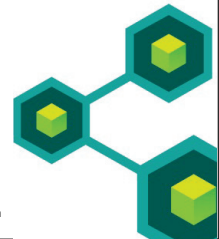




Explore Component Logging

IBM Cognos BI 10.2.2



Business Analytics software

© 2015 IBM Corporation

Objectives

- At the end of this module, you should be able to:
 - explore component logging for Gateway, Dispatcher, Report Server, and Universal Data Access layer



Report Server (RSVP) Trace

- for RSVP errors, do RSVP trace
 - trace SOAP requests sent to RSVP
 - trace data sent and received over BIBusTKServerMain socket
 - command and response documents associated with all Query Framework requests generated by RSVP
- enable recordings
- trace print commands and statuses

© 2015 IBM Corporation



This is for your information. You would only use this if you were asked to do so by customer support.

Traces are usually done in combination with other traces, and interpreting only one trace file is merely one part of the whole story. When troubleshooting, customer support would typically review more than one trace.

BIBus Trace

- capture HTTP and socket requests sent to and from the BIBusTKServerMain process
- activate through environment variables or a configuration file
- log to a file or console

© 2015 IBM Corporation



This is for your information. You would only use this if you were asked to do so by customer support.


Business Analytics software

IBM

Trace the Dispatcher

- Trace the Dispatcher: trace start-up problems and inter-component communication
- add elements

© 2015 IBM Corporation



Tracing the dispatcher is useful for determining what is happening between a dispatcher and the other IBM Cognos services. The resulting files can be used to troubleshoot general, performance, affinity and load balancing issues.

Historically, the dispatcher was once known as pogo. If you see a configuration file that contains pogo, it is related to the dispatcher.

This is for your information. You would only use this if you were asked to do so by customer support.


Explore Query Framework Logging

- primary purpose of file is to configure various OLAP providers
- most of the OLAP providers have the following logging capabilities
 - dump incoming execute/validate requests as XML files
 - dump processing MDX statements into a log file
 - dump incoming metadata requests into a log file
 - output processing MDX statements into a debug window

© 2015 IBM Corporation

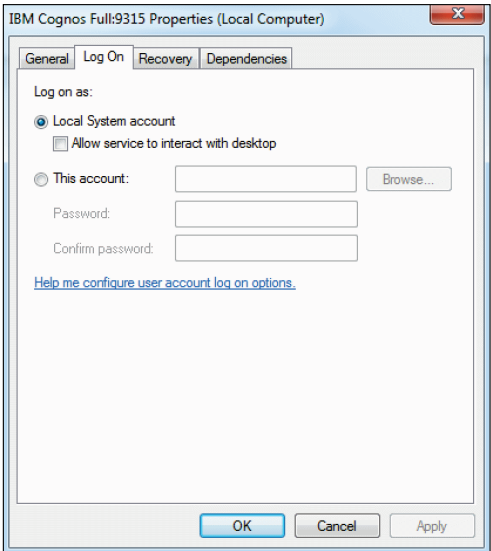



Query Framework logging can track information sent between components involved in processing OLAP requests.

Business Analytics software


Describe the Dispatcher Command Console

- activate
 - enable Allow service to interact with desktop functionality of the IBM Cognos 10 Windows service





© 2015 IBM Corporation

This console is only available on a Windows OS with a registered IBM Cognos 10 service. You can use startup.bat if in a development or test environment, as a quick way to start IBM Cognos 10 in a command window; do not use this method in a production environment.

You can use this to capture startup calls to troubleshoot service initialization issues.

Gateway Trace

- diagnose problems with single sign-on and secure sockets, standby dispatcher mechanism, and password encryption
- activate
 - in `..\cgi-bin\` directory, rename `cognoscgi.conf.sample` to `cognoscgi.conf`
 - add code to end of file


```
gw<type>LoggingConfig=<file_name>.log 5 GWSiMTLogger 1
```
- output: `..\logs\<file_name>.log`

© 2015 IBM Corporation



A gateway is an extension of a Web server program that transfers information from the Web server to another server. Gateways are often CGI programs, but may follow other standards such as Internet Server Application Program Interface (ISAPI), and Apache Modules (`apache_mod`).

An example of a `cognoscgi.conf` file (remove `#` to uncomment a line):

```
# optional overrides; defaults are shown, uncomment and edit as required
#dispatcher_host=localhost
#dispatcher_port=9300
#dispatcher_Encryption=enabled
#SystemRecoverableIterationLimit=50
gwISAPILoggingConfig=gwisapi.log 5 GWSiMTLogger 1
```

Valid `<type>` entries include `gwCGILoggingConfig` for CGI, `gwISAPILoggingConfig` for ISAPI, and `gwModLoggingConfig` for `apache_mod`.

Demo 1: Perform a Gateway Trace

At the beginning of this demo, no dispatcher is running.

Ensure that the following services are running:

- Apache Directory Server - default
- DB2-DB2COPY1 - DB2
- DB2 Remote Command Server (DB2COPY1)
- DB2DAS - DB2DAS00
- Lotus Domino Server (CProgramFilesx86IBMLotusDominodata)
- World Wide Web Publishing Service

Purpose:

You want to become familiar with the information available in a gateway trace. To do this, you will invoke a trace and review the output.

Task 1. Allow IIS 7 write permission to the logs directory.

In this environment, the IIS 7 Web server user (IUSR) requires permission to write in the logs directory. Before proceeding with the trace, you will enable this permission.

1. In Windows Explorer, navigate to **C:\Program Files\IBM\cognos\c10_64full**, right-click the **logs directory**, and then click **Properties**.
2. On the **Security** tab, to change permissions, click **Edit**.
3. Click **Add**.
4. In the **Enter the object names to select (examples)** pane, type **IUSR**, and then click **Check Names**.

If the user is found, it will appear with an underscore in the pane.

5. Click **OK**, and then in the **Group or user names** pane, click **IUSR**.
6. In the **Permissions for IUSR** pane, scroll to the **Write** permission, and in the **Allow** column, click the **Write** check box to select it.
7. Click **Apply**, and then click **OK** to close the **Permissions for logs** dialog box.
8. Click **OK** to close the **logs Properties** dialog box, and then close **Windows Explorer**.

Task 2. Enable the gateway trace.

1. On the **Taskbar**, click **Services**, ensure that the **IBM Cognos Full:9315** service has been stopped, and then close the **Services** window.
2. In **Windows Explorer**, navigate to **C:\Program Files\IBM\cognos\c10_64full\cgi-bin**, and then copy and paste **cognoscgi.conf.sample** to the same directory.
3. Rename to **cognoscgi.conf - Copy.sample** to **cognoscgi.conf**, and then open **cognoscgi.conf** in **Eclipse**.
4. Modify the code as follows:

- **dispatcher_host=vclassbase**
- **dispatcher_port=9315**
- **dispatcher_Encryption=enabled**
- **SystemRecoverableIterationLimit=50**
- **gwCGILoggingConfig=gwcgi.log 5 GWSiMTLogger 1**

This will direct the logging of the gateway to a log file called gwcgi.log. Do not add a path to the filename. The result of the modified section of text appears as follows:

```
# optional overrides... defaults are shown. Uncomment and edit as required.
dispatcher_host=vclassbase
dispatcher_port=9315
dispatcher_Encryption=enabled
SystemRecoverableIterationLimit=50
gwCGILoggingConfig=gwcgi.log 5 GWSiMTLogger 1
```

5. Save the file and then close **Eclipse**.
6. Start the **IBM Cognos Full:9315** service, clearing any messages of the service not starting in a timely fashion, and refreshing the status every 2 minutes until it has successfully started.

Task 3. Log on and run a report.

1. Launch **Internet Explorer**, go to **http://vclassbase:88/C10Full**, and then log on to the **LDAP_Dev** namespace with **admin/Education1** credentials.
2. Launch **IBM Cognos Connection**, navigate to **Public Folders\Samples_DQ\Models\GO Sales (query)\Report Studio Report Samples**, and run the **Horizontal Pagination_DQ** report.
3. Navigate through all pages of the report, and then close the browser window.

Task 4. Observe the results of the trace.

1. Open **C:\Program Files\IBM\cognos\c10_64full\logs\gwcgi.log** in **Eclipse**.

If a file is open in the middle pane (for example, cognoscgi.conf), close the tab first.

