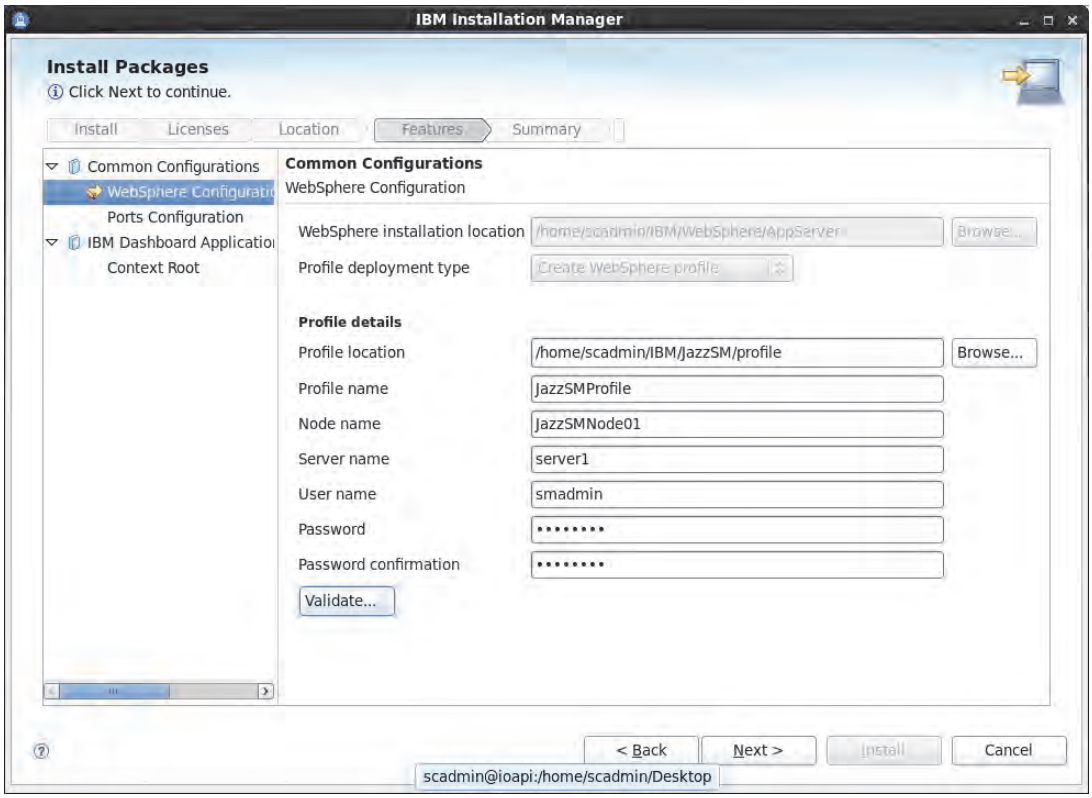


10. Review the packages that will be installed and click **Next**.

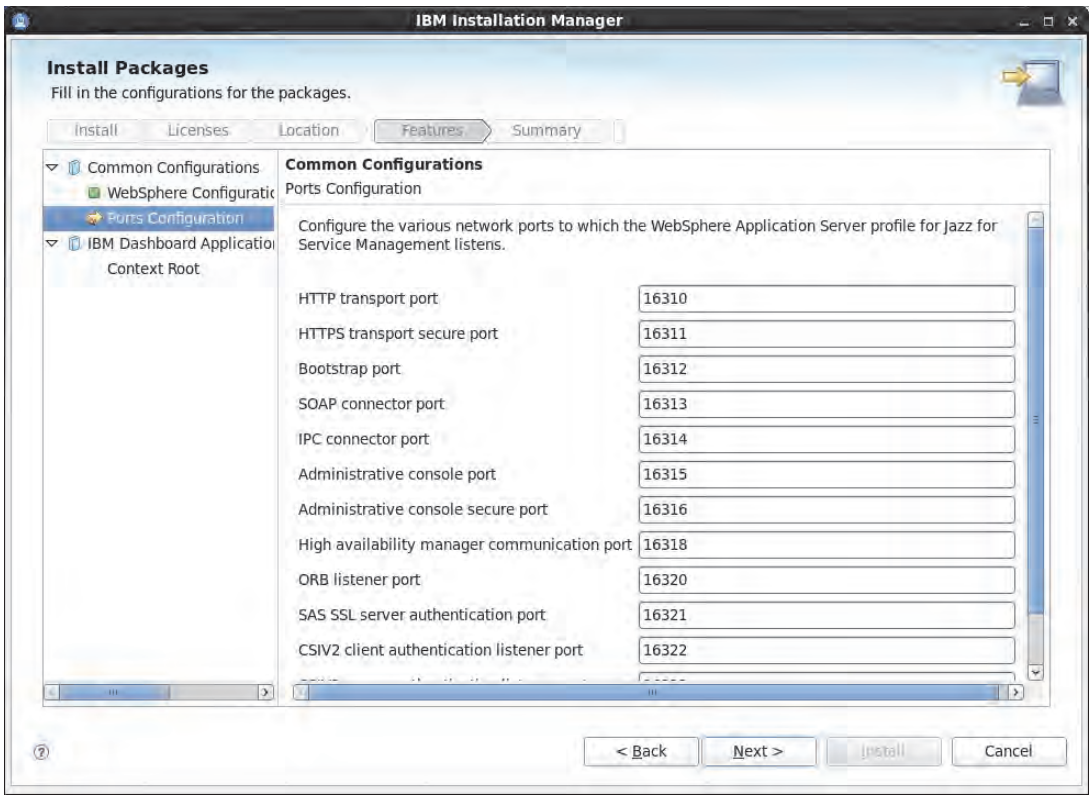


Note the default profile that will be created for Jazz for Service Management.

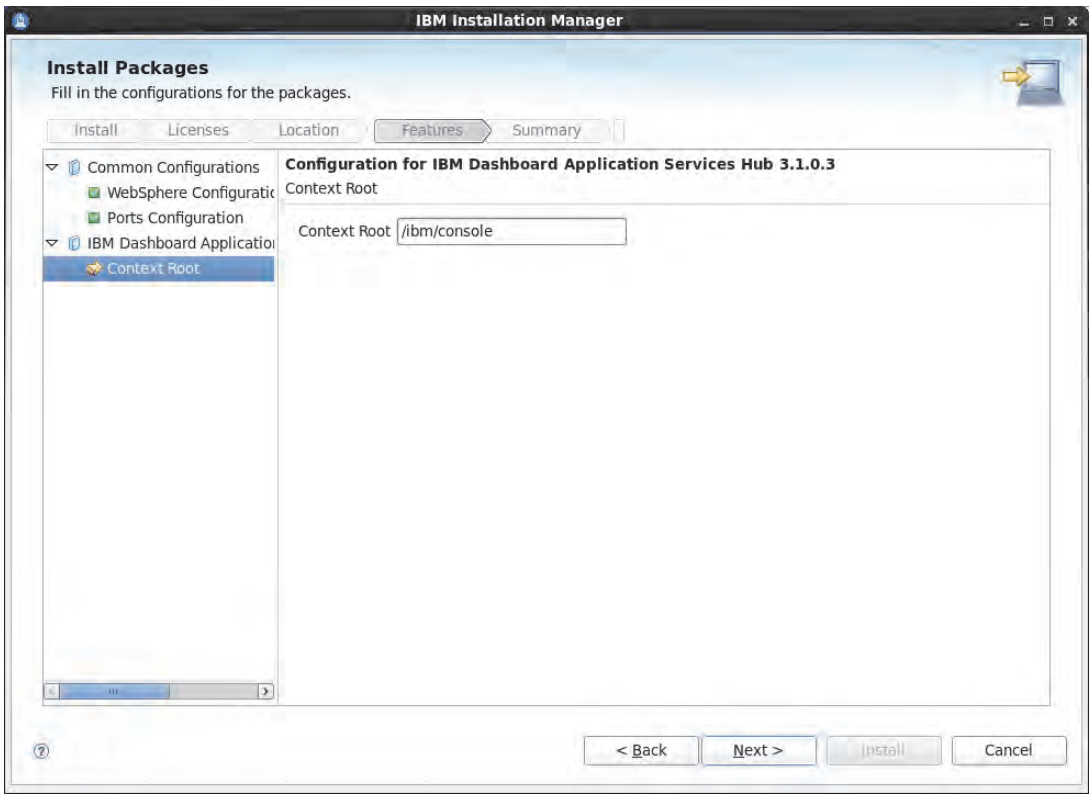
11. Change the profile location to be **/home/scadmin/IBM/JazzSM/profile**. Define the password for this profile (**object00**) and validate it. Click **Next**.



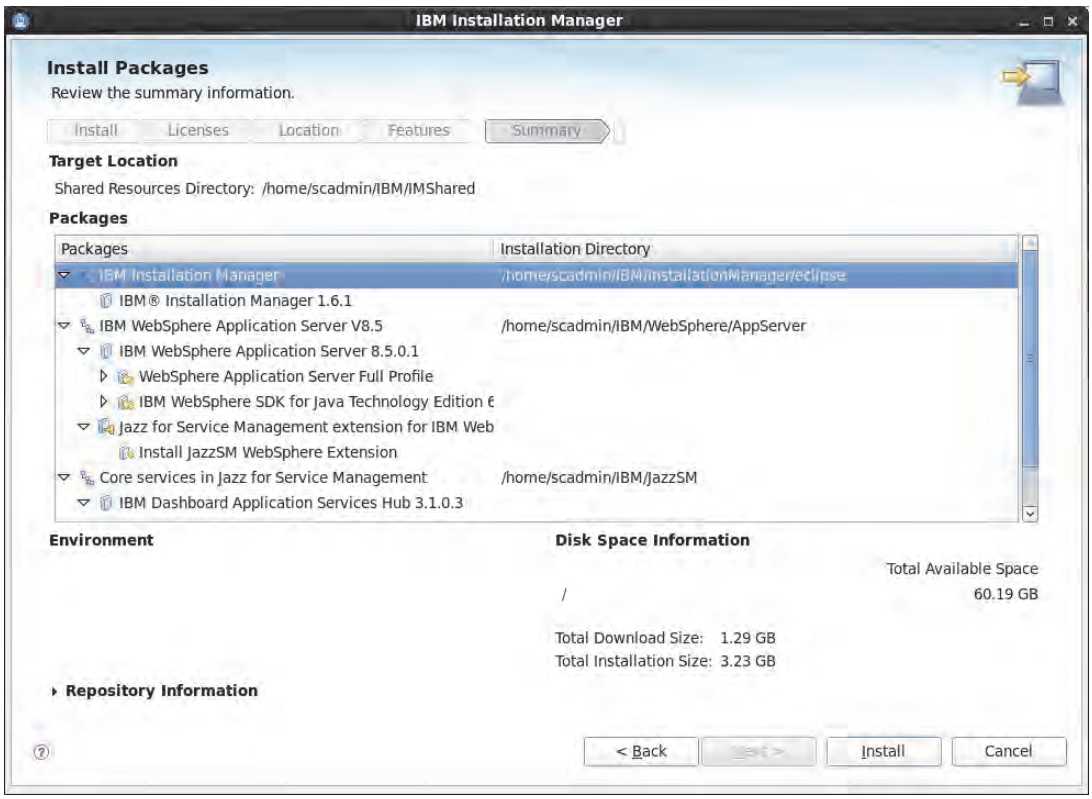
12. Review the default ports that are used for the application. Click **Next**.



13. Leave the default root context for the application. Click **Next**.



14. Review the summary. Click **Install**.



The installation took 60-70 minutes to complete on a laptop virtual machine.

15. When the installation is complete, do not create any profiles and just finish the installer.

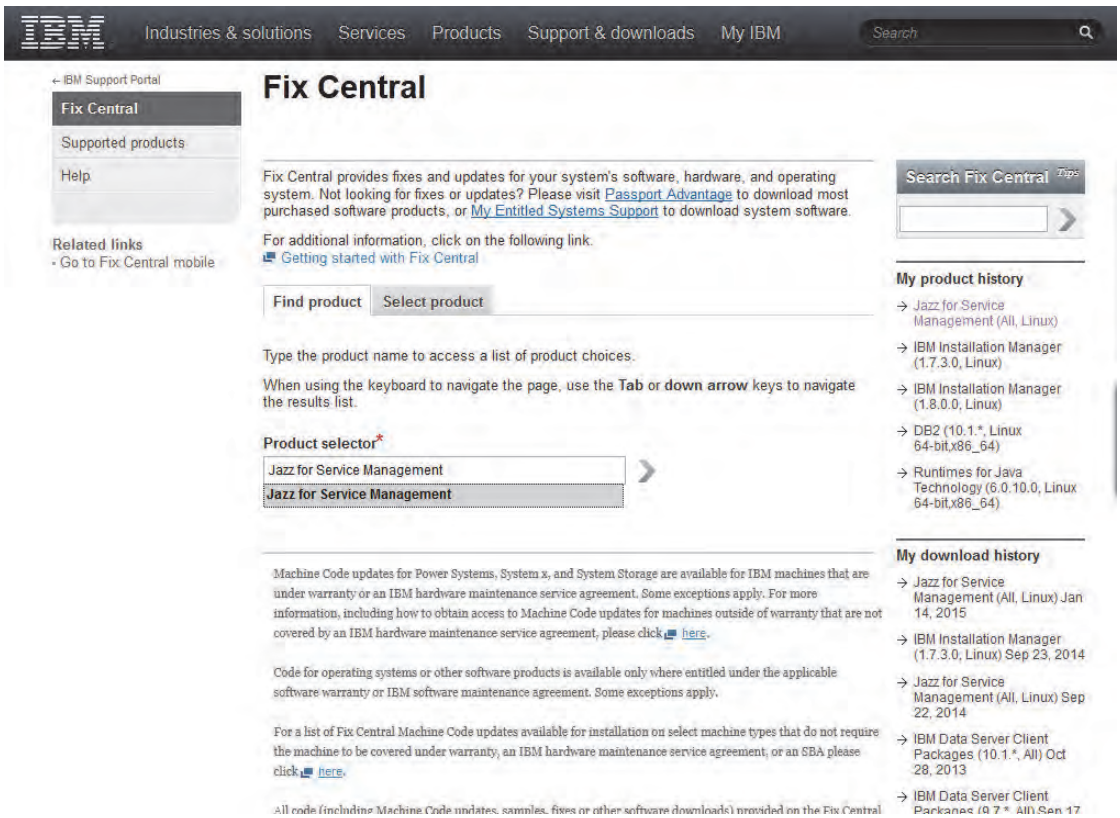
16. Close the Jazz for Service Management 1.1.0.3 Launchpad browser.

Patching Jazz for Service Management

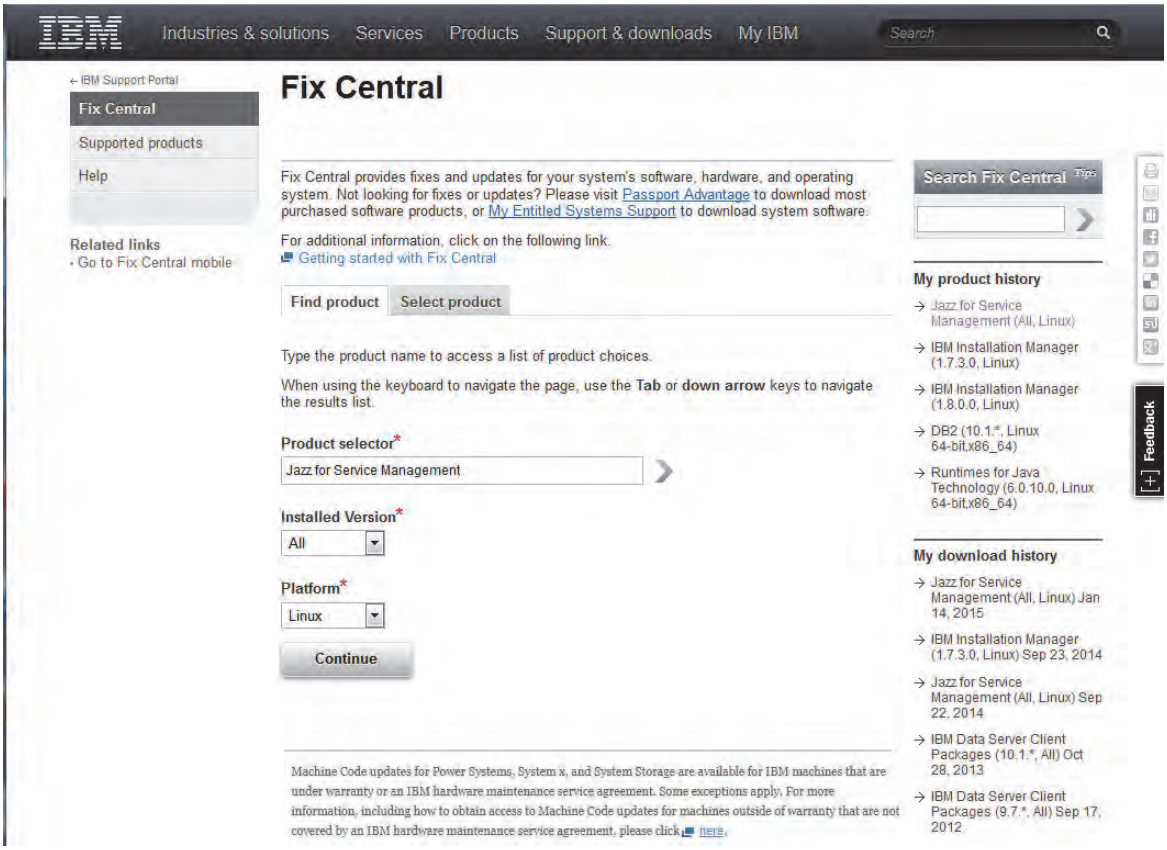
1. Obtain the patch software.
   a. Go to IBM Support Fix Central.

      http://www-933.ibm.com/support/fixcentral/options?selection=Software%3bibm%2fTivoli%3bibm%2fTivoli%2fTivoli+Integrated+Portal

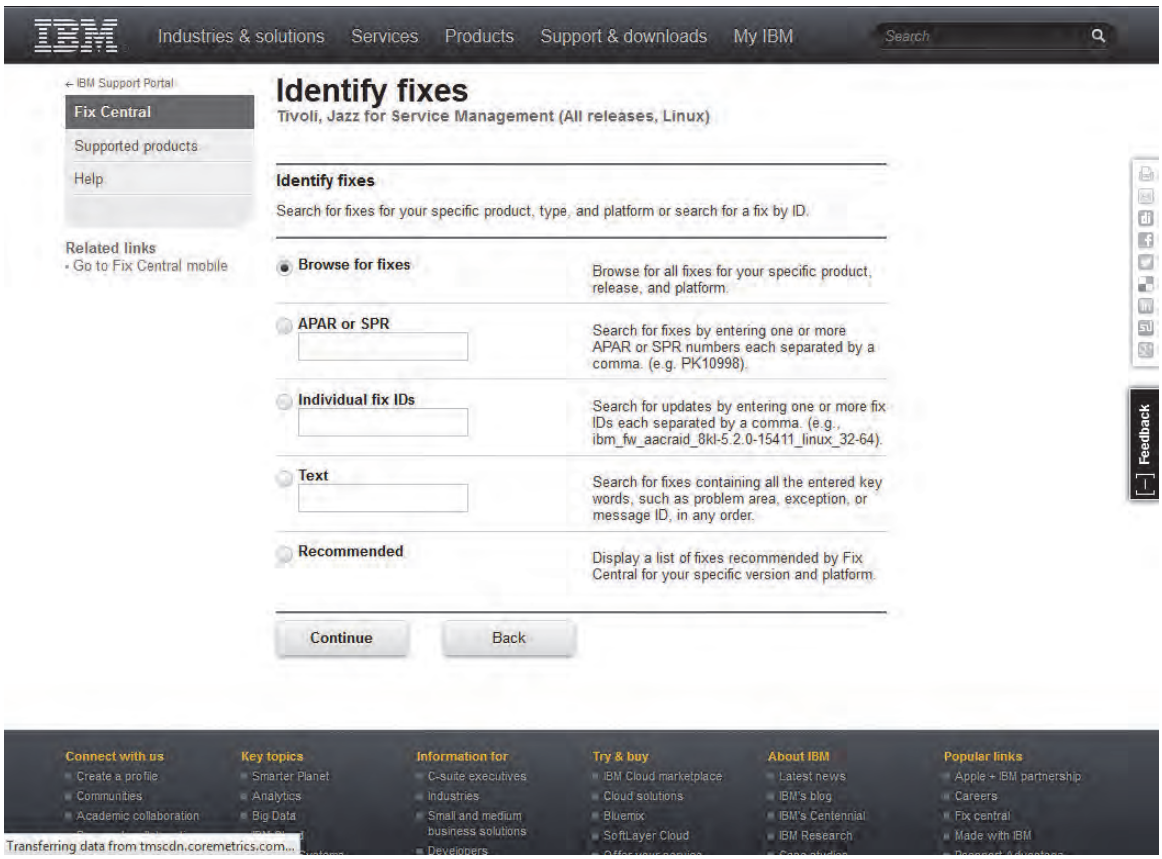
b. In the **Product Selector** field, enter **Jazz for Service Management**.