You will see results similar to the following:

```
11:40:51.656 - 2596 DEBUG t:2520 Calling CGIGatewayInterface::getenvvariable( REQUEST_METHOD )
11:40:51.656 - 2596 DEBUG t:2520 Leaving CGIGatewayInterface::getenvvariable() with value: POST
11:40:51.656 - 2596 DEBUG t:2520 started method= POST
11:40:51.656 - 2596 DEBUG t:2520 Calling CGIGatewayInterface::getenvvariable( QUERY_STRING )
11:40:51.656 - 2596 DEBUG t:2520 Leaving CGIGatewayInterface::getenvvariable() with value:
11:40:51.656 - 2596 DEBUG t:2520 Calling CGIGatewayInterface::getenvvariable( REQUEST_METHOD )
11:40:51.656 - 2596 DEBUG t:2520 Leaving CGIGatewayInterface::getenvvariable() with value: POST
11:40:51.656 - 2596 DEBUG t:2520 Calling CGIGatewayInterface::getenvvariable( CONTENT_LENGTH )
```

2. Scroll through the file to see the information that is captured in this trace. Items to look for include SOAP messages, HTML code, and elapsed time (scroll to the bottom of the file).
3. When you have finished reviewing the file, close the **gwcgi.log** tab, and then close **Eclipse**.

Task 5. Disable the trace.

1. From the **System Tray**, click **Services** window, stop the **IBM Cognos Full:9315** service.
2. In **Windows Explorer**, navigate to **C:\Program Files\IBM\cognos\c10_64full\cgi-bin**, and delete **cognoscgi.conf**.
Leave the **IBM Cognos Full:9315** service stopped for the next workshop.

Results:

You invoked a CGI gateway trace and reviewed the information available in the output.

Why Use a UDA Trace?

- capture SQL sent to native API for purposes of executing SQL through UDATest.exe
- analyze performance issues
- diagnose reporting database connection issues

© 2015 IBM Corporation



The Universal Data Access (UDA) layer is the component used for the processing of SQL queries to relational data sources.

UDATest is a utility which ships with Cognos 10. The purpose of the utility is to allow users to execute the SQL generated by UDA, which has been previously captured in a UDA trace, directly against a third party RDBMS. The output generated by UDATest from the SQL execution will show the attach to the 3rd party RDBMS API, the preparation of the SQL, the actual SQL, and will display the data returned.

It is important to understand that UDATest does not fix issues. Its purpose is to verify if the SQL sent to the RDBMS will execute successfully

Possible use cases of UDATest. Keep in mind that in some of these examples, the SQL generated was identical or ran successfully. UDATest did not help resolve the issue, but rather proved the SQL sent to the RDBMS could execute successfully and therefore helped the administrator determine that the actual issue may be elsewhere:

- Report runs against one DB vendor and fails against another. For example, you create a report against SQL Server, the report validates and runs as expected, then you create a similar report against a DB2 data source and the report fails with a UDA error. Run UDATest against both vendors RDBMS and compare the output to see where the process is failing and how the SQL differs.
- SQL Performance with Virtual View Manager versus directly against the vendor: Running a report which uses a Composite data source runs in x minutes. Running the same SQL, directly against Oracle takes x + minutes. Run UDATest, with output logging enabled, against both and compare to see what the differences are in the SQL.
- Upgrade: Comparing SQL generated in different versions of Cognos 10: Running a report against Cognos 10.1 runs successfully, running the same report after upgrading to Cognos 10.2 fails with an error. Run UDATest to see the differences in the SQL being generated.
- Intermittent Report Failures: Every day a summary report is sent out. Since upgrading report returns incorrect data usually once or twice a week. This is not due to a database issue because if you re-run the report again you will get the correct answer. Run UDATest a number of times in sequence, waiting a couple of seconds between each launch of UDATest.
See good results for run#2, run# 3, run#4 and run# 5: uda_2.log, uda_3.log, uda_4.log, uda_5.log. See bad results for run# 1, run #6 and run# 7: uda_1.log, uda_7.log and uda_7.log. Compare the logs to see the differences.

Describe Methods to Enable UDA Traces

Method	Pros	Cons
Server Environment Variables	<ul style="list-style-type: none"> ▪ set any EVs ▪ logs until EVs are unset 	<ul style="list-style-type: none"> ▪ requires IBM Cognos service restart
Session Environment Variables	<ul style="list-style-type: none"> ▪ set any EVs ▪ no IBM Cognos restart required 	<ul style="list-style-type: none"> ▪ variables only good for session
Use ipfUDAClientConfig.xml	<ul style="list-style-type: none"> ▪ captures performance categories ▪ no IBM Cognos restart required 	<ul style="list-style-type: none"> ▪ will probably want to use other IPF files at the same time and merging IPF files takes care and consideration

© 2015 IBM Corporation



There are several methods to enable UDA traces. The method you choose depends on your circumstances for enabling the UDA trace, your comfort level with the method and what flexibility you require for the tracing.

Before enabling UDA tracing:

- check the IBM Cognos *Business Intelligence Troubleshooting Guide 10.2.2* and the Readme file for common UDA issues
- search the IBM technical support knowledge base ([http://www-947.ibm.com/support/entry/portal/troubleshooting/software/software_support_\(general\)](http://www-947.ibm.com/support/entry/portal/troubleshooting/software/software_support_(general))) for the UDA error message and see if any are similar to the issue that is being encountered
- search the IBM Cognos Proven Practice document library (<http://www.ibm.com/developerworks/data/library/cognos/cognosprovenpractices.html/>) for potential solutions

Enable UDA tracing:

- To capture the SQL sent to the native API for the purposes of executing that SQL through .bin/UDATest.exe, thereby taking the rest of the Cognos services and components out of the picture.
- To help in analyzing performance issues. If you have encountered a performance issue where the actual slow points of performance have not been isolated across components you may want to enable UDA logging. Be aware that one will probably need to capture traces from other components to establish a timeline of activity across all components.
- The key place where the timeline becomes important is establishing where components are waiting for their consumers (either other Cognos components or 3rd Party APIs) to re-invoke them, etc. Looking at a single component trace on its own can mask that delay.
- The cogserver.log does not provide enough granularity to pinpoint where the performance hit is taking place.

Explore UDA Trace and Environment Variables

- activate with server environment variables
 - `TRACE_FILE=<trace_file_location>/<filename>.xml`
 - `TRACE_ALL_THREADS=Y`
 - `TRACE_LAYER_DMD_SQLAPIRW = 0x0002`

© 2015 IBM Corporation



When using `TRACE_FILE=<trace_file_location>/<filename>.xml`, if you are a Visual Studio user, if this is set to a value of `::DEBUGWIN`, the output will be directed to Visual Studio.

`TRACE_ALL_THREADS=Y` will catch tracing from all threads.

`TRACE_LAYER_DMD_SQLAPIRW = 0x0002` performs SQL API tracing.

There are other environment variables that are organized into trace layers with 32 trace categories within each layer. These environment variables apply to IBM Cognos only and will not affect any other processes.

For the settings in the environment variables to take effect, the IBM Cognos BI service will have to be re-started, as setting environment variables is a function of the operating system. In Windows, open My Computer, click View system information, click the Advanced tab, click the Environment Variables button. For UNIX/Linux, environment variables are usually set in shell scripts.

Describe Common Trace Layers and Categories

Trace Layer	Category
TRACE_LAYER_COMMON=0x0004	Trace low level initialization
TRACE_LAYER_COMMON=0x0008	Trace loading/unloading of shared libraries
TRACE_LAYER_DMD_SQLAPIRW = 0x0002	SQL API tracing
TRACE_LAYER_GENERIC=0x0400	Trace sort statistics
TRACE_FORMAT=bare	Removes trace file formatting such as the XML tags

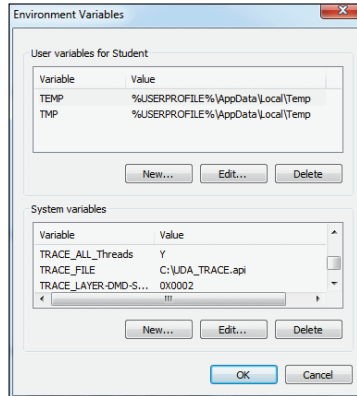
© 2015 IBM Corporation



The most common traces are displayed above.