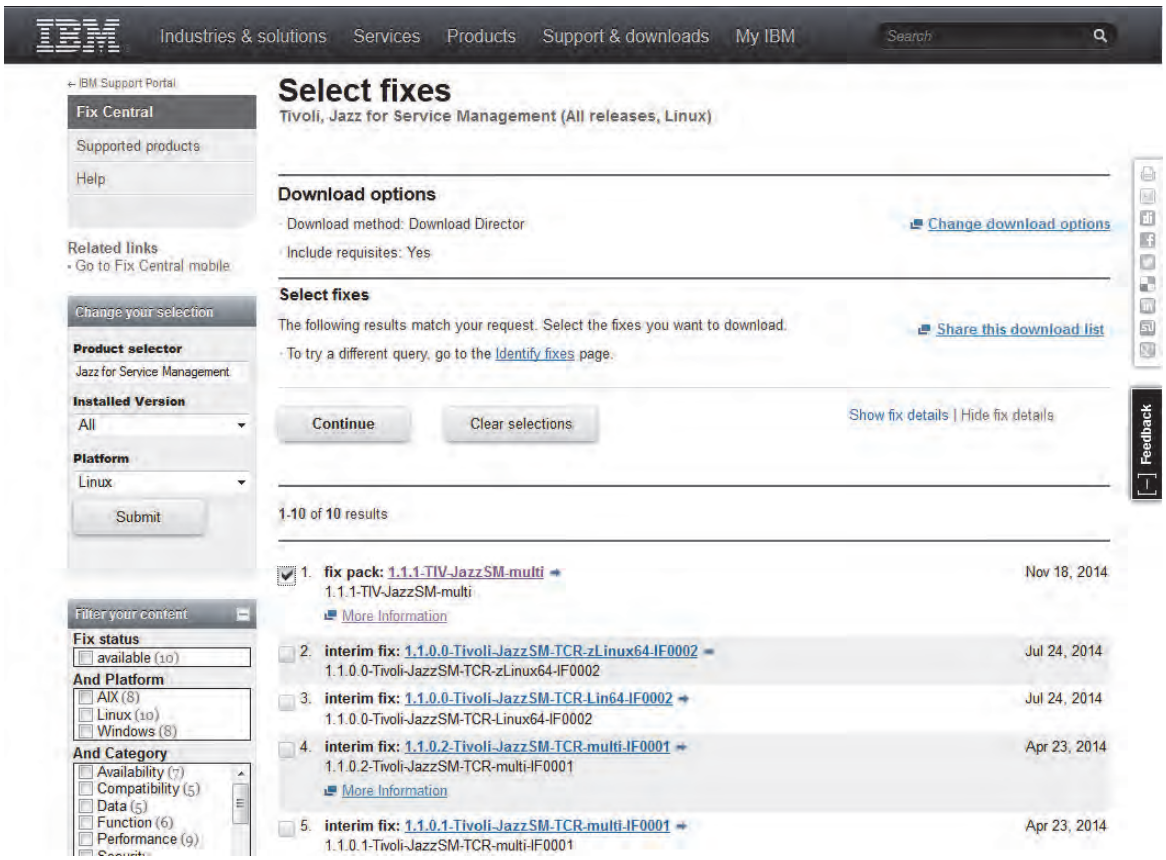
c. Enter **All for Installed Version and Linux for Platform**. Click **Continue**.



d. Select **Browse for fixes**. Click **Continue**.



e. Select fix pack **1.1.1-TIV-JazzSM-multi**. Click **Continue**3



f. Authenticate with your Fix Central credentials.

g. Click **Download now**. You receive the patch for Windows, AIX, and Linux systems.

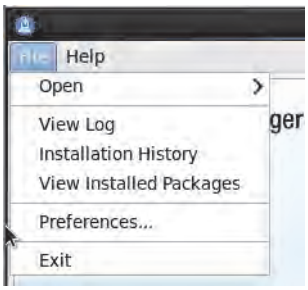
```
E:\hgfs-files\SCAPI\1.3\1.1.0-TIV-JazzSM-TCR-AIX64-FP002.tar.gz
E:\hgfs-files\SCAPI\1.3\1.1.0-TIV-JazzSM-TCR-LINUX64-FP002.tar.gz
E:\hgfs-files\SCAPI\1.3\1.1.0-TIV-JazzSM-TCR-LINUXzSeries64-FP002.tar.gz
E:\hgfs-files\SCAPI\1.3\1.1.0-TIV-JazzSM-TCR-WIN64-FP002.zip
E:\hgfs-files\SCAPI\1.3\1.1.1-TIV-JazzSM-multi.zip
E:\hgfs-files\SCAPI\1.3\1.1.1.0-TIV-JazzSM-RME-multi.zip
```

h. Move the **1.1.1-TIV-JazzSM-multi.zip** file to the server.

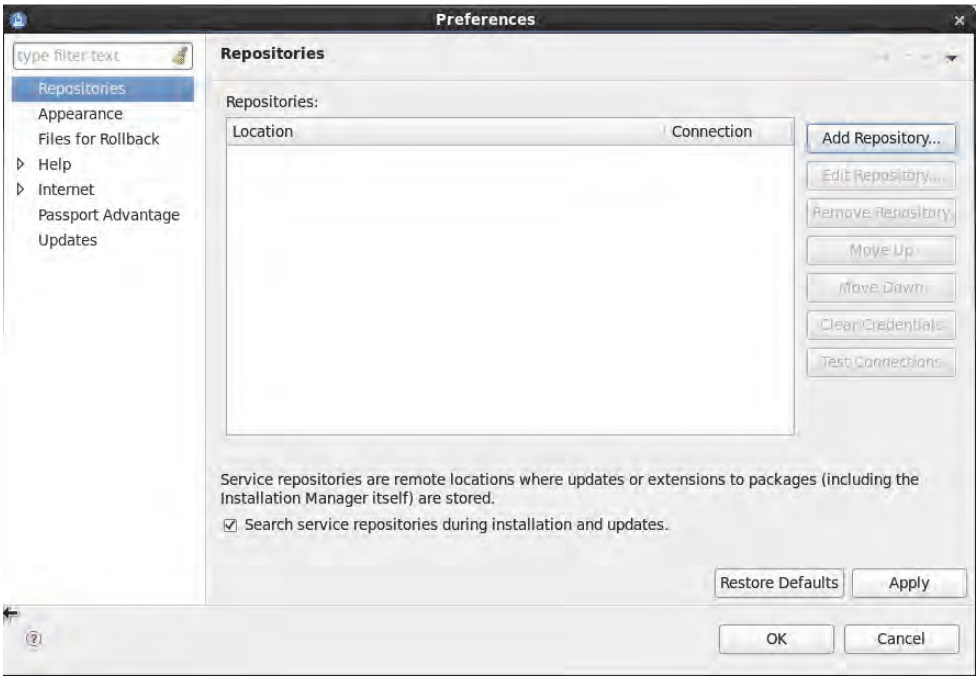
i. As the **scadmin** user, extract the file in a directory on server.

2. As the **scadmin** user, start the Installation Manager by running the command `/home/scadmin/IBM/InstallationManager/eclipse/launcher`

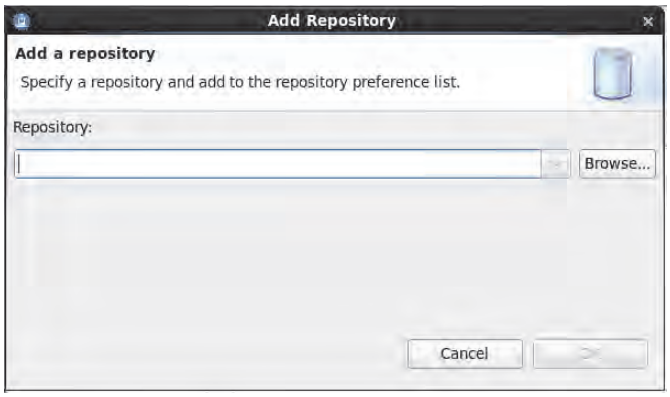
3. Reconfigure the Installation Manager to point to patch repository. Select **File > Preferences**.

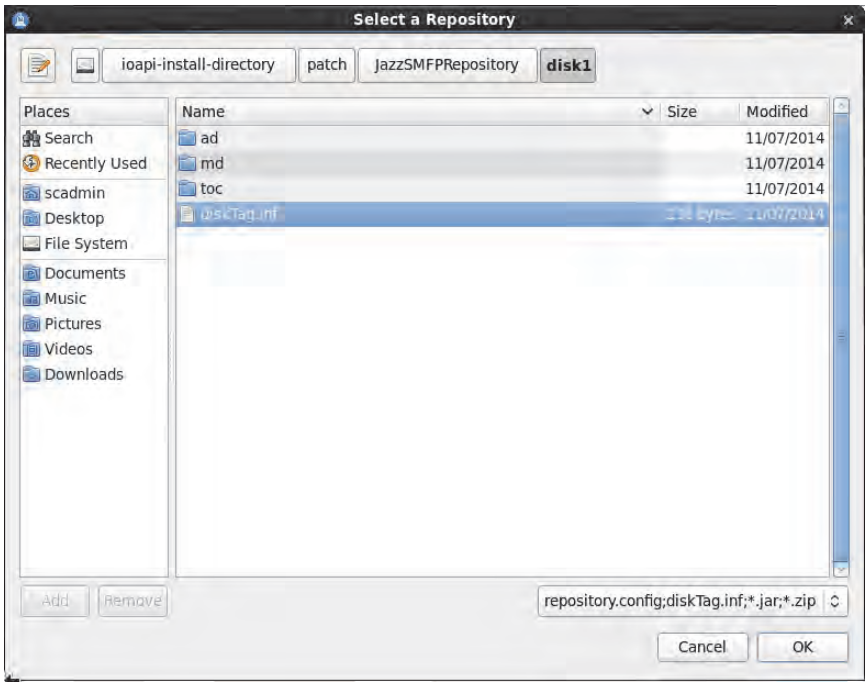


4. Click **Add Repository**.

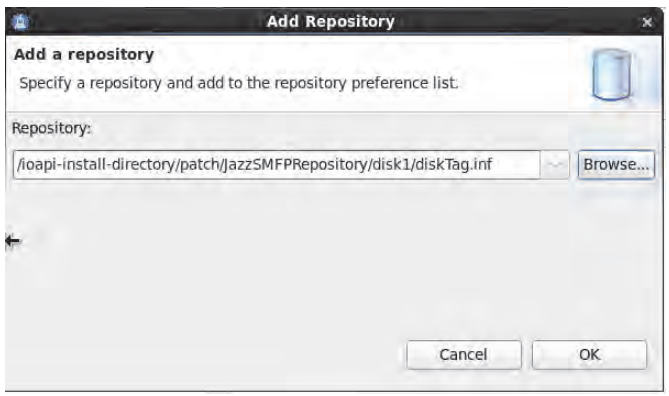


5. Browse to the location where you extracted the patch. In the JazzSMFPRepository, click **disk1** and the **diskTag.inf** file. Click **OK**.

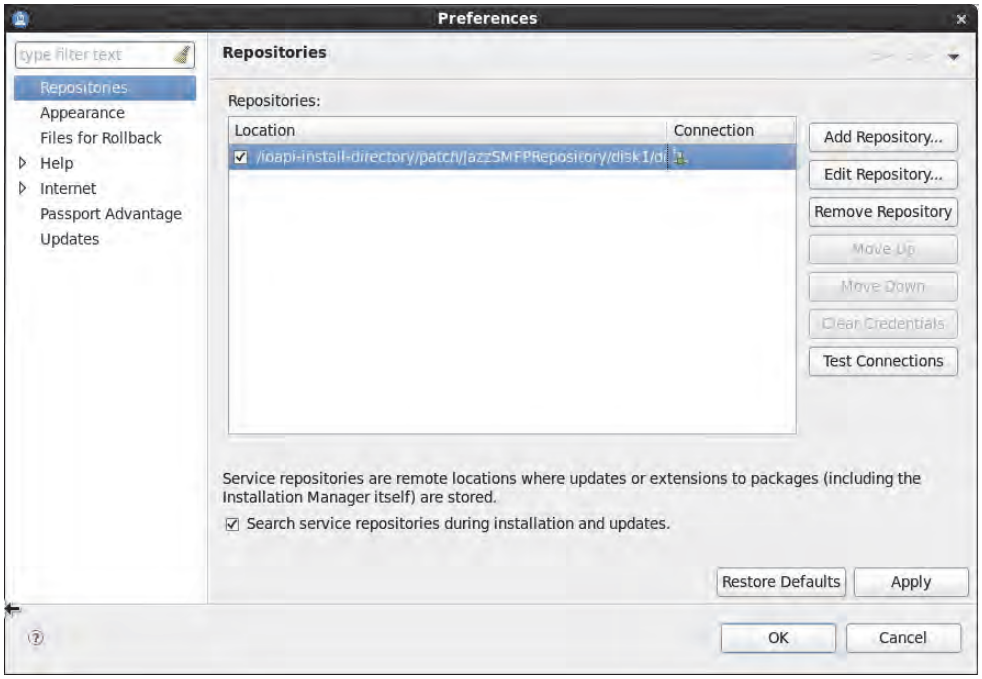




6. Click **OK**.

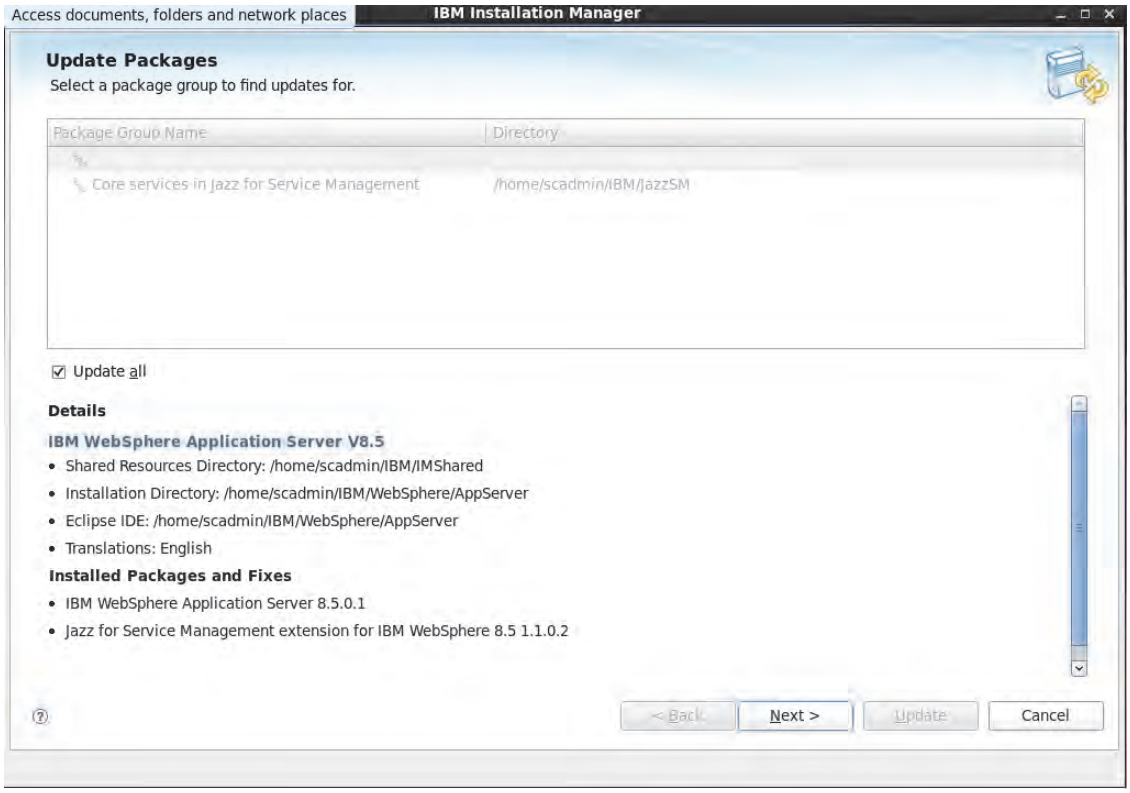


7. Click **OK**.



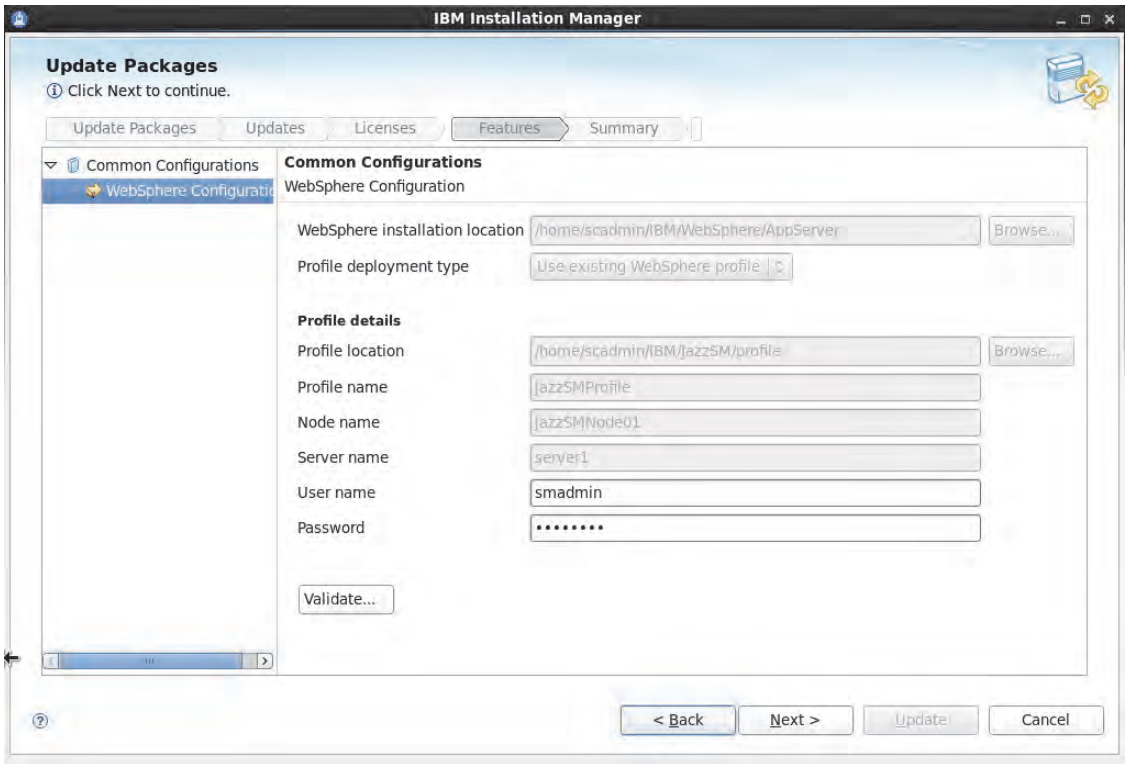
8. Click **Update**.

9. Select **Update all**. Click **Next**.

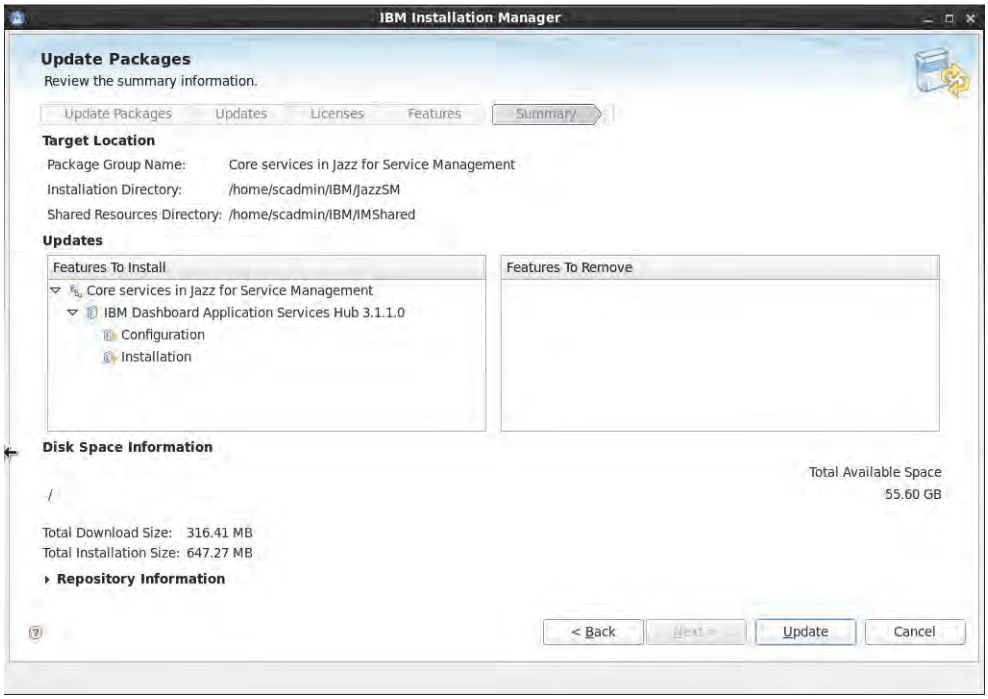


10. Accept license agreement. Click **Next**.

11. Enter the password for the **smadmin** profile that you created in the previous Jazz installation (**object00**). Click **Validate**. Click **Next**.



12. Click **Update**.



Update begins. Update takes 90 minutes.

Appendix C Installing and configuring OMNIbus

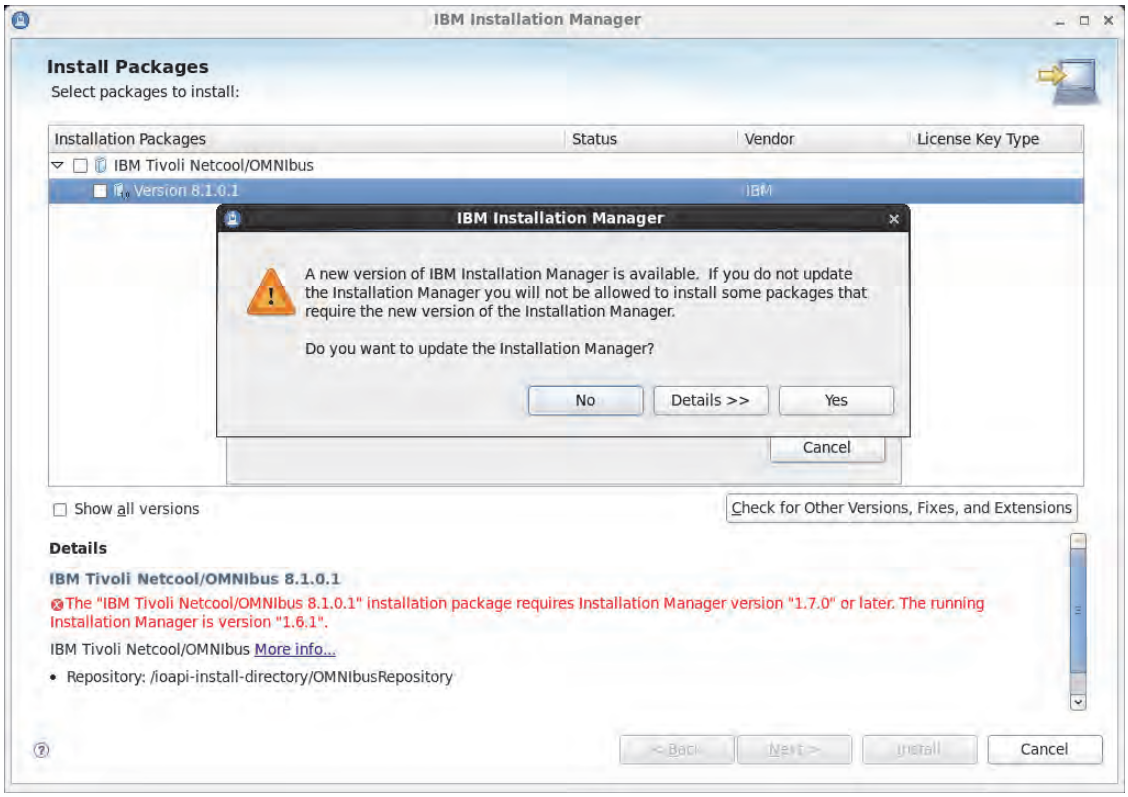
1. As **root**, create the directory /opt/IBM and change its ownership to scadmin:scadmin:

   ```
   mkdir /opt/IBM
   chown scadmin:scadmin /opt/IBM
   ```

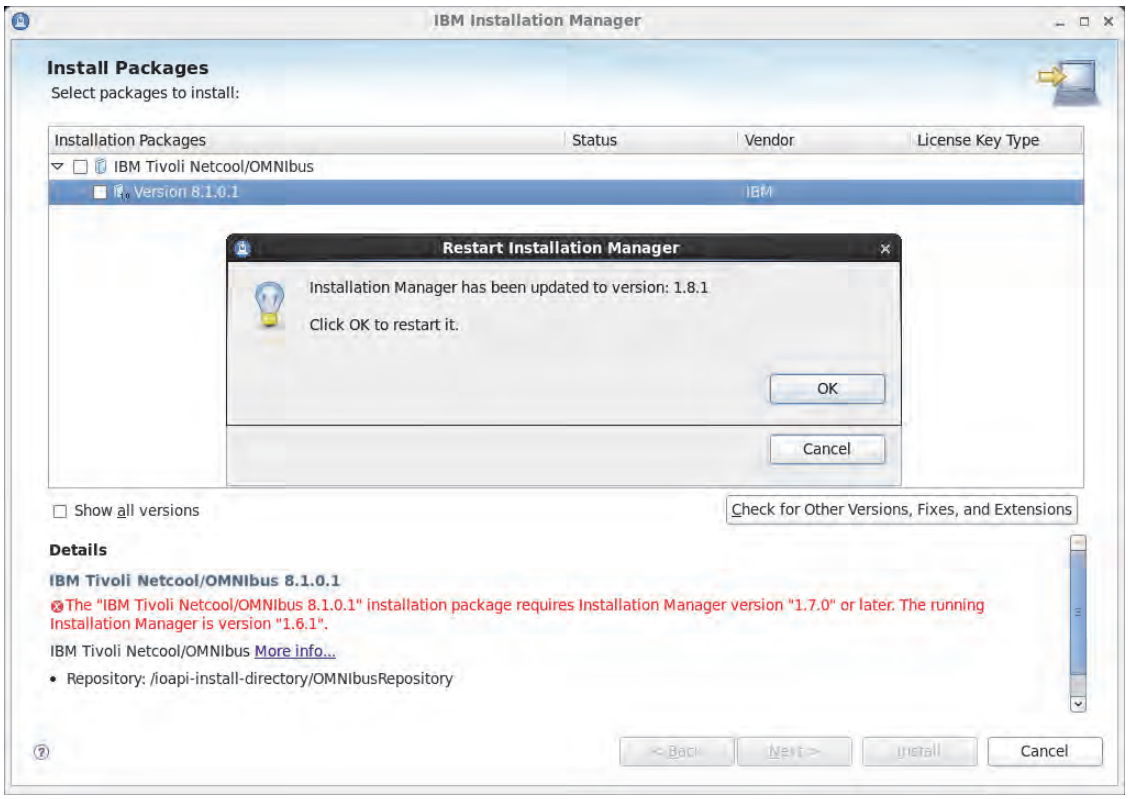
2. Log out as **root** and complete the following steps as the **scadmin** user.

3. Add installer **OMNIbus-v8.1-Core.linux64.zip** file to an installation directory. Extract the file.

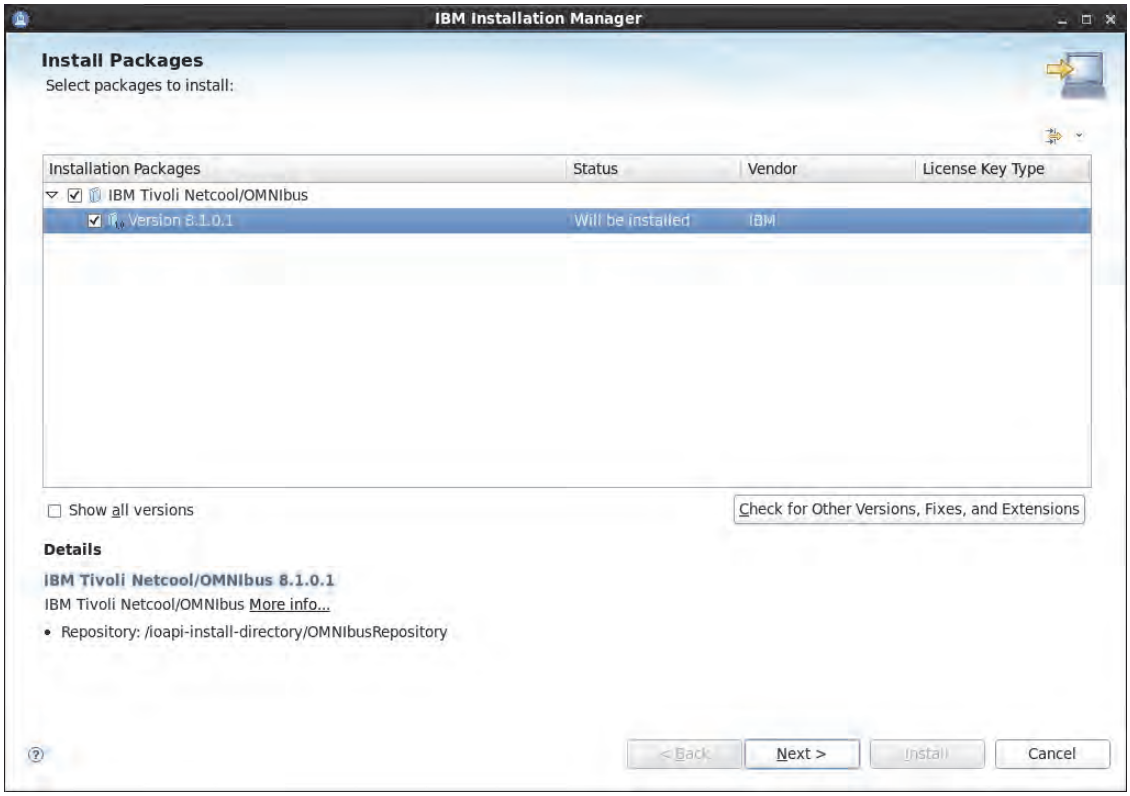
4. Run install_gui.sh. The current Installation Manager is 1.6.1. OMNIbus requires 1.7.0 or later. You are warned about this and are prompted to upgrade. Select yes. Access to the Internet is required.



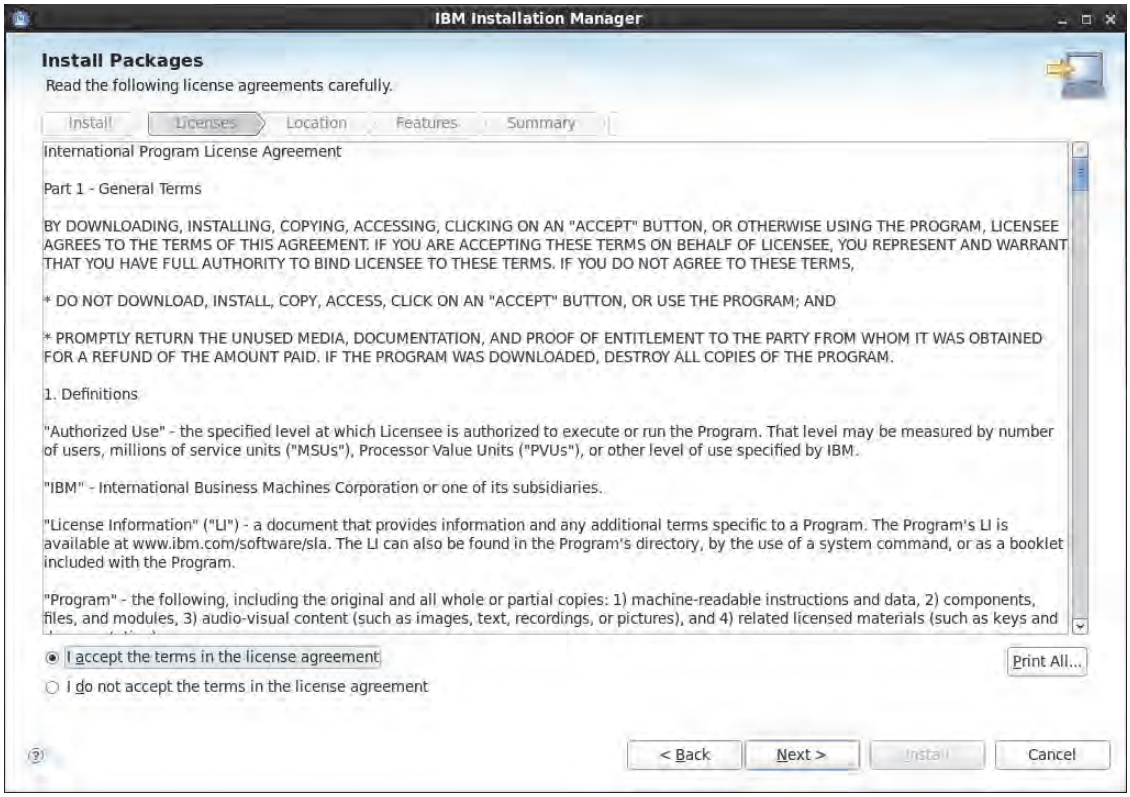
5. Click **OK** when upgrade is complete. Installation Manager restarts.



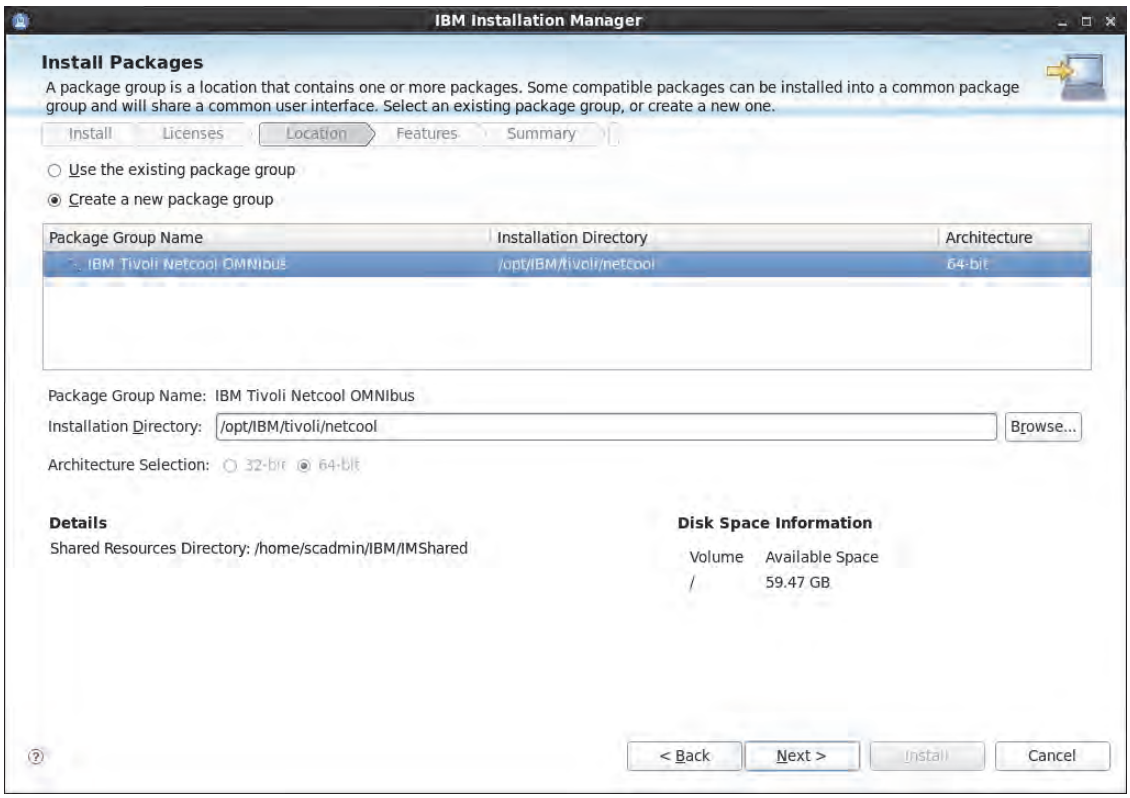
6. On the Install Packages window, click **Next**.