Example of a UDA Trace

- `TRACE_FILE = x:\<path>\<name>.api`
- `TRACE_LAYER_DMD_SQLAPIRW = 0x0002`
- `TRACE_ALL_THREADS=Y`



© 2015 IBM Corporation



The trace file can be located in any place that is accessible by the IBM Cognos 10 server and can be named at your discretion. The file extension does not matter; it could be .xml, .log, .txt, and so on. UDATest picks up .api by default, but you can just point to the trace file using Show all files.

The IBM Cognos Service has to be restarted for the trace to be enabled.

The example shows the setting of System Environment variables for UDA trace on a Windows server. Customer support frequently uses UDA traces for troubleshooting.

Activate UDA Trace With Session Variables

- stop IBM Cognos service
- modify ..\bin\startup.bat, add code below @echo off
- save and run startup.bat file to start IBM Cognos 10

```
@ echo off
set TRACE_FILE = <drive letter>:\<path>\<name>.api
set TRACE_LAYER_DMD_SQLAPIRW = 0x0002
set TRACE_ALL_THREADS=Y
```

Example

```
@ echo off
set TRACE_FILE = C:\UDATrace.txt
set TRACE_LAYER_DMD_SQLAPIRW = 0x0002
set TRACE_ALL_THREADS=Y
```

© 2015 IBM Corporation

This technique really only applies to Windows platforms and is only applicable in development or test environments since startup.bat should not be used in a production environments.

In the example shown, press Ctrl+C to stop.

Activate UDA Trace with IPF File

- rename
 `..\configuration\ipfUDAcientconfig.xml.sample` to
 `ipfclientconfig.xml`
- output to `..\logs` directory:
 - `UDA_Trace.log`
 - `UDA_Perf.log`
 - `UDA_RTUsage.log`

© 2015 IBM Corporation



Tracing will take effect within 30 seconds.

A UDA trace can be useful to determine how long things are taking with the database.

Business Analytics software

Example UDA Output

```

<?xml version="1.0"?><UDATrace version="UDA-AW-ML-8.500-WIP-31860-0(Production)" build="31860" patch="0"
Date="2015/03/11 11:06"><TraceEvent Layer="8" Category="2" ThreadId="00000C84"
Timestamp="1469"><SQLAPI><![CDATA[initialize e1 for "en" metadata callback m1,ThreadId="00000C84"
Timestamp="1469"><SQLAPI><![CDATA[initialize e1 for "en" metadata callback m1,
sql99datatypes;]]></SQLAPI></TraceEvent><TraceEvent Layer="8" Category="2" ThreadId="00000C84"
Timestamp="1469"><SQLAPI Id="e1"><![CDATA[multidbattachdirect d10001 dbloglname "CQE_DB" in e1(dbloglname "SQL-
Environment" "" QS );]]></SQLAPI></TraceEvent><TraceEvent Layer="8" Category="2" ThreadId="00000C84"
Timestamp="1469"><SQLAPI Id="e1"><![CDATA[show features d10001 passive;]]></SQLAPI></TraceEvent><TraceEvent
Layer="8" Category="2" ThreadId="00000C84" Timestamp="1469"><SQLAPI Id="e1"><![CDATA[start t10001 for d10001
read;]]></SQLAPI></TraceEvent><TraceEvent Layer="8" Category="2" ThreadId="00000C84" Timestamp="1469"><SQLAPI
Id="e1"><![CDATA[prepare r10001 from"Select * from [gosales].PRODUCT_LINE" in t10001 tag sql, enable
nagging;]]></SQLAPI></TraceEvent><TraceEvent Layer="8" Category="2" ThreadId="00000C84" Timestamp="1469"><SQLAPI
Id="e1"><![CDATA[show tagged sql for r10001;]]></SQLAPI></TraceEvent><TraceEvent Layer="8" Category="2"
ThreadId="00000C84" Timestamp="1469"><SQLAPI Id="e1"><![CDATA[release request
r10001;]]></SQLAPI></TraceEvent><TraceEvent Layer="8" Category="2" ThreadId="00000C84"
Timestamp="1531"><SQLAPI><![CDATA[initialize e2 for "en" metadata callback m2,
sql99datatypes;]]></SQLAPI></TraceEvent><TraceEvent Layer="8" Category="2" ThreadId="00000C84"
Timestamp="1531"><SQLAPI Id="e2"><!["CQE_DB" in e2(dbloglname "great_outdoors_sales"
"DBInfo_Type=MS;Provider=SQLOLEDB;User ID=sa;Password=*****;Data Source=localhost\\SQL2005;Provider_String=Initial
Catalog=gosales;@COLSEQ=" OL );]]></SQLAPI></TraceEvent><TraceEvent Layer="8" Category="2" ThreadId="00000C84"
Timestamp="1672"><SQLAPI Id="e2"><![CDATA[get attribute max name length context dbname "great_outdoors_sales" for
database d20001 ;]]></SQLAPI></TraceEvent><TraceEvent Layer="8" Category="2" ThreadId="00000C84"
Timestamp="1672"><SQLAPI Id="e2"><![CDATA[show features d20001 case sensitive;]]></SQLAPI></TraceEvent><TraceEvent
Layer="8" Category="2" ThreadId="00000C84" Timestamp="1672"><SQLAPI Id="e2"><![CDATA[show features d20001 case
sensitive;]]></SQLAPI></TraceEvent><TraceEvent Layer="8" Category="2" ThreadId="00000C84" Timestamp="1672"><SQLAPI
Id="e1"><![CDATA[prepare r10002 from"select PRODUCT_LINE.PRODUCT_LINE_CODE

```

© 2015 IBM Corporation

In the slide is a snippet of some UDA output. The items in red bold are the connect string, SQL and bulkfetch. Timings are in milliseconds. The second timing in red bold (Timestamp="1531") is the connection to the 3rd party data source vendors API. The line (Timestamp="1672") is the first response back from that call. Therefore you can determine that at least part of 1.5 seconds of this request was the time it took to attach and get a response back.

By finding the timestamp in the log file, looking at the response, and looking at the timestamp, you can determine timings to help troubleshoot. For example, how long did it take to log on? This trace may help you to identify where the time is being spent, including in connection of third party database vendor applications.

This type of code will be displayed in Workshop 1: Explore UDA Trace and UDATest. In the example description provided for the scenario, a use case for this would be to say, for example, that previously a report was taking 8 seconds to run. Through a UDA trace you were able to determine that 5 seconds was taken up by this attach time. The issue in this case was a bug in a third party vendor driver which was taking too long to encrypt the information. A bug was logged with the vendor and eventually the report execution time was reduced by 3 seconds. The connection time will never be zero as all vendors require some time to process this information.

What is UDATest?

- utility in IBM Cognos 10 at `..\bin\udatest.exe` or `..\bin64\udatest.exe`
- users can execute SQL generated by UDA directly against a third party data source
- UDATest verifies if the SQL sent to the RDBMS will execute successfully

© 2015 IBM Corporation



It is important to understand that UDATest does not fix issues. Over the next few pages, some possible use cases for UDATest are presented.

UDATest Scenario 1

- report runs against one DB vendor and fails against another
 - you create a report against SQL Server, the report validates and runs as expected
 - you create a similar report against a DB2 data source and the report fails with a UDA error
 - run UDATest against each vendor RDBMS and compare output to see where process is failing and how the SQL differs

© 2015 IBM Corporation



UDATest Scenario 2

- SQL Performance with Composite/VVM vs. directly against the vendor
 - a report which uses a Composite/VVM data source runs in x minutes
 - running the same SQL, directly against Oracle takes x + minutes
 - run UDATest, with output logging enabled, against both, and compare differences in the SQL



UDATest Scenario 3

- compare SQL generated on different operating systems
 - report performance on AIX going through DataDirect ODBC driver to MS SQL Server is unacceptable
 - database administrators see multiple cursors on AIX
 - running on Windows, the cursors are not sent to the database and performance is acceptable
 - run UDATest against both, and review the SQL generated and the output it produced

© 2015 IBM Corporation



UDATest Scenario 4

- compare SQL generated in different versions of IBM Cognos BI
 - running a report against IBM Cognos 8.4.1 runs successfully
 - running the same report after upgrading to IBM Cognos 10 fails with an error
 - report still fails with an error when upgraded to IBM Cognos 10.2.2
 - run UDATest to see the differences in the SQL being generated

© 2015 IBM Corporation



UDATest Scenario 5

- specific version of IBM Cognos
 - running a report in IBM Cognos 8.4.1 runs successfully
 - same report fails with an error in IBM Cognos 10
 - same report runs successfully in IBM Cognos 10.2.2
- probably not a database issue
- run UDATest and compare the resulting SQL to look for differences

© 2015 IBM Corporation



UDATest Scenario 6

- intermittent report failures
 - each day a summary report is sent out
 - since upgrading, the report returns incorrect data usually once or twice a week
 - not due to a database issue because if you re-run the report again you will get the correct data
 - run UDATest a number of times in sequence, waiting a couple of seconds between each launch of UDATest
 - see good results for run #2, run #3, run #4 and run #5: uda_2.log, uda_3.log, uda_4.log, uda_5.log.
 - see bad results for run #1, run #6 and run #7: uda_1.log, uda_7.log and uda_7.log.
 - compare the logs to see the differences

© 2015 IBM Corporation



Be aware that UDA is not used with the QueryService which processes the 64-bit ReportServer service. The scenario presented here assumes that you are running a report against the 32-bit ReportServer service. For 64-bit ReportServer traces of queries, use the Dynamic Query Analyzer utility.