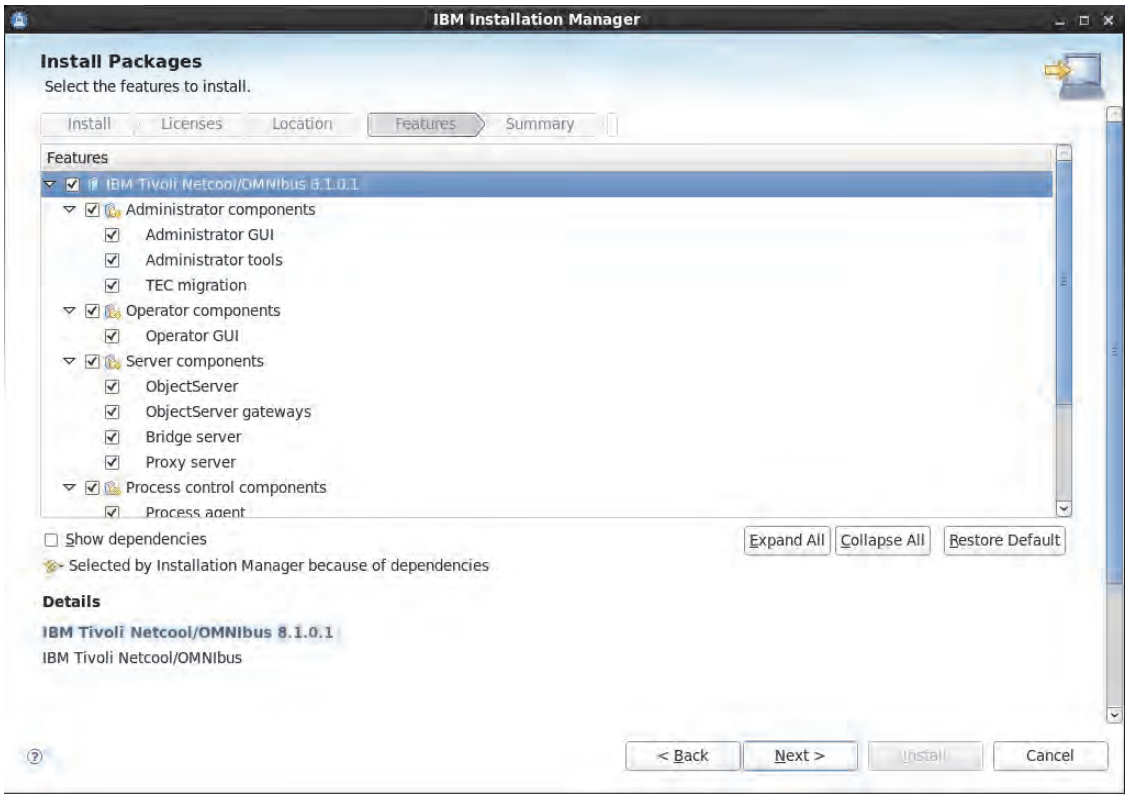
7. Accept the license agreement. Click **Next**.



8. Create a new package group. Click **Next**.

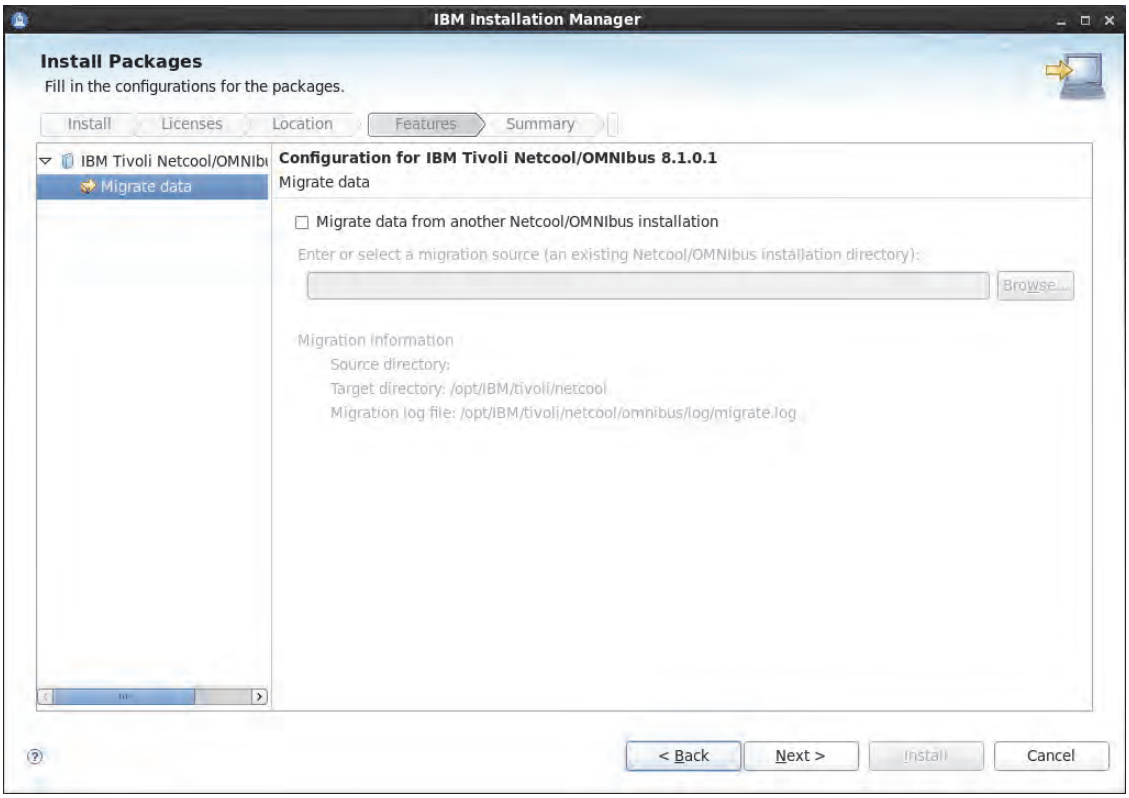


9. Keep all the default features. Click **Next**.

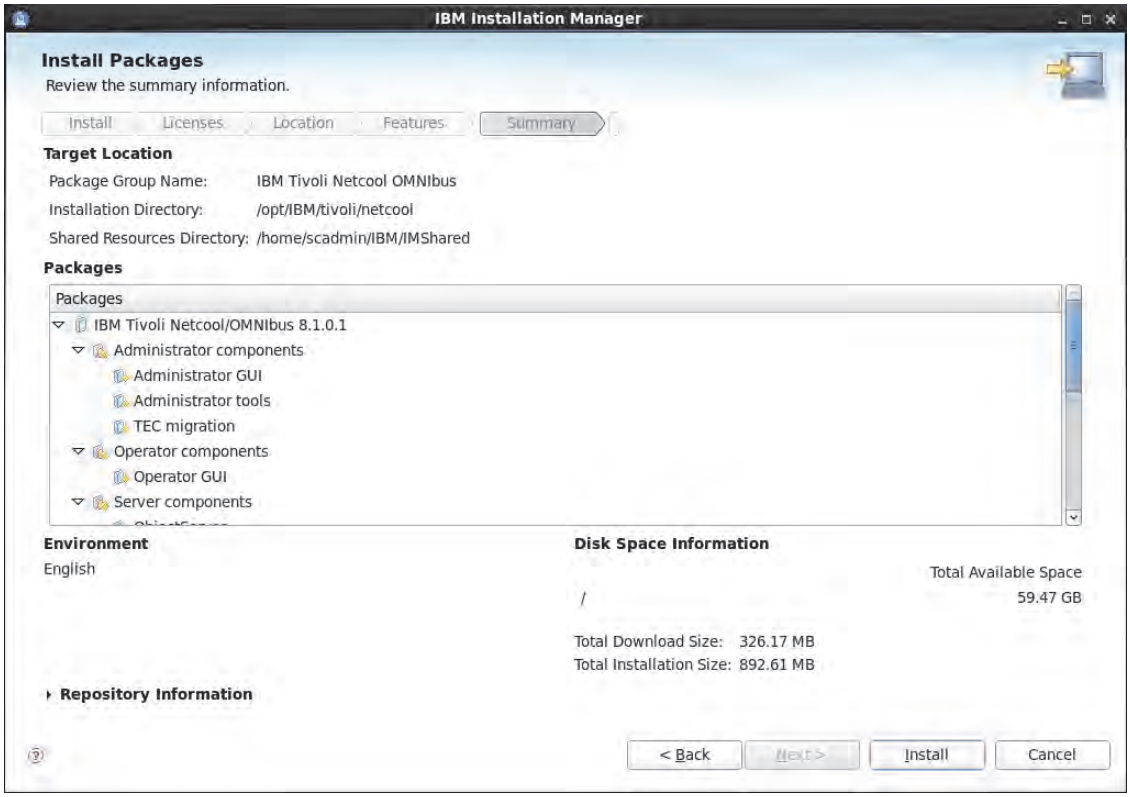


There is no need to migrate data.

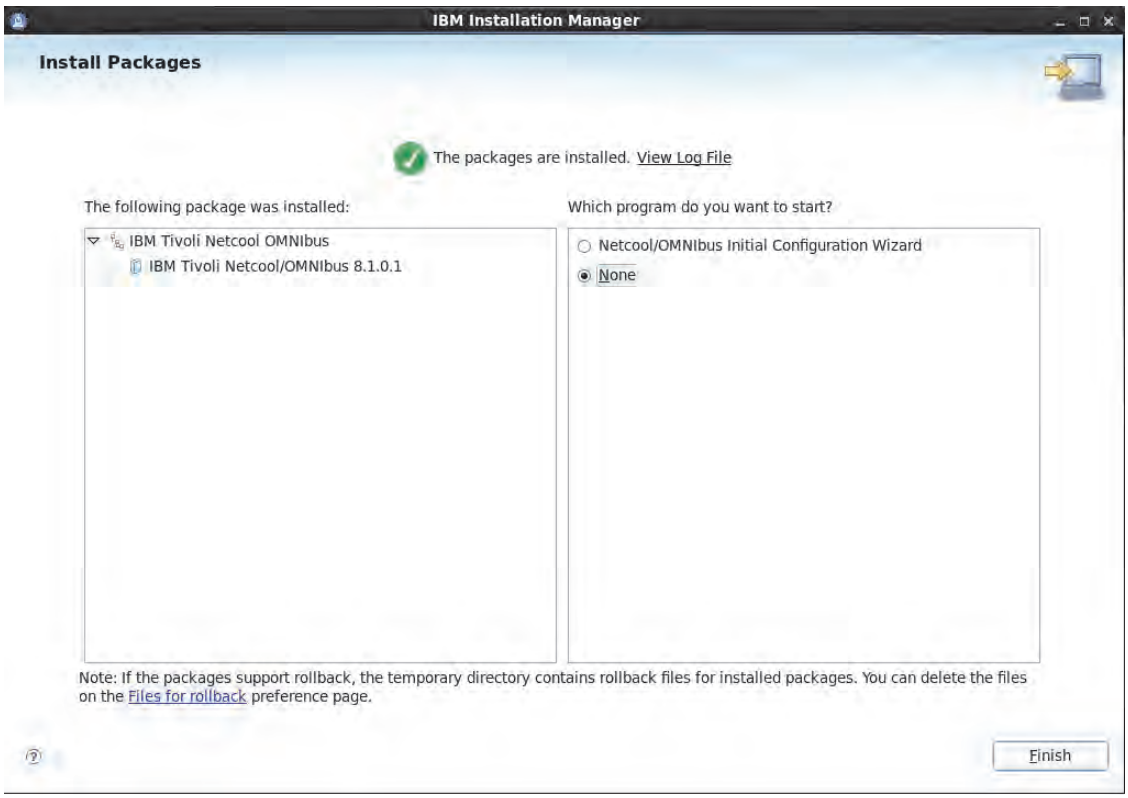
10. Click **Next**.



11. Click **Install**.



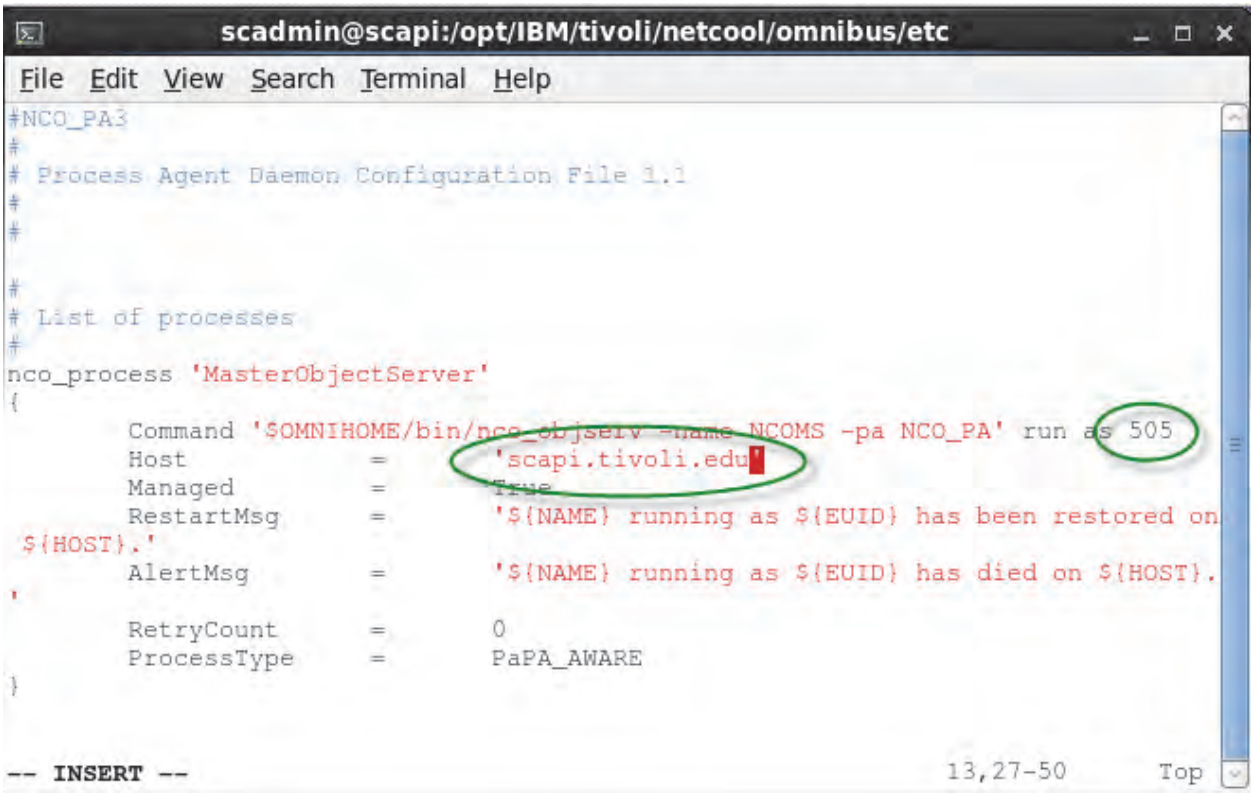
12. After installation, click **None** for which program you want to start. Click **Finish**.



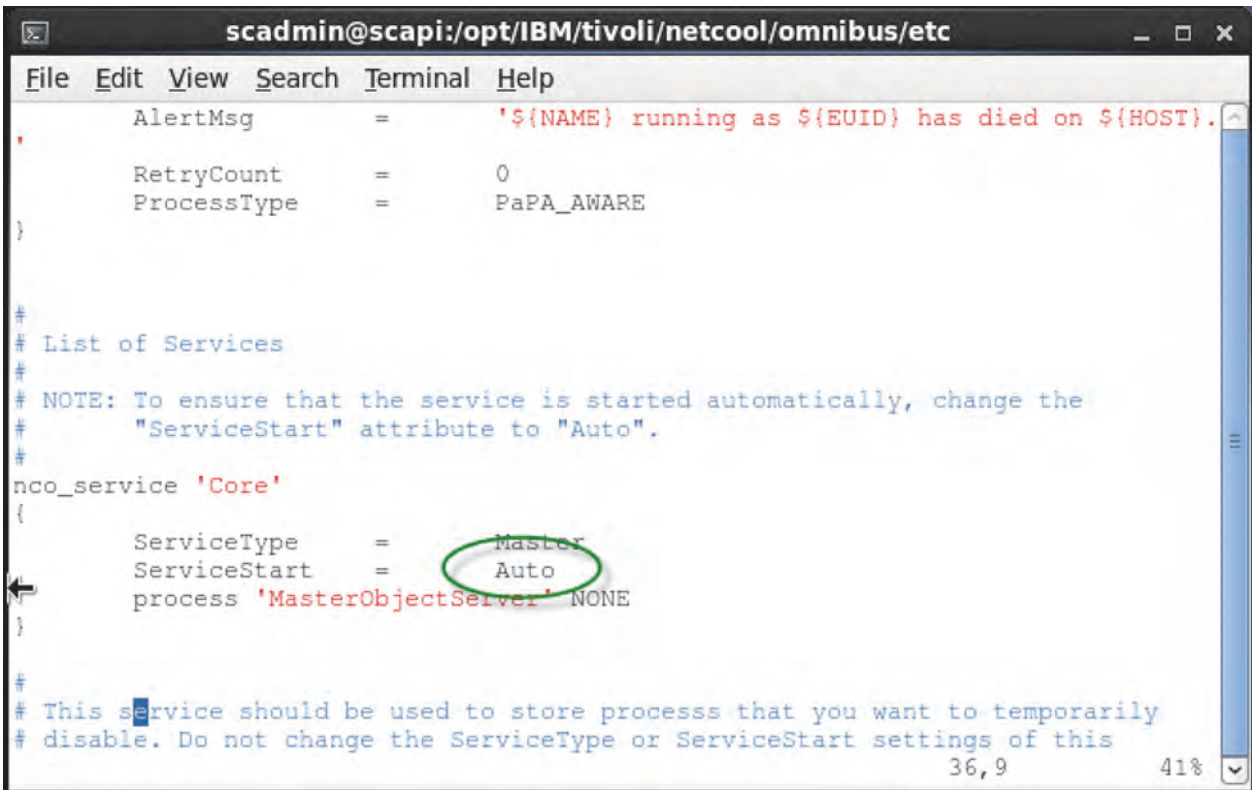
13. Open a terminal window as the **scadmin** user.
14. Change to the directory.
    ```
    cd /opt/IBM/tivoli/netcool/etc
    ```
15. Make a copy of the **omni.dat** file.
    ```
    mv omni.dat omni.dat.orig
    ```
16. Run the following command to update the omnihost server name with the host name of your system, such as scapi.tivoli.edu:
    ```
    sed s/omnihost/scapi.tivoli.edu/g omni.dat.orig > omni.dat
    ```
17. Change to the directory.
    ```
    cd /opt/IBM/tivoli/netcool/bin
    ```
18. Run the command to populate the interface files.
    ```
    ./nco_igen -out /opt/IBM/tivoli/netcool/etc/interfaces.linux2x86
    ```
19. Change to the directory.
    ```
    cd /opt/IBM/tivoli/netcool/etc
    ```
20. Ensure that both interface files are populated. If not, copy interfaces.linux2x86 to interfaces.
    ```
    cp interfaces.linux2x86 interfaces
    ```
21. Change to the directory.
    ```
    cd /opt/IBM/tivoli/netcool/omnibus/bin
    ```
22. Run the following command to create the database:
    ```
    ./nco_dbinit -server NCOMS
    ```
23. Start the object server by running the following command:
    ```
    ./nco_objserv -name NCOMS &
    ```
24. Confirm that the ObjectServer is running by using the following commands:
    ```
    ps -ef | grep nco
    ```
    ```
    [scadmin@scapi bin]$ ps -ef | grep nco
    scadmin  22523  4610  2 15:33 pts/0    00:00:00 /opt/IBM/tivoli/netcool/omnibus/platform/li
    nux2x86/bin64/nco_objserv -name NCOMS
    scadmin  22587  4610  0 15:34 pts/0    00:00:00 grep nco
    [scadmin@scapi bin]$ ^C
    [scadmin@scapi bin]$
    ```
25. Set a password on the **OMNIbus** root user.
    ```
    ./nco_sql  -server NCOMS -user root -password ''
    1> alter user 'root' set password 'object00';
    2> go
    (0 rows affected)
    exit
    ```

Setting up OMNIbus to start automatically after restart

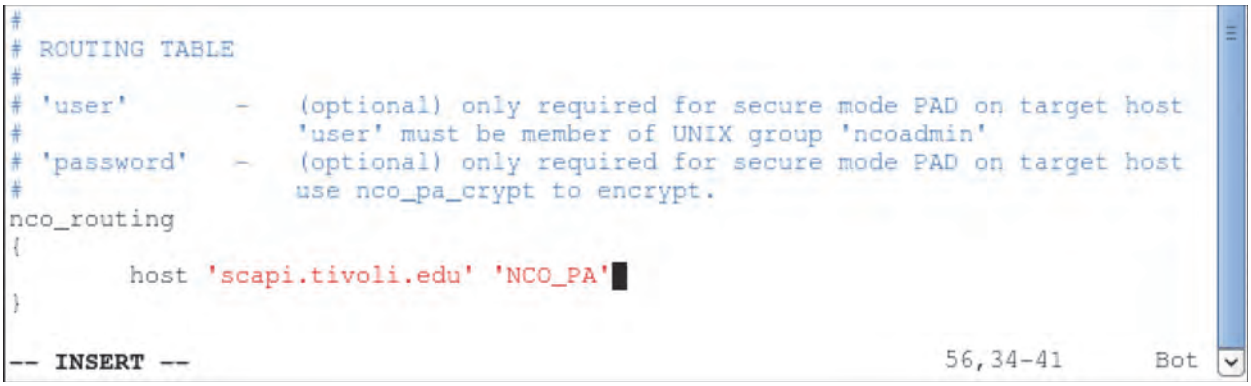
1. Determine the user ID for the **scadmin** user.
   `id -u scadmin`
2. Configure the Process Activity daemon by going to the **/opt/IBM/tivoli/omnibus/etc** directory.
   `cd /opt/IBM/tivoli/netcool/omnibus/etc`
3. Create a copy of the **nco_pa.conf**.
   `cp nco_pa.conf nco_pa.conf-orig`
4. Edit the **nco_pa.conf** file.
   `vi nco_pa.conf`
5. Change the user ID and host name for the nco_process MasterObjectServer. Use the **scadmin** user ID number and the FQDN for the server.



6. Change ServiceStart in the nco_service Core to **Auto**.



7. Change the host name and remove the user name an password from the nco_routing section.

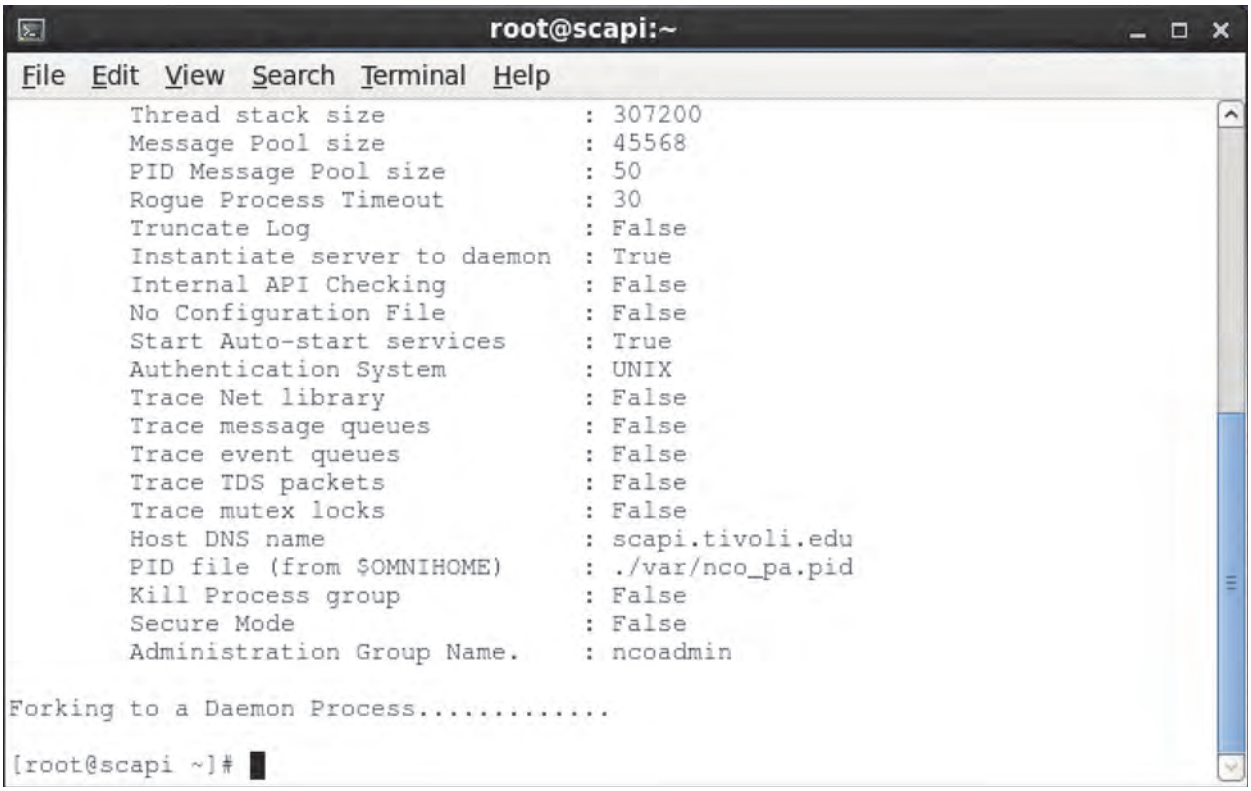


8. Save the file.

9. Change to the **root** user with password **object00**.

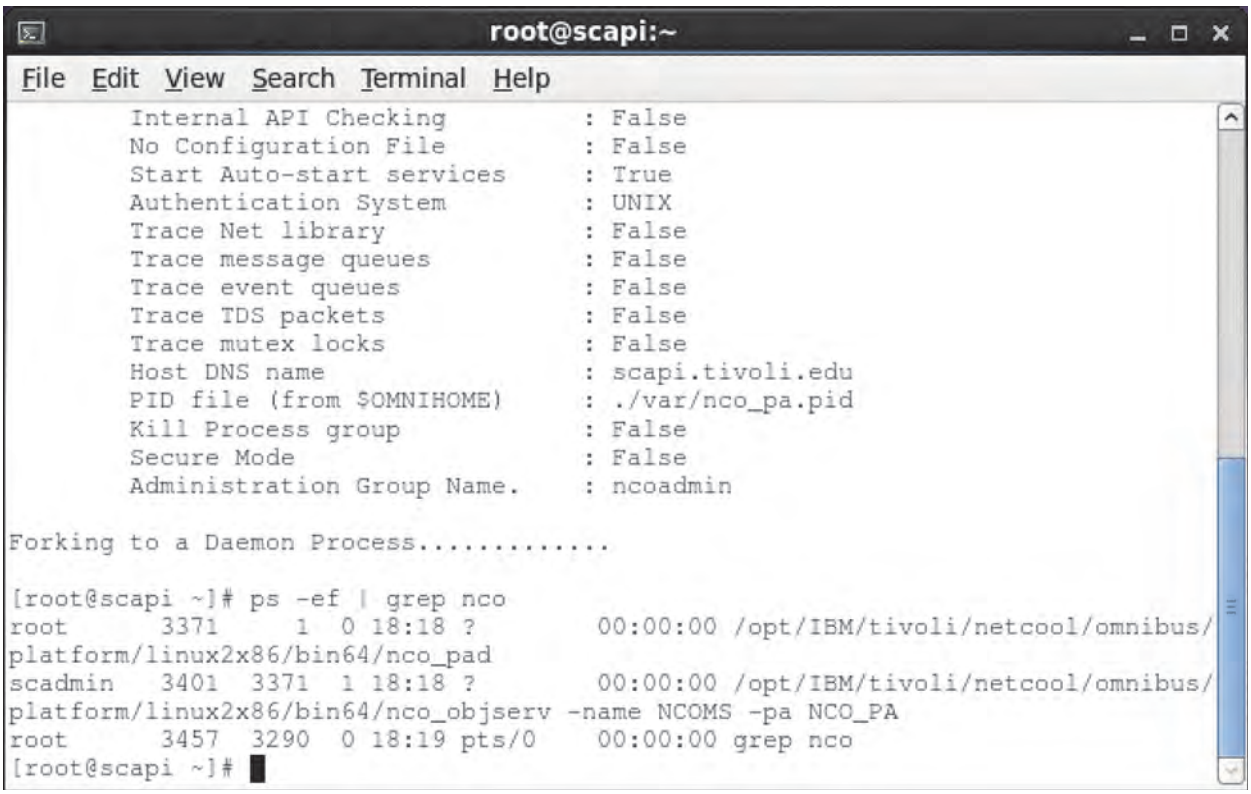
10. Start the Process Activity Daemon to install NCO_PA in nonsecure mode.

    `/opt/IBM/tivoli/netcool/omnibus/bin/nco_pad`



11. Check to see if the object server was started.

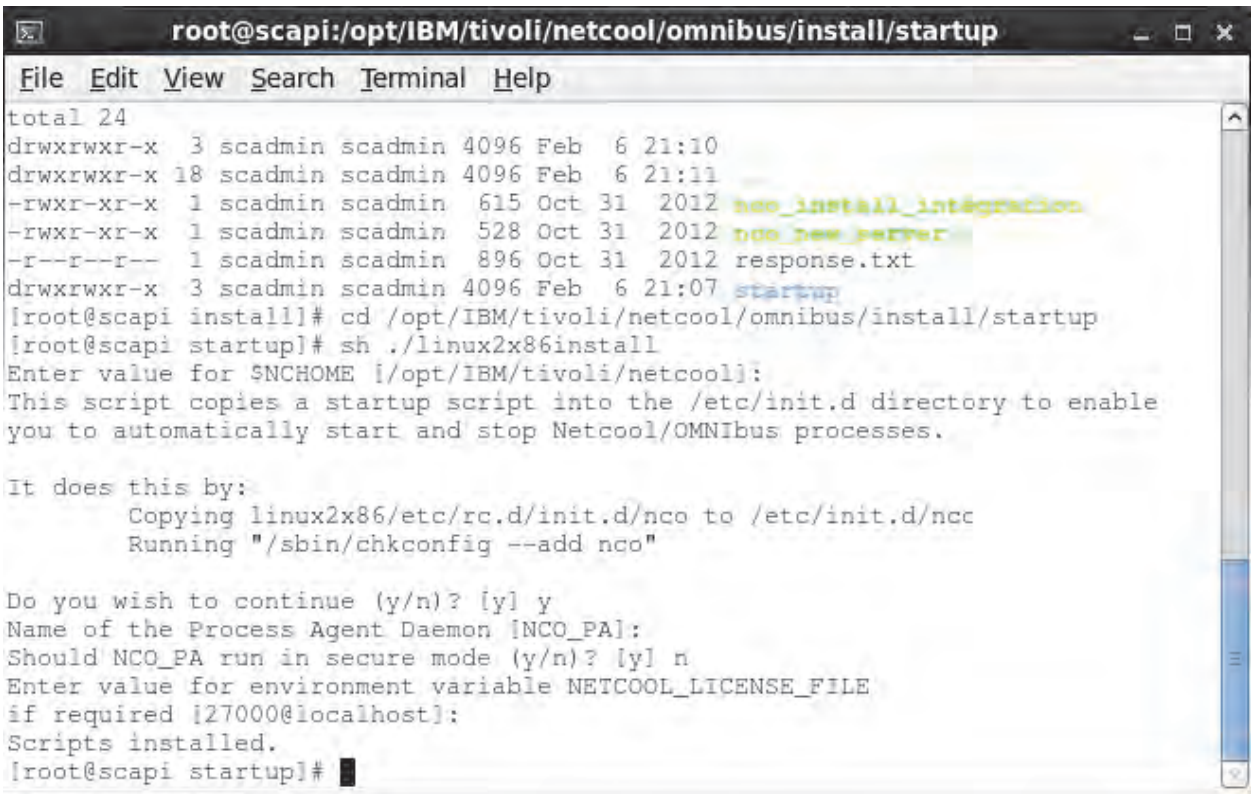
    `ps –ef | grep nco`



12. Install the system auto-start script as **root**.

```
cd /opt/IBM/tivoli/netcool/omnibus/install/startup
sh ./linux2x86install
```

13. Answer the questions. Answer **No** to running NCO_PA in secure mode.

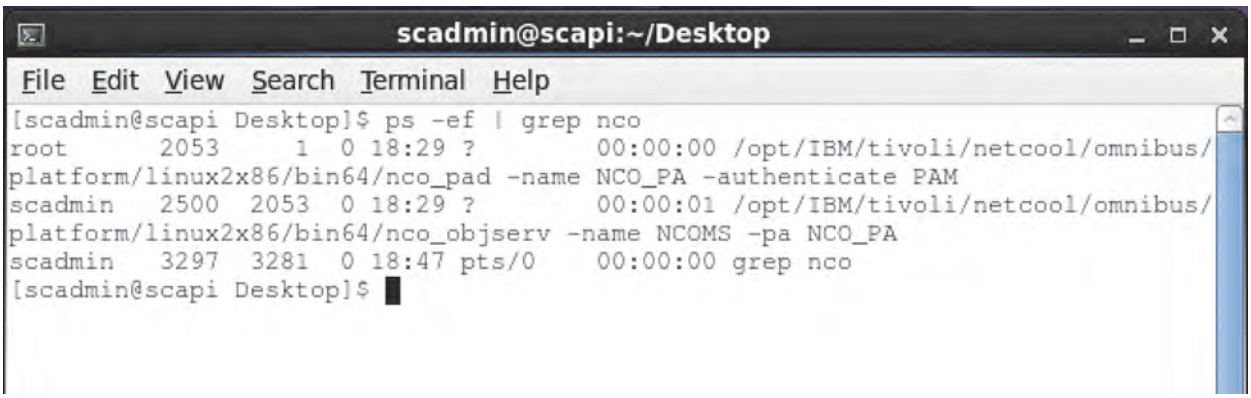


14. Restart server to check whether OMNIbus is started.

15. Log in as **scadmin** with password **object00**.

16. Open terminal and execute the following command:

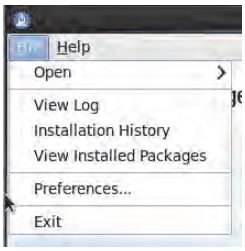
```
ps –ef | grep nco
```



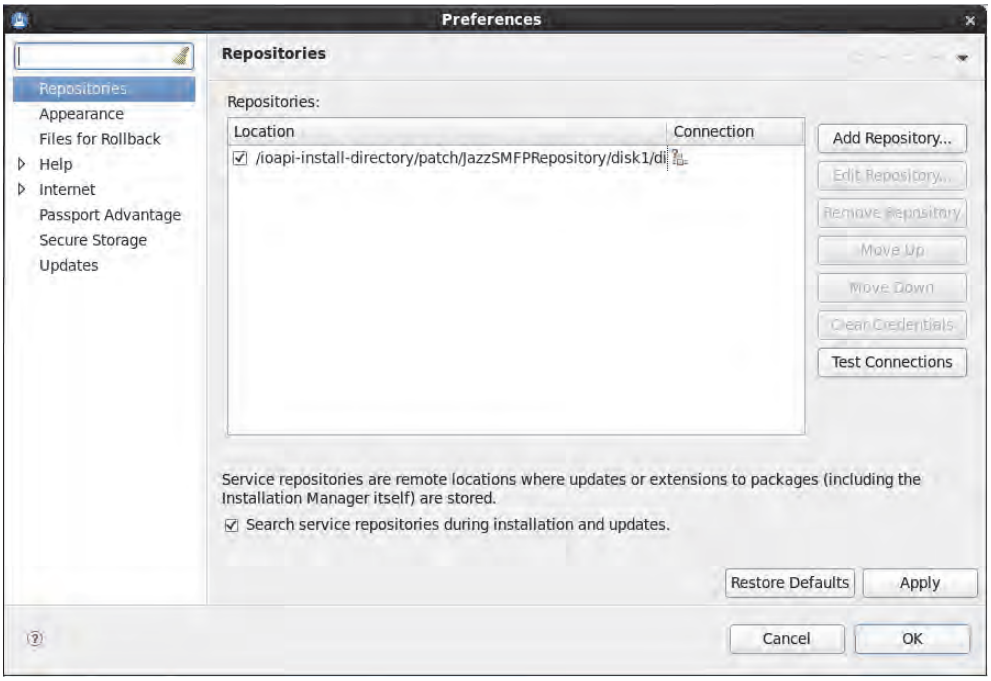
Appendix D Installing Netcool/OMNIbus Web GUI

1. Move the installation file, **OMNIbus-v8.1-WebGUI.linux64.zip**, to an installation directory on the server and extract it.
2. Start the Installation Manager with the following command:
   `/home/scadmin/IBM/InstallationManager/eclipse/launcher`

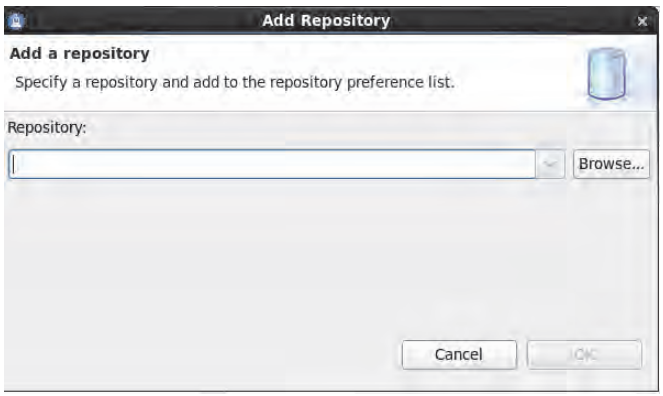
3. Select **File > Preferences**.