Workshop 1: Explore UDA Trace and UDATest (Optional)

UDA is not used with the QueryService which processes the 64-bit ReportServer service. In this workshop, you will run a report against the 32-bit ReportServer service. For 64-bit ReportServer traces of queries, use the Dynamic Query Analyzer utility.

You are an administrator testing a report in Report Studio, and want to know if the query submitted in IBM Cognos returns the correct information from the database. You need to extract the native SQL and query execution plan of Relational Query Planner (RQP). The native SQL then can be tested in the database vendor's native query tool. You will use the UDATest utility to accomplish this.

UDATest is used to isolate the database information. You will perform a trace to get this information.

You will:

- Ensure that the IBM Cognos Full:9315 service and the IBM Cognos DispCM:9320 service are stopped.
- Enable SQL comments in CQEconfig.xml making it easier to identify the query and the user running it. The log file created will be tested using the udatest.exe and the native query will be executed in the native client query tool.
- Set Universal Data Access (UDA) environment variables (on both Dispatchers) to enable the creation of the log files when a query is executed.
- Create a Report Studio report (using admin/Education1 credentials) that will be used to run UDA trace.
- Modify the uda_trace.log file and save as .api file.

- Run udatest.exe to examine the .api file from the Cognos 10 bin folder.
- Copy the native SQL from the .api file and test its performance in IBM Data Studio.
- Reset all configurations back to their original state

Keep in mind, that in a real troubleshooting scenario, it is best to run multiple traces (UDA, BIBus, Dispatcher) to get a better understanding of the causes of the issues.

For more information about where to work and the workshop results, refer to the Tasks and Results section that follows. If you need more information to complete a task, refer to earlier demos for detailed steps.

Workshop 1: Tasks and Results

At the beginning of this workshop, no dispatcher is running.

Task 1. Enable SQL comments in the UDA trace and native SQL.

- In the **Services** window, ensure the **IBM Cognos Full:9315** service and the **IBM Cognos DispCM:9320** service are stopped.
- In **Windows Explorer**, navigate to **C:\Program Files\IBM\cognos\c10_64full\configuration**, and then copy and paste **CQEConfig.xml.sample** to the same directory.
- Rename to **CQEConfig.xml - Copy.sample** to **CQEConfig.xml**, and then open **CQEConfig.xml** in **Notepad**.

- Locate the following section:

```
<section name="QueryEngine">
```

```
...
```

```
</section>
```

- Uncomment the following four lines of code in this section by deleting the **!--** characters:

```
<!-- entry name="GenerateCommentInNativeSQL" value="1"-->
```

```
<!-- entry name="GenerateCommentInCognosSQL" value="1"-->
```

```
<!-- entry name="NativeCommentMacro" value="#'user=' +
$account.defaultName + ' report-Path=' + $reportPath + ' queryName='
+ $queryName + 'REMOTE_ADDR=' + $REMOTE_ADDR + ' SERVER_NAME='
+$SERVER_NAME + ' requestID=' + $requestID#"/-->
```

```
<!-- entry name="CognosCommentMacro" value="#'user=' +
$account.defaultName + ' report-Path=' + $reportPath + ' queryName='
+ $queryName + 'REMOTE_ADDR=' + $REMOTE_ADDR + ' SERVER_NAME='
+$SERVER_NAME + ' requestID=' + $requestID#"/-->
```

You can also copy the contents of **C:\Edcognos\B5A19\08-Explore_Component_Logging\Mod 8_Wkshp 1_Task 1.txt** into **CQEconfig.xml**, replacing these lines of code.

The result appears as follows:

```

<section name="QueryEngine">
  <!-- Description: queryReuse feature -->
  <!-- value="0" means disable the feature -->
  <!-- default is value="5" which means cache up to 5 result sets per
session -->
  <entry name="queryReuse" value="5"/>
  <!-- -->
  <!-- Description: References to model query items may have 2-part
names. (default(off)=0; choices=0,1) -->
  <!-- Off: A parsing error is returned for a reference to a model
query item using a 2-part name. -->
  <!-- On: The expression resolver will allow 2-part name references
to model query items. -->
  <!-- NOTE: Cognos 8.1 MR2 was the last release in which the default
setting allowed 2-part names. This release (8.2) -->
  <!-- has the default set to disallow 2-part names. The next release
(8.3) will no longer allow 2-part names. -->
  <!-- The use of 2-part names will generate the QE-DEF-496 warning
message in the log file. -->
  <!-- entry name="AllowModelQueryItem2PartNameReference"
value="0"/-->
  <!-- -->
  <!-- Generation of comments in native sql and cognos sql.-->
  <entry name="GenerateCommentInNativeSQL" value="1"-->
  <!-- ( default(off)=0, on=1) -->
  <entry name="GenerateCommentInCognosSQL" value="1"-->
  <!-- ( default(off)=0, on=1) -->
  <!-- The content of the comments is controlled with two entries,
their defaults are specified in the value attribute -->
  <entry name="NativeCommentMacro" value="'user=' +
$account.defaultName + ' reportPath=' + $reportPath + ' queryName=' + $queryName + '
REMOTE_ADDR=' + $REMOTE_ADDR + ' SERVER_NAME=' + $SERVER_NAME + ' requestID=' +
$requestID#"/-->
  <entry name="CognosCommentMacro" value="'user=' +
$account.defaultName + ' reportPath=' + $reportPath + ' queryName=' + $queryName + '
REMOTE_ADDR=' + $REMOTE_ADDR + ' SERVER_NAME=' + $SERVER_NAME + ' requestID=' +
$requestID#"/-->

```

- Save the file, and then close Notepad.

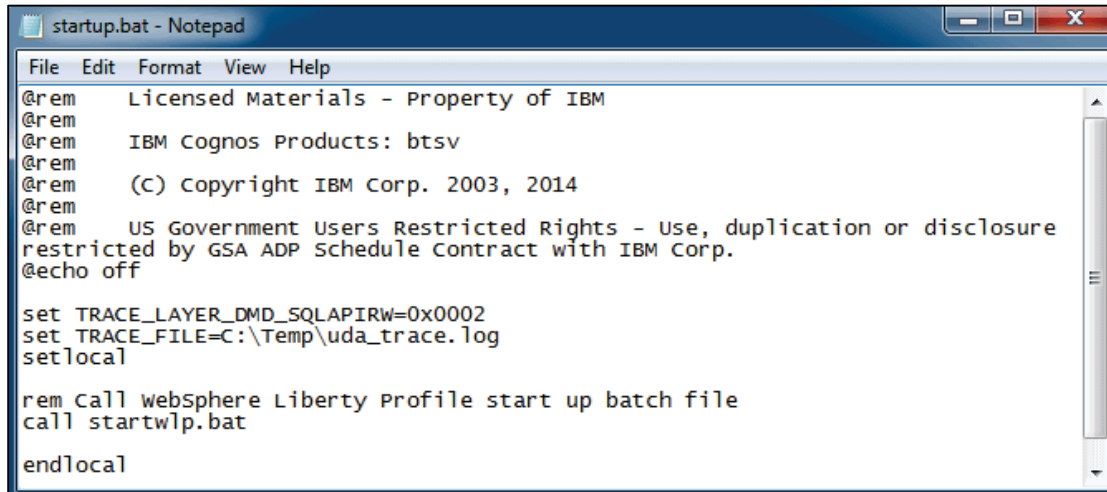
Task 2. Set environment variables to enable UDA tracing.