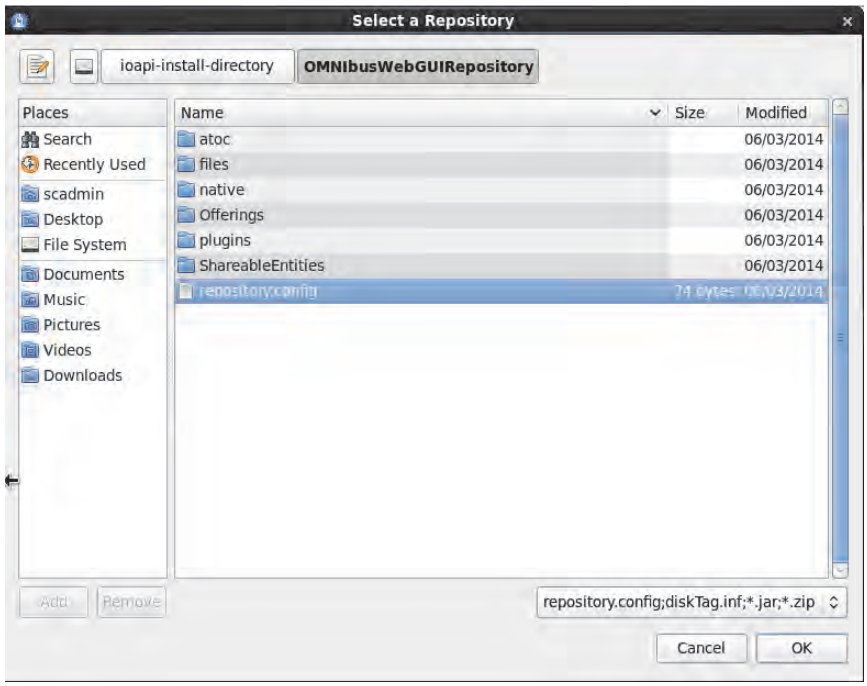
4. Click **Add Repository**.



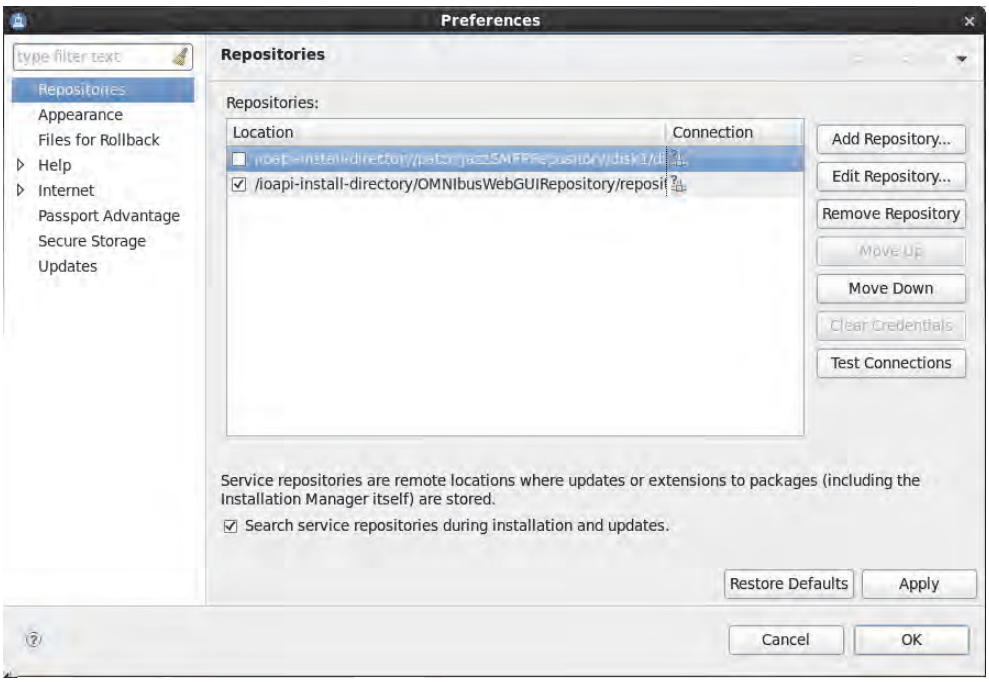
5. Click **Browse** to point the installer to the correct repository.



6. Navigate to the directory where you uncompressed the files. Select **OMNIbusWebGUIRepository** and click **repository.config**. Click **OK**.

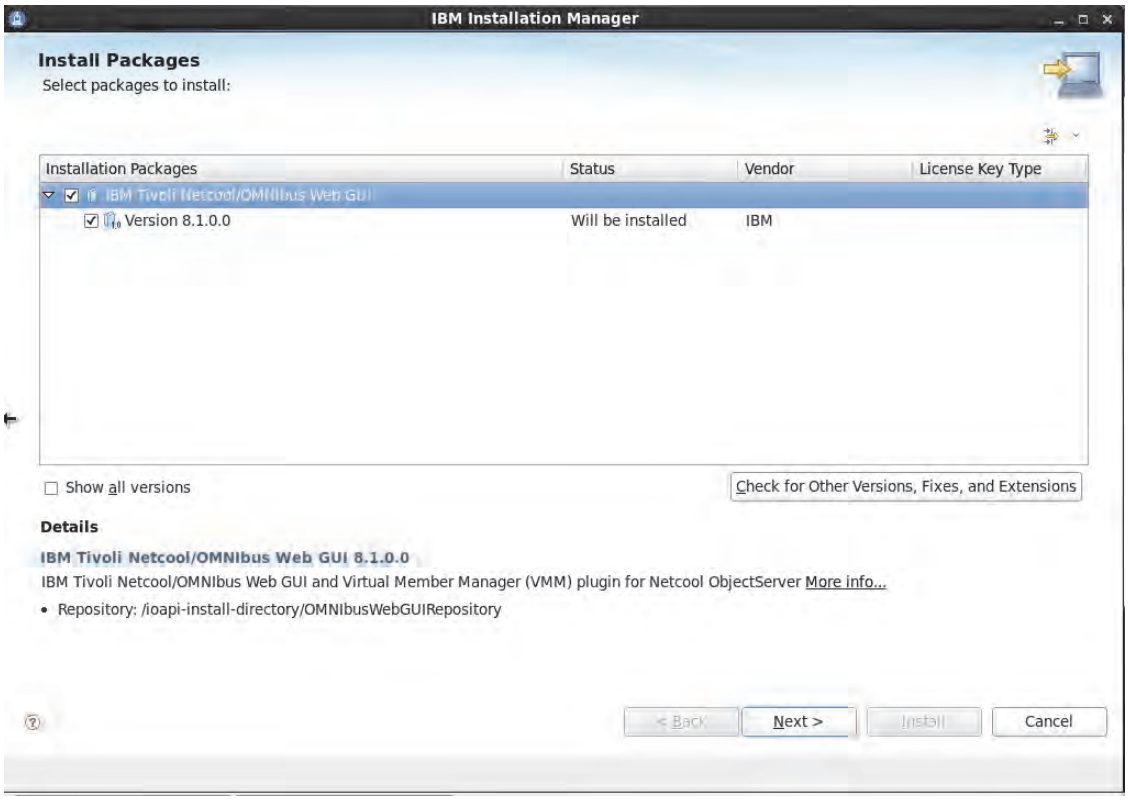


7. Clear any other repositories that are defined in the installer, for example, JazzSMFPRepository. Click **OK**.

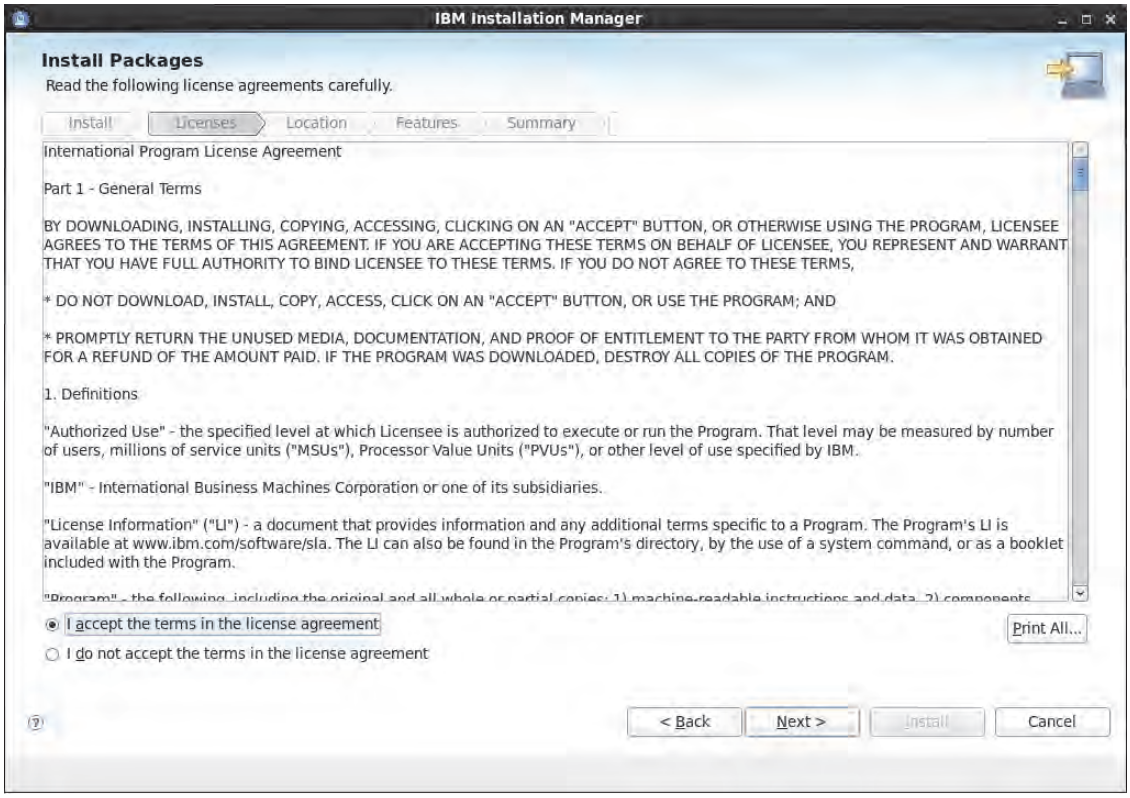


8. Click Install.

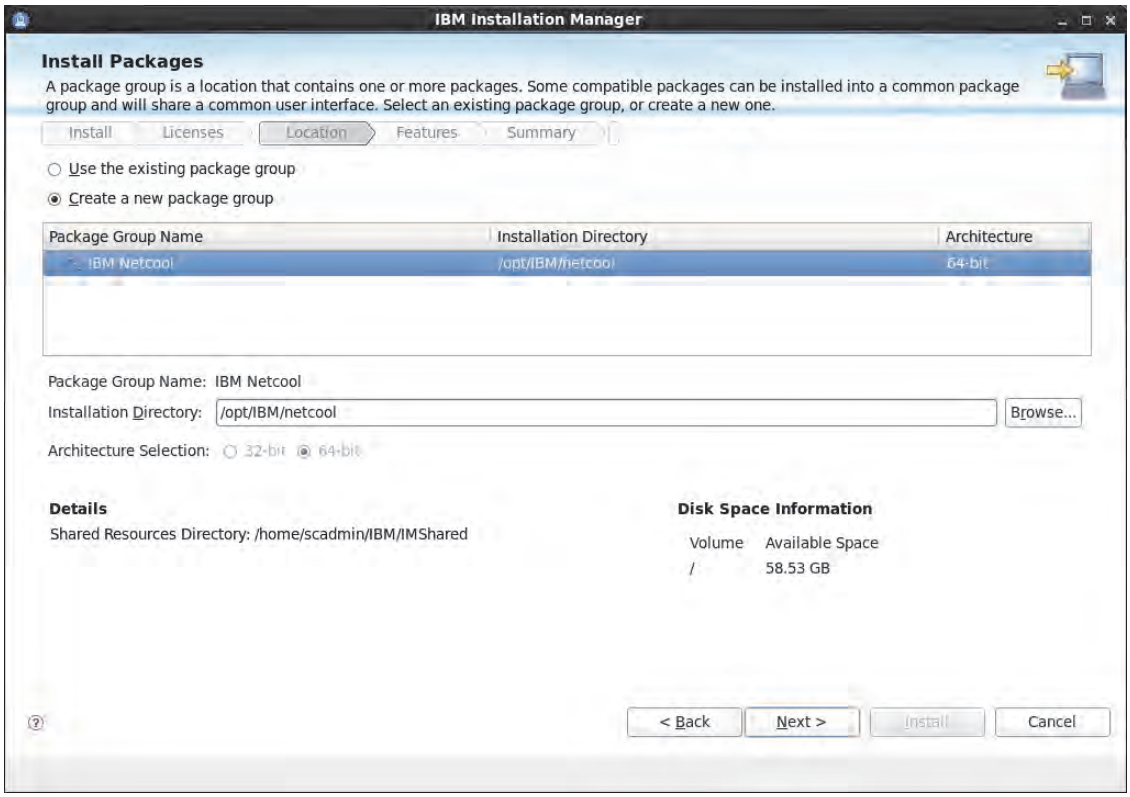
9. Ensure that the Web GUI components are checked. Click **Next**.



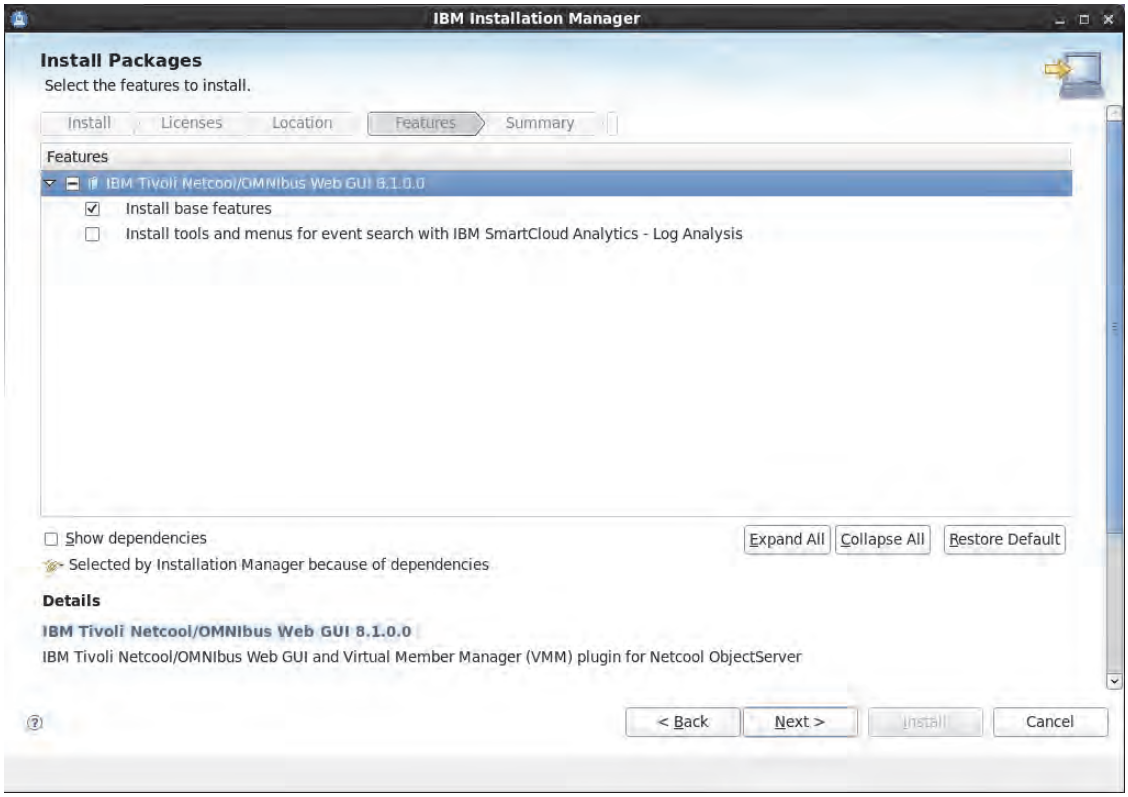
10. Accept the license agreement. Click **Next**.



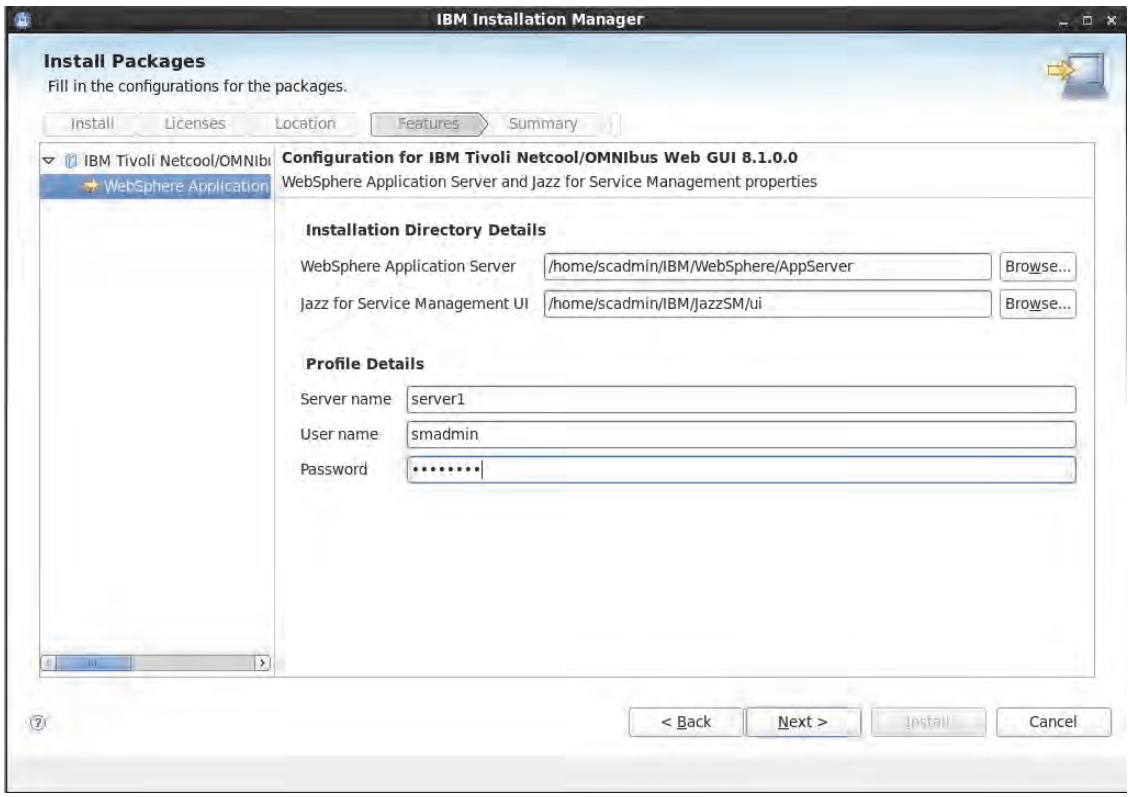
11. Keep the default installation location for the software. Click Next.