- In Windows Explorer, create a folder named Temp at the root of C:.
- Navigate to C:\Program Files\IBM\cognos\c10_64full\bin64, right-click startup.bat, and then click Properties.
- Ensure that the Read-only check box is clear, and then click OK.
- Right-click startup.bat, and then click Edit.

- Preceding the **setlocal** line, add the following lines of code:

```
set TRACE_LAYER_DMD_SQLAPIRW=0x0002
set TRACE_FILE=C:\Temp\uda_trace.log
```

The results appear as follows:



```
startup.bat - Notepad
File Edit Format View Help
@rem Licensed Materials - Property of IBM
@rem
@rem IBM Cognos Products: btsv
@rem
@rem (C) Copyright IBM Corp. 2003, 2014
@rem
@rem US Government Users Restricted Rights - Use, duplication or disclosure
restricted by GSA ADP Schedule Contract with IBM Corp.
@echo off

set TRACE_LAYER_DMD_SQLAPIRW=0x0002
set TRACE_FILE=C:\Temp\uda_trace.log
setlocal

rem call websphere Liberty Profile start up batch file
call startwlp.bat

endlocal
```

This trace will contain all SQL statements sent to the UDA layer. In addition it will have the timings of how long each statement took to execute. The trace file will be written to `uda_trace.log`.

- Save and then close the file.
- Repeat the preceding steps for the **C:\Program Files\IBM\cognos\c10_64DispCM\bin64\startup.bat** file.

Task 3. Create a Report Studio report that will be used to run the UDA trace.

- Double-click **C:\Program Files\IBM\cognos\c10_64full\bin64\startup.bat** to start the IBM Cognos 10 - 64 Full instance with the new `CQEconfig.xml` and environment variable settings.

If a Windows Security Alert dialog box appears with a message related to Windows Firewall, select only the Private networks, such as my home or work network check box, and then click **Allow access**.

You should not use startup.bat to start IBM Cognos 10 in a production environment, but you are doing testing in a non-production environment for this workshop.


Allow a few minutes for the services to start.

- When **The dispatcher is ready to process requests** message appears in the command window, repeat with the **C:\Program Files\IBM\cognos\c10_64DispCM\bin64\startup.bat** file to start the IBM Cognos 10 - 64 DispCM instance.

UDA is not used with the QueryService which processes the 64-bit ReportServer service. You will be running a report against the 32-bit ReportServer service, and in this environment, that is on the DispCM server. Therefore you must start both instances to execute the trace. For 64-bit ReportServer traces of queries, use the Dynamic Query Analyzer utility.

- Launch **Internet Explorer**, go to **http://vclassbase:88/C10Full**, and then log on to the **LDAP_Dev** namespace with **admin/Education1** credentials.
- On the **IBM Cognos software** page, click **Author advanced reports** to launch Report Studio.
- In the **List of all packages**, navigate to **Public Folders\Samples\Models\GO Sales (query)** package, when prompted, click **Create New**, and then double-click **List**.

It may take a few moments for the package to load in Report Studio.

- From the **Source** tab, populate the list report with the following items from the **Sales (query)** namespace:
 - **Products: Product line, Product type**
 - **Sales: Product cost**
- In the report layout, click the **<Product line>** column, and then on the toolbar click **Group / Ungroup** .



- On the **Properties** pane in the bottom left corner, click **Select Ancestor**, and then click **List**.
- In the **Data** section of the **Properties** pane, change the **Rows Per Page** to **25**.
- Save the report as **UDA Test** in the **GO Sales (query)** package.
Ensure that the report is saved to a location that is accessible to all, and is not saved to a private location, like My Folders.
- Close **Report Studio**, launch **IBM Cognos Connection**, navigate to the **Public Folders\Samples\Models\GO Sales (query)** package, and then click **UDA Test** to run the report.

You may have to click Refresh to see the report entry listed. Count the number of rows of data that are returned in the report.

The results appear similar to the following:

Product line	Product type	Product cost
Camping Equipment	Cooking Gear	\$167,128,146.52
	Lanterns	\$72,808,366.01
	Packs	\$213,232,893.15
	Sleeping Bags	\$188,159,844.17
	Tents	\$360,908,320.53
Golf Equipment	Golf Accessories	\$19,927,608.85
	Irons	\$135,602,473.09
	Putters	\$55,499,773.14
	Woods	\$163,187,870.64
Mountaineering Equipment	Climbing Accessories	\$39,594,854.17
	Rope	\$78,380,688.26
	Safety	\$52,250,948.68
	Tools	\$76,157,733.00

- After the report opens in **IBM Cognos Viewer**, leave the browser open.

Task 4. Modify the uda_trace.log file and save as an .api file.

- In **Windows Explorer**, navigate to **C:\Temp**, to identify the presence of the **uda_trace.<PID #>.log** file.

If there is more than one log file, it is because the trace was enabled on both dispatchers. You will want to open the file with the most recent timestamp, and which has some content in it. Likely the other file will be 0 KB in size and will be blank.

- Open the **uda_trace.<PID #>.log** file in **Eclipse** and find *********.

This line will contain the connection string information to the database. The eight asterisks represent the connection password to the database.

- Change ********* to **Education1**, and then close the **Find/Replace** dialog box.
- Press **Ctrl+Home** to return to the beginning of the file, find the second instance of **multidbattach**, and then copy the following text from the file to the clipboard:

```
multidbattachdirect d20001 dblogicname "CQE_DB" in e2(dblogicname
"great_outdoors_sales"
"DSN=GS_DB;UID=GOSALES;PWD=Education1;@ASYN=0@0/0@COLSEQ=IBM_JD_CNX
_STR:^User ID:^?Password:;LOCAL;JD-
D2;URL=jdbc:db2://VCLASBASE:50000/GS_DB;DRIVER_NAME=com.ibm.db2.jcc
.DB2Driver" D2 );
```

- In **Windows Explorer**, navigate to **C:\Edcognos\B5A19\08-Component_Logging**, and then open **Mod 8_Wkshp 1_Task 4.txt** in **Notepad**.

This file will serve as a template to create a query for UDATest that will return the native SQL and the execution plan.

- Delete the following text:

<<INSERT MULTIDBATTACHDIRECT STATEMENT HERE>>;

- Paste the contents of the clipboard.
- Locate the following section:

get attribute max name length context dbname "<<INSERT DATA SOURCE NAME HERE>>" for database d20001 ;

- Replace **<<INSERT DATA SOURCE NAME HERE>>** with **great_outdoors_sales**.

This line appears as follows:

```
get attribute max name length context dbname "great_outdoors_sales"
for database d20001 ;
```

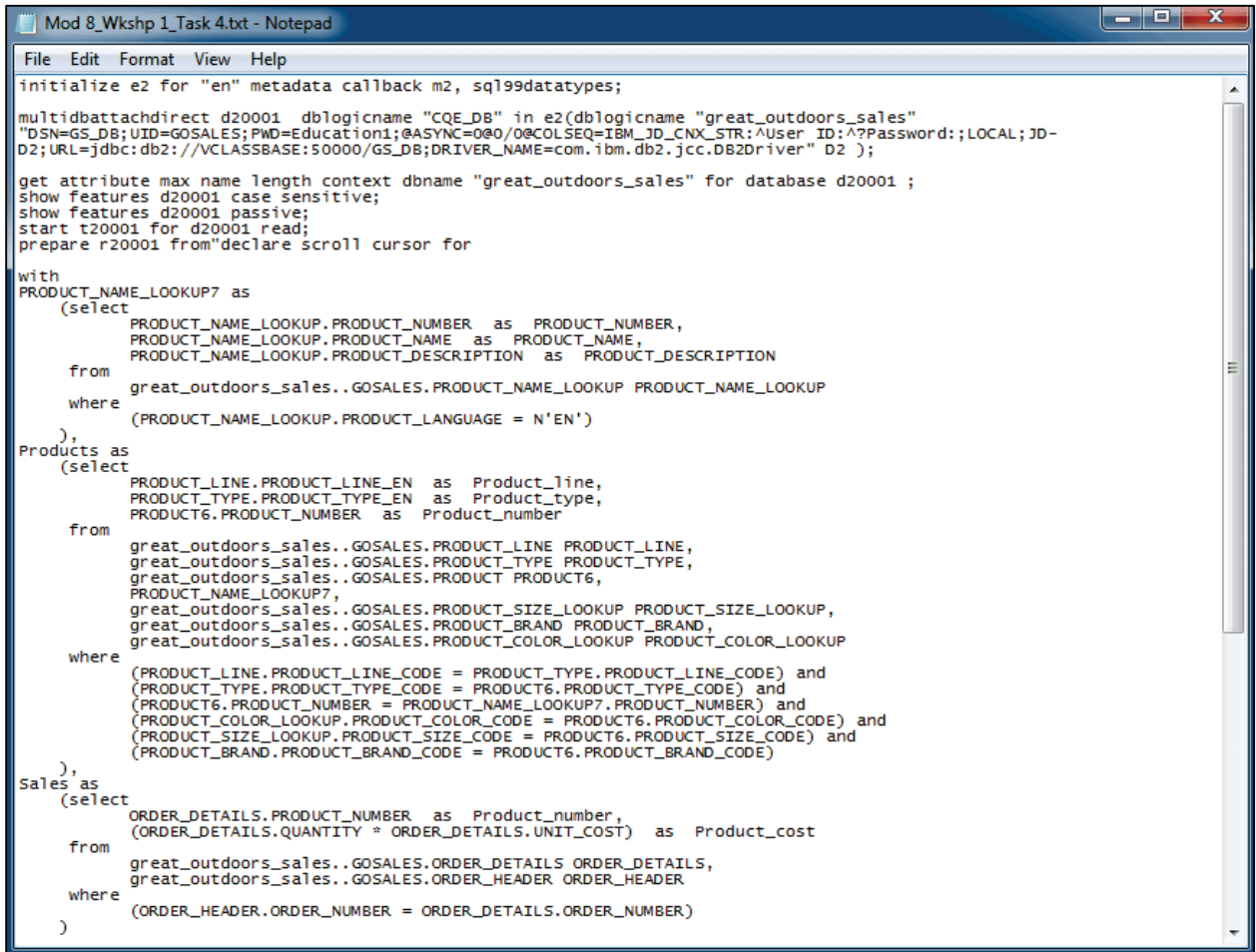
- In **Eclipse**, in the **uda_trace.<PID #>.log** file, find **with**, and then search for the semicolon at the end of the SQL statement.
- Copy all the text from the **with** clause to the end of the SQL including the **avoid zero division** and **;** (semicolon), to the clipboard.
- In the **Mod 8_Wkshp 1_Task 4.txt** file, delete the following text:
<<INSERT SQL STATEMENT HERE>>

- Paste the contents of the clipboard.

In this example, the two features for which you want to get information are:

- **show native sql for r20001;** (this will generate the native SQL)
- **show execution plan for r20001;** (this will show the execution plan for RQP)

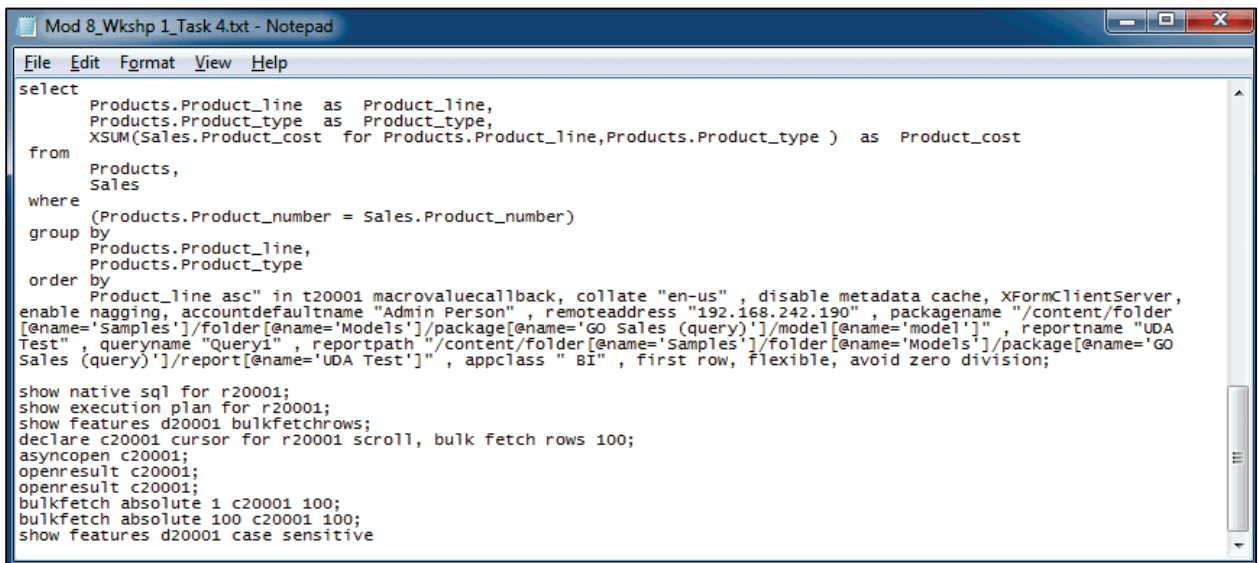
The results appear as follows:



```

Mod 8_Wkshp 1_Task 4.txt - Notepad
File Edit Format View Help
initialize e2 for "en" metadata callback m2, sql99datatypes;
multidbattachdirect d20001 dblogname "CQE_DB" in e2(dblogname "great_outdoors_sales"
"DSN=GS_DB;UID=GOSALES;PWD=Education1;@ASYN=0@0/0@COLSEQ=IBM_JD_CNX_STR:User_ID:Password:LOCAL;JD-
D2;URL=jdbc:db2://VCLASBASE:50000/GS_DB;DRIVER_NAME=com.ibm.db2.jcc.DB2Driver" D2 );
get attribute max name length context dbname "great_outdoors_sales" for database d20001 ;
show features d20001 case sensitive;
show features d20001 passive;
start t20001 for d20001 read;
prepare r20001 from"declare scroll cursor for
with
PRODUCT_NAME_LOOKUP7 as
(select
PRODUCT_NAME_LOOKUP.PRODUCT_NUMBER as PRODUCT_NUMBER,
PRODUCT_NAME_LOOKUP.PRODUCT_NAME as PRODUCT_NAME,
PRODUCT_NAME_LOOKUP.PRODUCT_DESCRIPTION as PRODUCT_DESCRIPTION
from
great_outdoors_sales..GOSALES.PRODUCT_NAME_LOOKUP PRODUCT_NAME_LOOKUP
where
(PRODUCT_NAME_LOOKUP.PRODUCT_LANGUAGE = N'EN')
),
Products as
(select
PRODUCT_LINE.PRODUCT_LINE_EN as Product_line,
PRODUCT_TYPE.PRODUCT_TYPE_EN as Product_type,
PRODUCT6.PRODUCT_NUMBER as Product_number
from
great_outdoors_sales..GOSALES.PRODUCT_LINE PRODUCT_LINE,
great_outdoors_sales..GOSALES.PRODUCT_TYPE PRODUCT_TYPE,
great_outdoors_sales..GOSALES.PRODUCT PRODUCT6,
PRODUCT_NAME_LOOKUP7,
great_outdoors_sales..GOSALES.PRODUCT_SIZE_LOOKUP PRODUCT_SIZE_LOOKUP,
great_outdoors_sales..GOSALES.PRODUCT_BRAND PRODUCT_BRAND,
great_outdoors_sales..GOSALES.PRODUCT_COLOR_LOOKUP PRODUCT_COLOR_LOOKUP
where
(PRODUCT_LINE.PRODUCT_LINE_CODE = PRODUCT_TYPE.PRODUCT_LINE_CODE) and
(PRODUCT_TYPE.PRODUCT_TYPE_CODE = PRODUCT6.PRODUCT_TYPE_CODE) and
(PRODUCT6.PRODUCT_NUMBER = PRODUCT_NAME_LOOKUP7.PRODUCT_NUMBER) and
(PRODUCT_COLOR_LOOKUP.PRODUCT_COLOR_CODE = PRODUCT6.PRODUCT_COLOR_CODE) and
(PRODUCT_SIZE_LOOKUP.PRODUCT_SIZE_CODE = PRODUCT6.PRODUCT_SIZE_CODE) and
(PRODUCT_BRAND.PRODUCT_BRAND_CODE = PRODUCT6.PRODUCT_BRAND_CODE)
),
Sales as
(select
ORDER_DETAILS.PRODUCT_NUMBER as Product_number,
(ORDER_DETAILS.QUANTITY * ORDER_DETAILS.UNIT_COST) as Product_cost
from
great_outdoors_sales..GOSALES.ORDER_DETAILS ORDER_DETAILS,
great_outdoors_sales..GOSALES.ORDER_HEADER ORDER_HEADER
where
(ORDER_HEADER.ORDER_NUMBER = ORDER_DETAILS.ORDER_NUMBER)
)