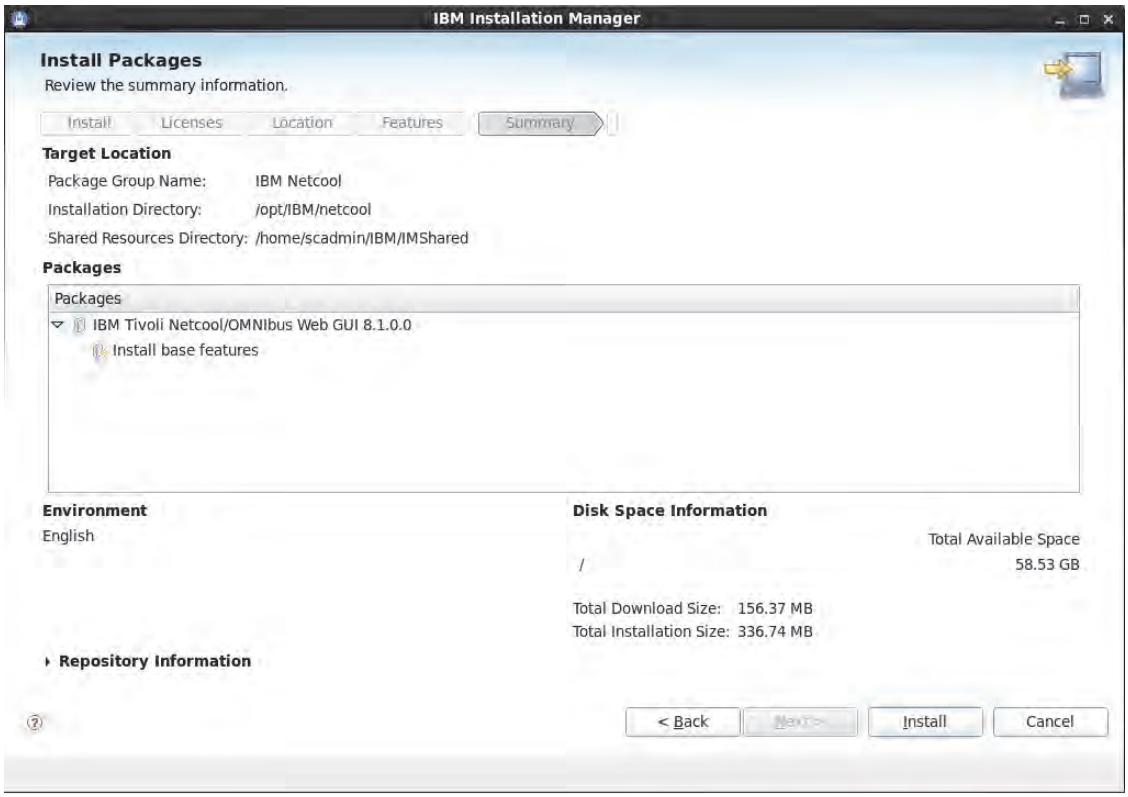
12. Keep the default to install the base features. Click **Next**.



13. Add the Jazz profile information by providing the password (**object00**) for the **smadmin** profile.



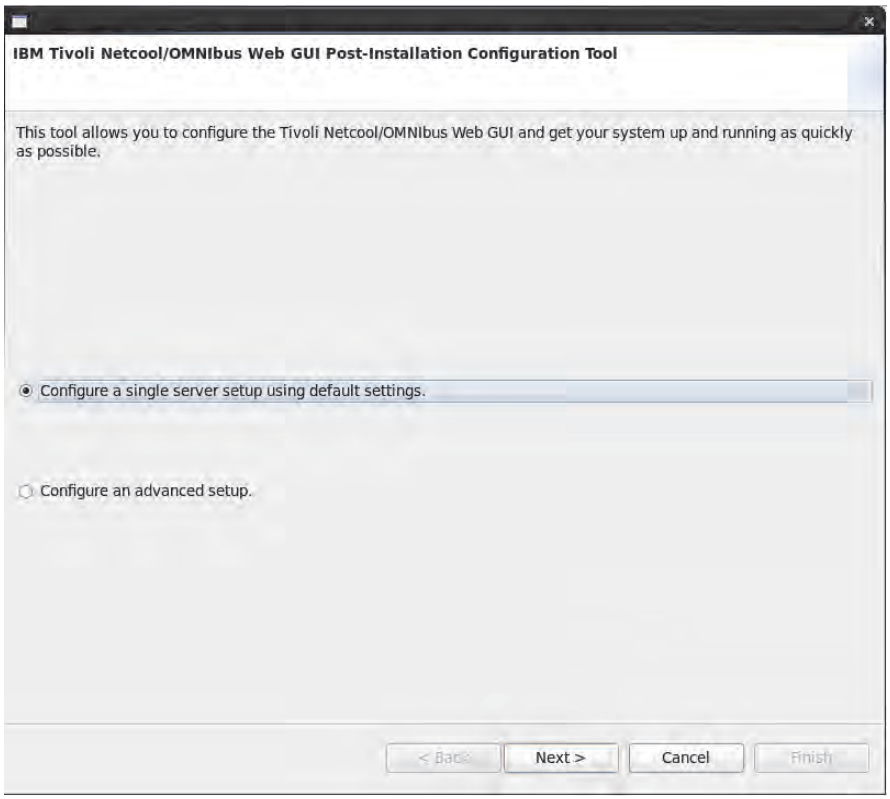
14. Click Install.



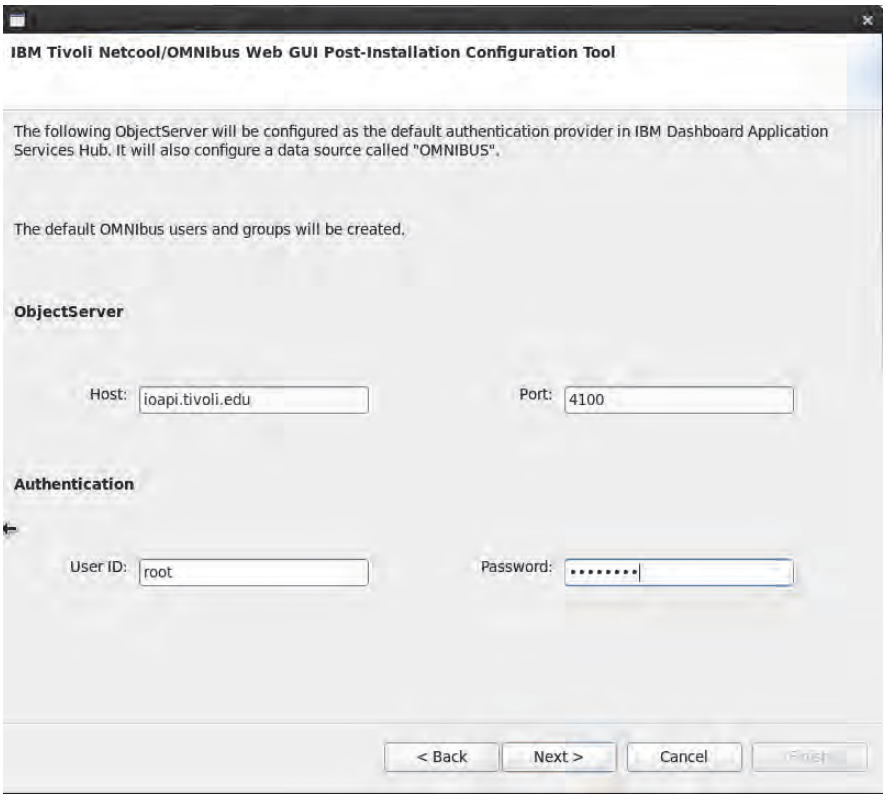
15. When the installation is complete, verify that **Configure IBM Tivoli Netcool/OMNIbus Web GUI** is selected. Click **Finish**.

The configuration wizard opens.

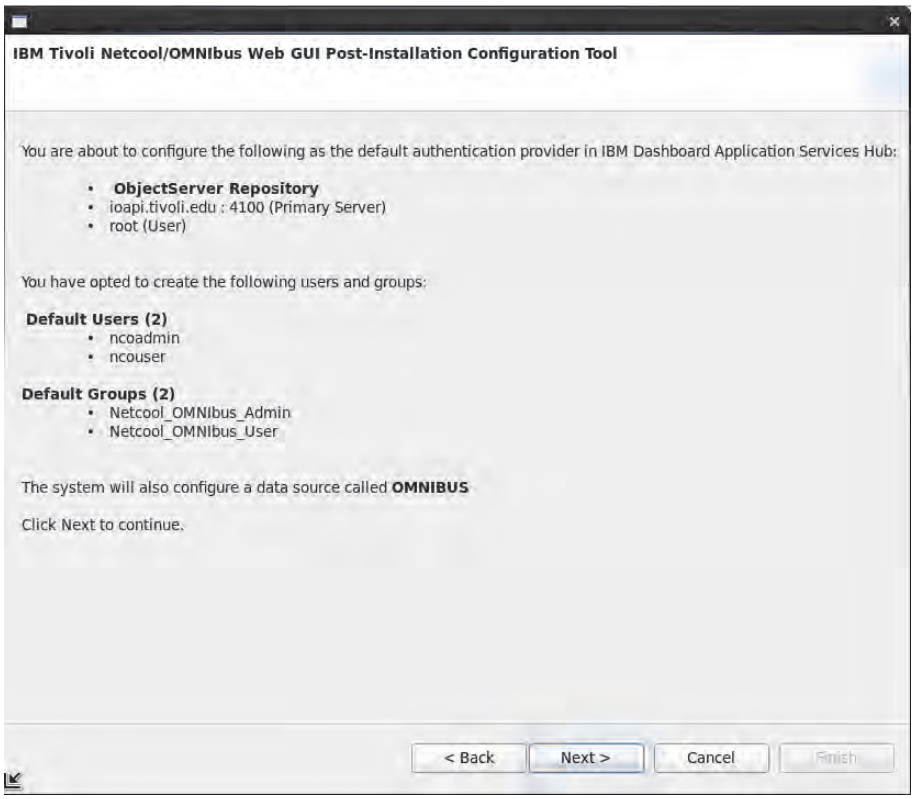
16. Ensure that **Configure a single server setup using default settings** is selected. Click **Next**.



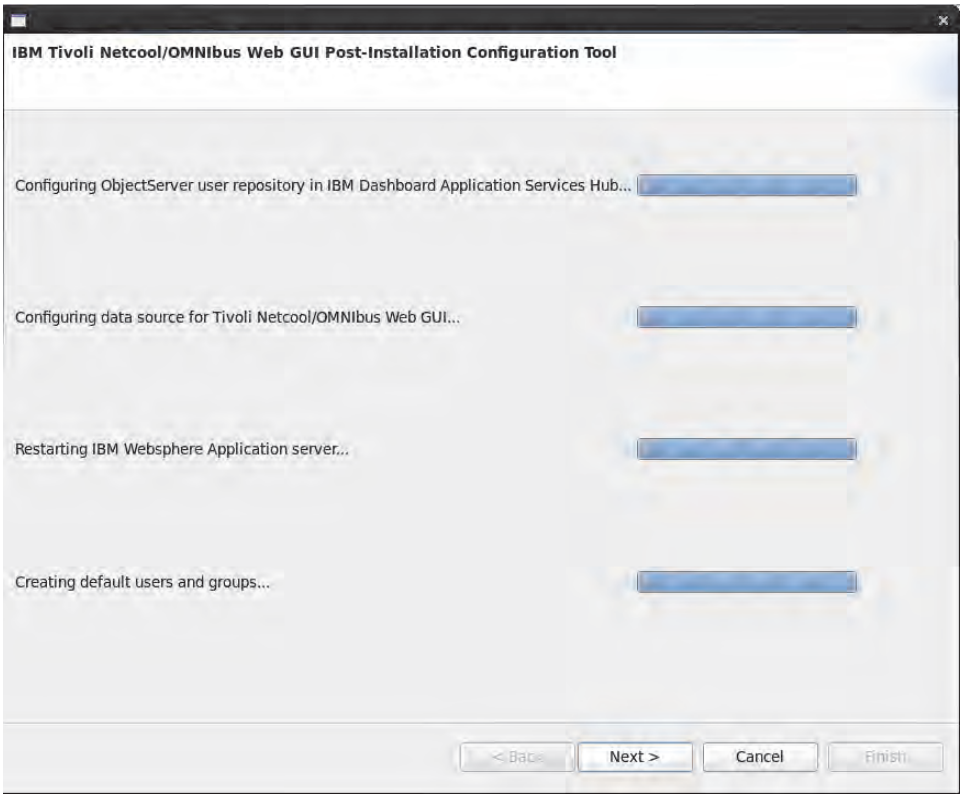
17. Enter the host name of the OMNIbus server (ioapi.tivoli.edu) and the OMNIbus **root** password (**object00**). Do not change the OMNIbus port. Click **Next**.



18. Review summary. Click **Next**.

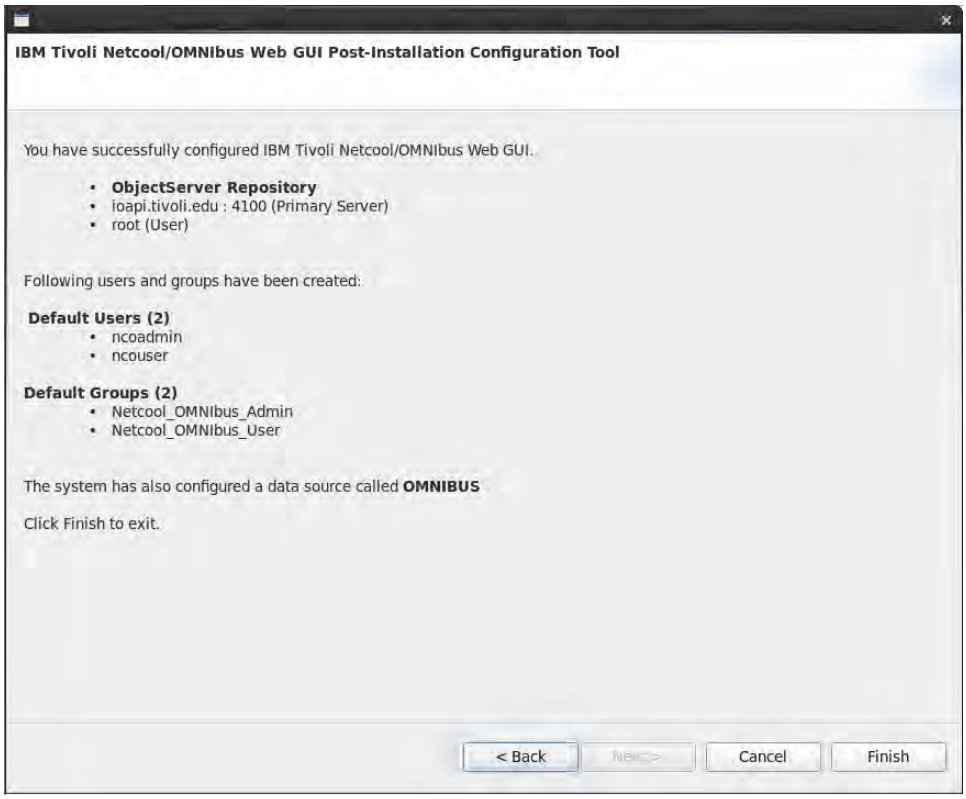


The wizard configures Web GUI and Jazz for Service Management.



Installer completes.

19. Click **Finish**.

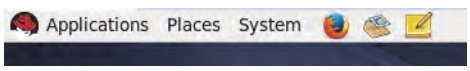


20. For the Active Event List to work, you must make a link for Firefox to reference the appropriate plug-ins. As **root**, enter the following command in a terminal window:

    `ln -s /opt/ibm/db2/V10.5/java/jdk64/jre/lib/amd64/libnpjp2.so /usr/lib64/mozilla/plugins`

Confirming installation

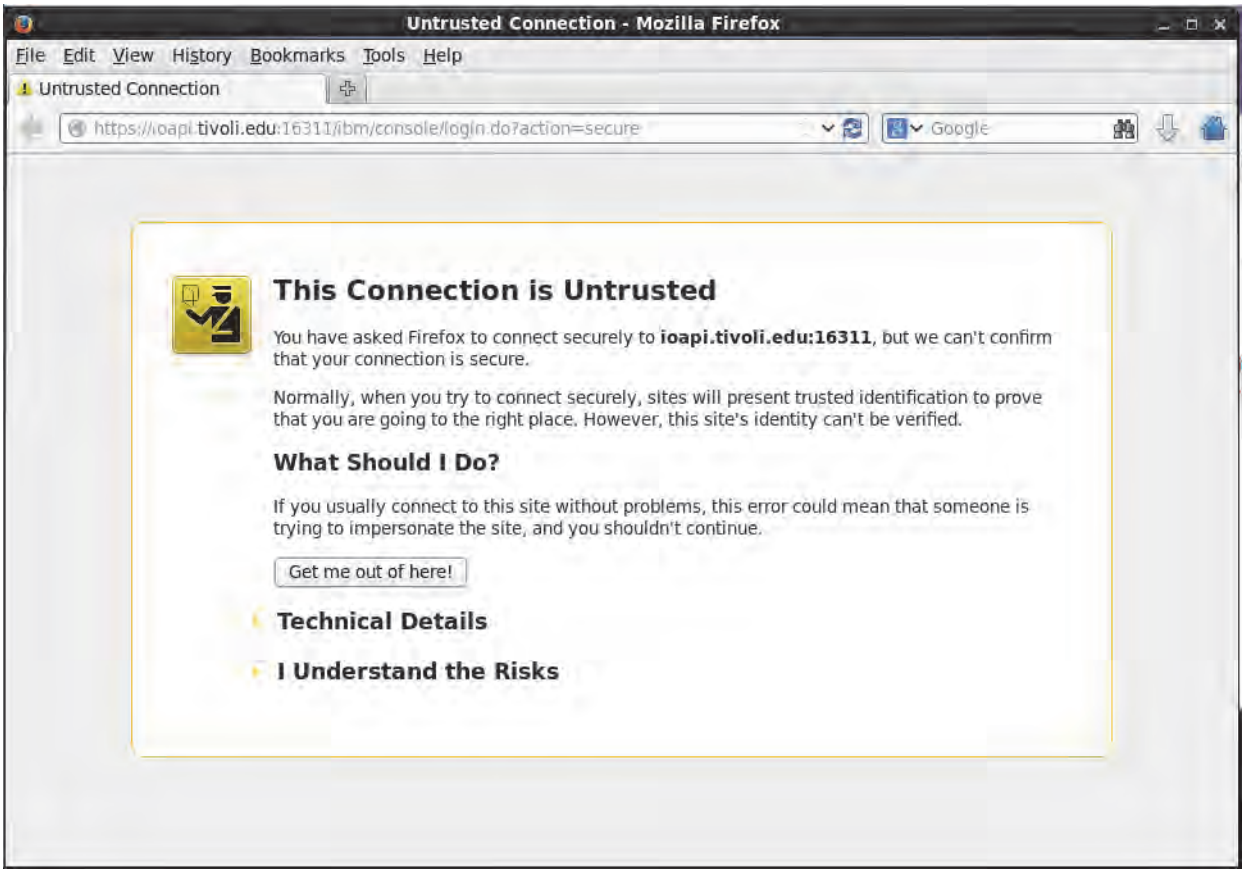
1. Start Firefox by clicking the icon on the top menu bar.