```

```

select
  Products.Product_line as Product_line,
  Products.Product_type as Product_type,
  XSUM(Sales.Product_cost for Products.Product_line,Products.Product_type ) as Product_cost
from
  Products,
  Sales
where
  (Products.Product_number = Sales.Product_number)
group
  by
  Products.Product_line,
  Products.Product_type
order
  by
  Product_line asc" in t20001 macrovaluecallback, collate "en-us" , disable metadata cache, XFormClientServer,
enable nagging, accountdefaultname "Admin Person" , remoteaddress "192.168.242.190" , packagename "/content/folder
[@name='Samples']/folder[@name='Models']/package[@name='GO Sales (query)']/model[@name='model1']" , reportname "UDA
Test" , queryname "Query1" , reportpath "/content/folder[@name='Samples']/folder[@name='Models']/package[@name='GO
Sales (query)']/report[@name='UDA Test']" , appclass " BI" , first row, flexible, avoid zero division;

show native sql for r20001;
show execution plan for r20001;
show features d20001 bulkfetchrows;
declare c20001 cursor for r20001 scroll, bulk fetch rows 100;
asynccopen c20001;
openresult c20001;
openresult c20001;
bulkfetch absolute 1 c20001 100;
bulkfetch absolute 100 c20001 100;
show features d20001 case sensitive

```

- In **Notepad**, from the **File** menu, click **Save As**, and then in the **Save in** list, navigate to **C:\Program Files\IBM\cognos\c10_64full\bin64**.
- In the **File name** box, type **Test.api**, in the **Save as type** list, click **All Files (*.*)**, in the **Encoding** list, click **ANSI**, and then click **Save**.

You may wonder why another format, other than ANSI is not used here. UDA is a low level component; and this format has been traditionally used. As an experiment, you could try to use different encoding types (UTF-8, Unicode) and determine if alternate types work with UDA. Do this only after you have successfully completed this workshop, and if you have the time to do this.

- Close **Notepad** without saving changes.
- In **Eclipse**, close the **uda_trace.<PID #>.log** tab without saving changes, and then close **Eclipse**.

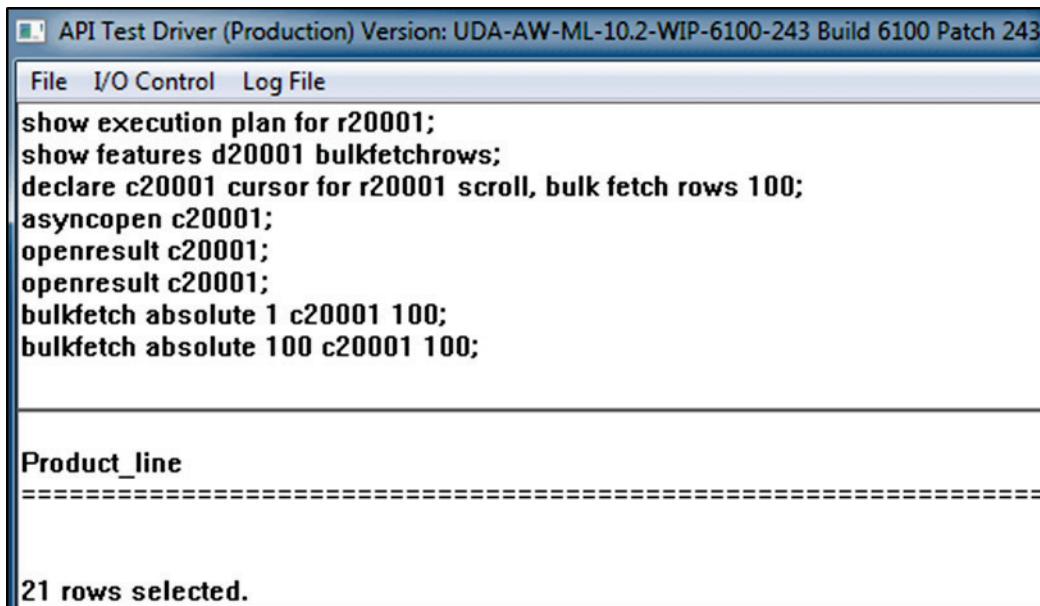
Task 5. Run udatest.exe and test the native SQL.

- Navigate to **C:\Program Files\IBM\cognos\c10_64full\bin64** and then double-click **udatest.exe**.
- From the **I/O Control** menu, click **Read from**, and then open **Test.api**.

The UDATest utility executes the test.api file and produces a log file. In this case, test.log will be created in the bin folder.

The results in the UDATest utility indicate that 21 rows were selected.

The results appear as follows:



- In the **C:\Program Files\IBM\cognos\c10_64full\bin64** folder open the **TEST.log** file in **Eclipse**.

The log file contains the original query, the native SQL, the execution plan and the data that is returned from the query.

- Find **native SQL**.

This section of the file contains the native SQL that is sent to the database. Notice that most of the code appears on one line (displayed below with word wrap).

```
SQL> show native sql for r20001;
```

```
Native SQL statements ( 1 ) :
with "PRODUCT_NAME_LOOKUP7" as (select "PRODUCT_NAME_LOOKUP"."PRODUCT_NUMBER"
"PRODUCT_NUMBER" , "PRODUCT_NAME_LOOKUP"."PRODUCT_NAME" "PRODUCT_NAME" ,
"PRODUCT_NAME_LOOKUP"."PRODUCT_DESCRIPTION" "PRODUCT_DESCRIPTION" from
"GOSALES"."PRODUCT_NAME_LOOKUP" "PRODUCT_NAME_LOOKUP" where
"PRODUCT_NAME_LOOKUP"."PRODUCT_LANGUAGE" = 'EN'), "Products" as (select
"PRODUCT_LINE"."PRODUCT_LINE_EN" "Product_line" ,
"PRODUCT_TYPE"."PRODUCT_TYPE_EN" "Product_type" , "PRODUCT6"."PRODUCT_NUMBER"
"Product_number" from "GOSALES"."PRODUCT_LINE" "PRODUCT_LINE",
"GOSALES"."PRODUCT_TYPE" "PRODUCT_TYPE", "GOSALES"."PRODUCT" "PRODUCT6",
"PRODUCT_NAME_LOOKUP7", "GOSALES"."PRODUCT_SIZE_LOOKUP" "PRODUCT_SIZE_LOOKUP",
"GOSALES"."PRODUCT_BRAND" "PRODUCT_BRAND", "GOSALES"."PRODUCT_COLOR_LOOKUP"
"PRODUCT_COLOR_LOOKUP" where "PRODUCT_LINE"."PRODUCT_LINE_CODE" =
"PRODUCT_TYPE"."PRODUCT_LINE_CODE" and "PRODUCT_TYPE"."PRODUCT_TYPE_CODE" =
"PRODUCT6"."PRODUCT_TYPE_CODE" and "PRODUCT6"."PRODUCT_NUMBER" =
"PRODUCT_NAME_LOOKUP7"."PRODUCT_NUMBER" and
"PRODUCT_COLOR_LOOKUP"."PRODUCT_COLOR_CODE" = "PRODUCT6"."PRODUCT_COLOR_CODE"
and "PRODUCT_SIZE_LOOKUP"."PRODUCT_SIZE_CODE" = "PRODUCT6"."PRODUCT_SIZE_CODE"
and "PRODUCT_BRAND"."PRODUCT_BRAND_CODE" = "PRODUCT6"."PRODUCT_BRAND_CODE"),
"Sales" as (select "ORDER_DETAILS"."PRODUCT_NUMBER" "Product_number" ,
"ORDER_DETAILS"."QUANTITY" * "ORDER_DETAILS"."UNIT_COST" "Product_cost" from
"GOSALES"."ORDER_DETAILS" "ORDER_DETAILS", "GOSALES"."ORDER_HEADER"
"ORDER_HEADER" where "ORDER_HEADER"."ORDER_NUMBER" =
"ORDER_DETAILS"."ORDER_NUMBER") select "Products"."Product_line" "Product_line"
, "Products"."Product_type" "Product_type" , sum("Sales"."Product_cost")
"Product_cost" from "Products", "Sales" where "Products"."Product_number" =
"Sales"."Product_number" group by "Products"."Product_line",
"Products"."Product_type" order by "Product_line" asc FOR FETCH ONLY
```

- Copy the SQL from **with** to the end of **asc** to the clipboard.