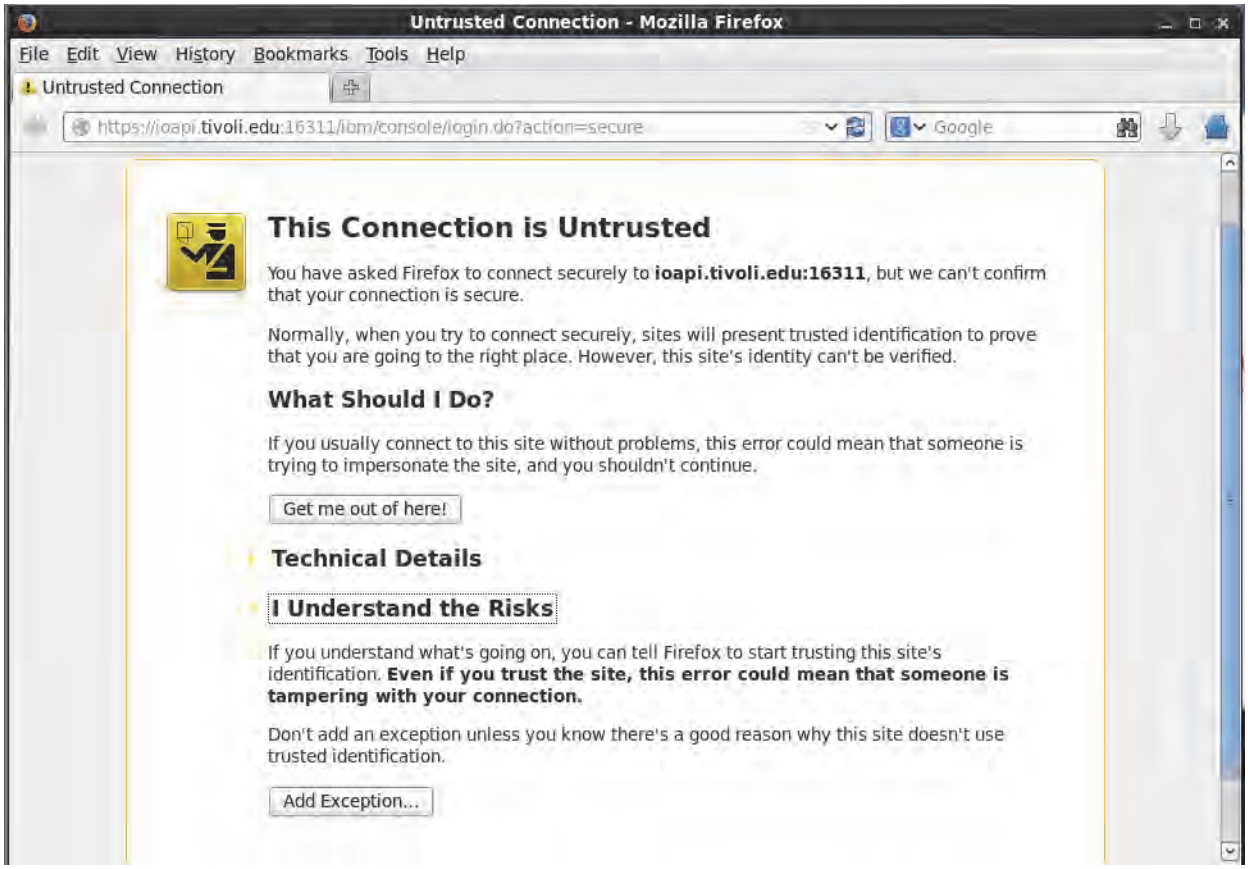
2. Enter the following URL:

   http://scapi.tivoli.edu:16310/ibm/console

3. Accept the untrusted website by selecting **I Understand the Risks**.

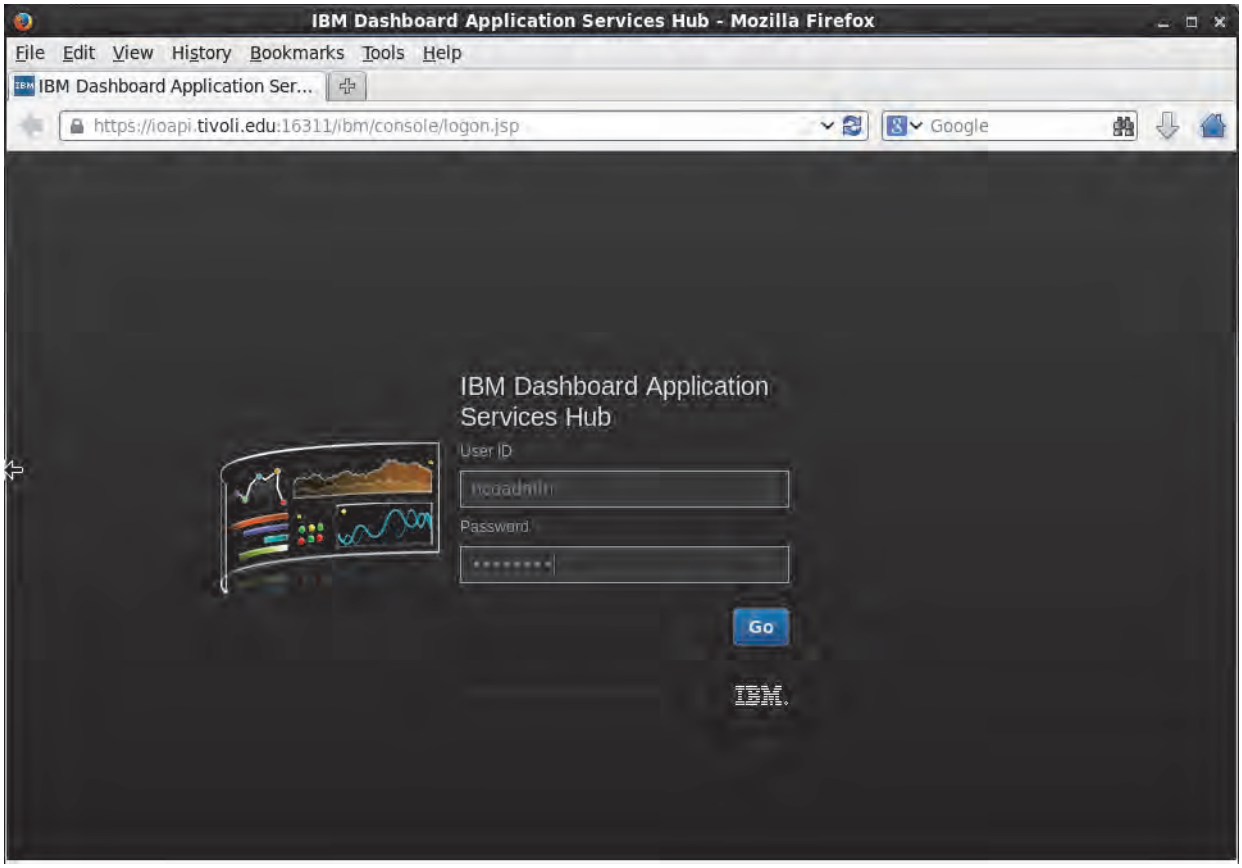


4. Click **Add Exception**.

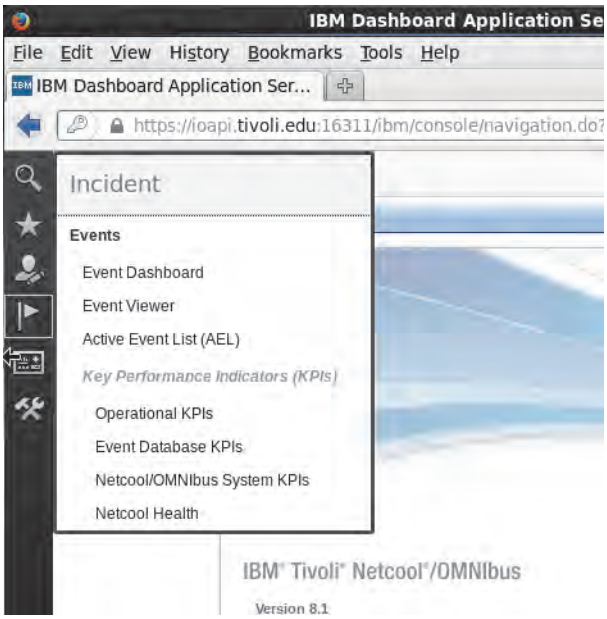


5. Click **Confirm Security Exception**.

6. Log in with the user name **ncoadmin** and password **object00**.

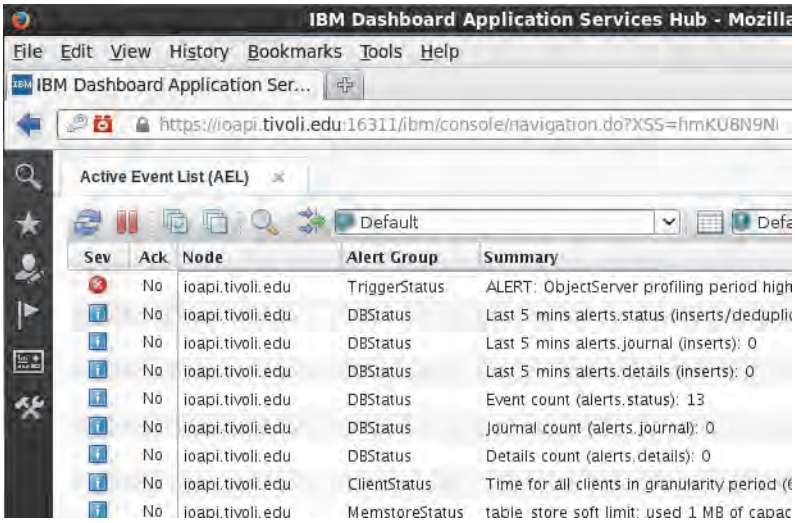


7. Select the flag icon and click **Active Event List** menu.



8. Accept the security exception using the Java plug-in.

9. Ensure that you see events in list.



Setting up Jazz for Service Management to start automatically

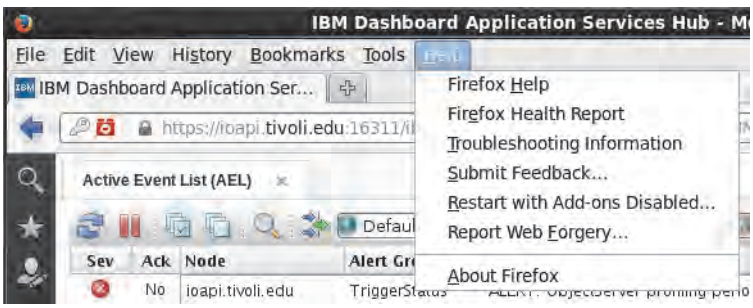
As **root**, add the following line to **/etc/rc.local**:

```
su scadmin -c '/home/scadmin/IBM/JazzSM/profile/bin/startServer.sh server1'
```

Downgrading Firefox if necessary

Predictive Insights is federated into DASH, which requires accepting an exception when trying to connect to the server from DASH. You need an older version of Firefox for this procedure to work correctly.

1. Open Firefox.
2. Select **Help > About Firefox.**



3. Confirm that the Firefox version is 24.x.

4. If the version is higher, you must uninstall Firefox and install a older version. This might require the recreation of the symbolic link that you made earlier.

Appendix E Logstash configuration files

These are the configuration files you created during this lab. You can reference these for future logstash development efforts.

mixed-skinny-data.conf

```
input {
   stdin {
   }
}
filter {
   mutate {
      add_tag => "this is a test"
   }
   if [message] =~ /.*<error>.*/ or [message] =~ /.*<message>.*/ {
      mutate {
        add_tag => "UNUSEFUL"
        }
   }
   if "UNUSEFUL" in [tags] {
      drop {}
   }
   csv {
       columns => ["timestamp","hostname","metric-name","metric-value"]
       add_tag => "added fields to CSV data"
   }
}
output {
   stdout {
      codec => rubydebug {}
   }
   csv {
      fields => ["timestamp","hostname","metric-name","metric-value"]
      path => "/home/scadmin/Desktop/skinny-metrics.csv"
   }
}
```

mixed-data.conf

```
input {
   stdin {
   }
}
filter {
   if [message] =~ /.*mpmstats: rdy.*/  {
      mutate {
             add_tag => "USEFUL"
             }
   }
   else {
        drop {}
   }
   grok {
        match => ["message",".*_%{DATA:server}/.*\.%{DATE:date}:\[%{WORD} %{WORD}
%{WORD} %{TIME:time}.*mpmstats: %{GREEDYDATA:metrics}"]
        add_field => {"timestamp" => "%{date} %{time}"}
   }
   grok {
        match => ["metrics","%{WORD} %{INT:rdy} %{WORD} %{INT:bsy} %{WORD}
%{INT:rd} %{WORD} %{INT:wr} %{WORD} %{INT:ka} %{WORD} %{INT:log} %{WORD} %{INT:dns}
%{WORD} %{INT:cls}"]
   }
}
output {
   stdout {
      codec => rubydebug
   }
   scacsv {
      fields =>["timestamp","server","rdy","bsy","rd","wr","ka","log","dns","cls"]
      path => "/home/scadmin/Desktop/server.csv"
      group => "serverData"
      file_interval_width => "HOUR"
      time_field => "timestamp"
      time_field_format => "dd-MM-yyyy HH:mm:ss"
      timestamp_output_format => "yyyy-MM-dd_HH-mm-ss"
   }
}
```

extract-hosts.conf

```
input {
   file {
      path => "/opt/scapi-training-data/logstash-examples/report*.csv"
      start_position => "beginning"
   }
}

filter {
   if [message]  =~ /^(?=.*Frequency)(?=.*Server)/ {
      grok {
      match => ["message","%{DATA} Server: %{DATA:FQDNServer},"]
      }
      grok {
      match => ["path","%{GREEDYDATA}/%{GREEDYDATA:filename}.csv"]
      }
      }
   else {
      drop {}
   }
}

output {
   stdout {
      codec => rubydebug {}
   }
   file {
      path => "/home/scadmin/Desktop/serverName.map"
      message_format => "%{filename} : %{FQDNServer}"
      flush_interval => 0
   }
}
```

merge-reports.conf

```
input {
   file {
      path => "/opt/scapi-training-data/logstash-examples/*.csv"
      start_position => "beginning"
   }
}

filter {
   if [message] =~ /.*-2014.*/ {
      csv {
      columns =>
["date","timezone","time","processor","process-percent","memory","net-stat","netwo
rk"]
      add_field => {"timestamp" => "%{date} %{time}"}
      add_tag => "added fields to CSV data"
      }
      grok {
      match => ["path","%{GREEDYDATA}/%{GREEDYDATA:filename}.csv"]
      }
      translate {
      dictionary_path => "/home/scadmin/Desktop/serverName.map"
      field => "filename"
      destination => "FQDNServer"
      add_tag => "figured out server name"
      }
      mutate {
        gsub => ["process-percent", "%",""]
        gsub => ["net-stat", "-",""]
        add_tag => "removed % and -"
      }
   }
   else {
      drop {}
   }
}

output {
   stdout {
      codec => rubydebug {}
```

```
   }
   scacsv {
      fields =>
["timestamp","FQDNServer","processor","process-percent","memory","net-stat","netwo
rk"]
      path => "/home/scadmin/Desktop/servertemp.csv"
      group => "serverPerformance"
      time_field[1] => "timestamp"
      time_field_format => "MM-dd-yyyy HH:mm:ss"
      timestamp_output_format => "MM-dd-yyyy_HH-mm-ss"
   }
}
```

1

database-extract.conf

```
input {
   genjdbc {
      jdbcHost       => 'scapi.tivoli.edu'
      jdbcPort       => '50000'
      jdbcTargetDB   => 'db2'
      jdbcDBName     => 'MONITOR'
      jdbcUser       => 'db2inst1'
      jdbcPassword   => 'object00'
      jdbcDriverPath => '/opt/ibm/db2/V10.5/java/db2jcc4.jar'
      jdbcSQLQuery   => 'select INTERFACE_TRAFFIC.TIME, NODES.RESOURCENAME,
INTERFACES.INTERFACENAME, INTERFACE_TRAFFIC.IN_TOTAL_BYTES, from NODES,
INTERFACES, INTERFACE_TRAFFIC where INTERFACE_TRAFFIC.RESOURCEID=NODES.RESOURCEID
AND INTERFACE_TRAFFIC.INTERFACEID=INTERFACES.INTERFACEID'
        jdbcTimeField  => 'INTERFACE_TRAFFIC.TIME'
        jdbcCollectionStartTime => '2015-05-28__00:00:00'
        }
}

filter {
}

output {
   stdout {
      codec => rubydebug {}
   }
   scacsv {
      fields =>
["TIME","RESOURCENAME","INTERFACENAME","IN_TOTAL_BYTES","OUT_TOTAL_BYTES"]
      path => "/home/scadmin/Desktop/data.csv"
      group => "databaseData"
      time_field => "TIME"
      time_field_format => "yyyy-MM-dd__HH:mm:ss"
      timestamp_output_format => "yyyy-MM-dd_HH-mm-ss"
   }
}
```

IBM Training