The following is selected:

```
with "PRODUCT_NAME_LOOKUP7" as (select "PRODUCT_NAME_LOOKUP"."PRODUCT_NUMBER"
"PRODUCT_NUMBER" , "PRODUCT_NAME_LOOKUP"."PRODUCT_NAME" "PRODUCT_NAME" ,
"PRODUCT_NAME_LOOKUP"."PRODUCT_DESCRIPTION" "PRODUCT_DESCRIPTION" from
"GOSALES"."PRODUCT_NAME_LOOKUP" "PRODUCT_NAME_LOOKUP" where
"PRODUCT_NAME_LOOKUP"."PRODUCT_LANGUAGE" = 'EN'), "Products" as (select
"PRODUCT_LINE"."PRODUCT_LINE_EN" "Product_line" ,
"PRODUCT_TYPE"."PRODUCT_TYPE_EN" "Product_type" , "PRODUCT6"."PRODUCT_NUMBER"
"Product_number" from "GOSALES"."PRODUCT_LINE" "PRODUCT_LINE",
"GOSALES"."PRODUCT_TYPE" "PRODUCT_TYPE", "GOSALES"."PRODUCT" "PRODUCT6",
"PRODUCT_NAME_LOOKUP7", "GOSALES"."PRODUCT_SIZE_LOOKUP" "PRODUCT_SIZE_LOOKUP",
"GOSALES"."PRODUCT_BRAND" "PRODUCT_BRAND", "GOSALES"."PRODUCT_COLOR_LOOKUP"
"PRODUCT_COLOR_LOOKUP" where "PRODUCT_LINE"."PRODUCT_LINE_CODE" =
"PRODUCT_TYPE"."PRODUCT_LINE_CODE" and "PRODUCT_TYPE"."PRODUCT_TYPE_CODE" =
"PRODUCT6"."PRODUCT_TYPE_CODE" and "PRODUCT6"."PRODUCT_NUMBER" =
"PRODUCT_NAME_LOOKUP7"."PRODUCT_NUMBER" and
"PRODUCT_COLOR_LOOKUP"."PRODUCT_COLOR_CODE" = "PRODUCT6"."PRODUCT_COLOR_CODE"
and "PRODUCT_SIZE_LOOKUP"."PRODUCT_SIZE_CODE" = "PRODUCT6"."PRODUCT_SIZE_CODE"
and "PRODUCT_BRAND"."PRODUCT_BRAND_CODE" = "PRODUCT6"."PRODUCT_BRAND_CODE"),
"Sales" as (select "ORDER_DETAILS"."PRODUCT_NUMBER" "Product_number" ,
"ORDER_DETAILS"."QUANTITY" * "ORDER_DETAILS"."UNIT_COST" "Product_cost" from
"GOSALES"."ORDER_DETAILS" "ORDER_DETAILS", "GOSALES"."ORDER_HEADER"
"ORDER_HEADER" where "ORDER_HEADER"."ORDER_NUMBER" =
"ORDER_DETAILS"."ORDER_NUMBER") select "Products"."Product_line" "Product_line"
, "Products"."Product_type" "Product_type" , sum("Sales"."Product_cost")
"Product_cost" from "Products", "Sales" where "Products"."Product_number" =
"Sales"."Product_number" group by "Products"."Product_line",
"Products"."Product_type" order by "Product_line" asc
```

- From the **Start** menu, navigate to **All Programs\IBM Data Studio**, and then click **Data Studio 4.1.0.0 Client**.

You need to submit this query to the vendor database vendor query tool for the specific database you are referencing; in your environment, the database is DB2.


- In the **Administration Explorer** pane on the left side, expand **localhost** and **DB2**.

- Click **New SQL Script** .

- Click the **No Connection** link, click **GS_DB**, and then click **Finish**.

Once you have connected to a database as an authorized user, you can then issue SQL statements or DB2 commands against that database.

- If you are prompted to enter a password, type **Education1** in the **Password** box, select the **Save password** check box, and then click **OK**.

- Paste the clipboard contents into the middle pane, click **OK** to close the **Statement Terminator** dialog box, and then click **Run SQL** .

The query runs and the **Result1** tab in the bottom right corner displays the output of 21 rows.

The results appear as follows:

Status	Result1	
Product_line	Product_type	Product_cost
Camping Equipment	Cooking Gear	167128146.52
Camping Equipment	Lanterns	72808366.01
Camping Equipment	Packs	213232893.15
Camping Equipment	Sleeping Bags	188159844.17
Camping Equipment	Tents	360908320.53
Golf Equipment	Golf Accessories	19927608.85
Golf Equipment	Irons	135602473.09
Golf Equipment	Putters	55499773.14
Golf Equipment	Woods	163187870.64
Mountaineering Equipment	Climbing Accessories	39594854.17
Mountaineering Equipment	Rope	78380688.26
Mountaineering Equipment	Safety	52250948.68
Mountaineering Equipment	Tools	76157733.00
Outdoor Protection	First Aid	6428902.92
Outdoor Protection	Insect Repellents	12631783.58
Outdoor Protection	Sunscreen	10950326.97
Personal Accessories	Binoculars	78645643.41
Personal Accessories	Eyewear	514880568.49
Personal Accessories	Knives	95092176.77
Personal Accessories	Navigation	129008368.71
Personal Accessories	Watches	291463483.11
Total 21 records shown		

You have checked the performance of the query at the database level, to see if the same values are returned as those seen in the report in IBM Cognos Viewer. In both environments, 21 rows were returned.

- Return to the **TEST.log** file in **Eclipse**, and locate the **execution plan** section.

- Examine the fetch, prepare, execution and CPU times by locating the following section.

```
<operator CPUTime="0.000000" elapsedTime="0.000000" nRows="0"
operatorType="dbScan" rowSize="448" totalCPUTime="0.000000"
totalElapsedTime="0.000000">
```

In this scenario, the execution time of the query was minimal, and there were no issues. This workshop provides you with a method to get this information; there is nothing to troubleshoot here, but you were able to confirm that the SQL results returned were the same in the IBM Cognos query as they were in the direct SQL query against the database.

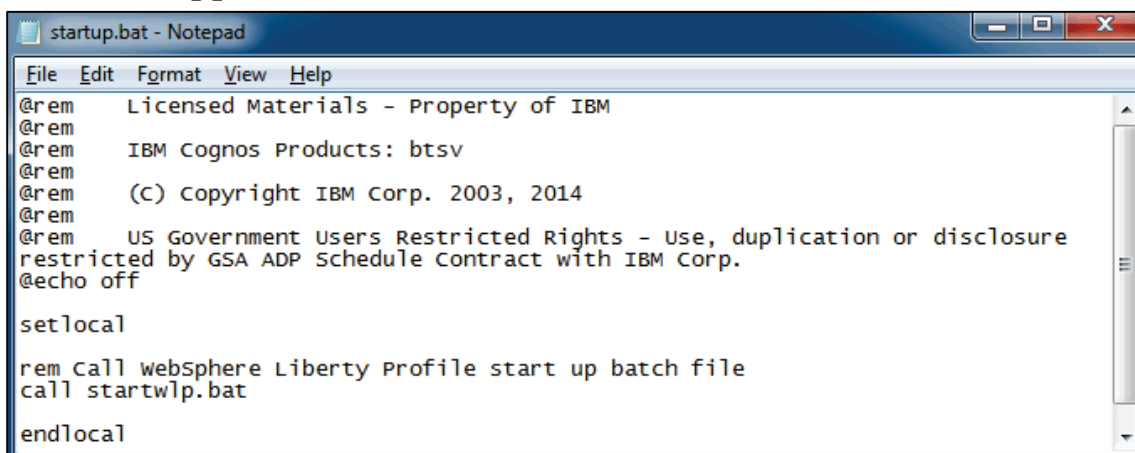
This is an example of a low level development trace. You could also test the SQL going through Cognos SQL, as opposed to the native SQL. Does it take longer to execute?

If you do comparisons with earlier versions of IBM Cognos, the SQL may not be the same. This does not necessarily indicate an issue

Task 6. Restore the settings and files to their original state.

- Close all open windows without saving.
- In **Windows Explorer**, navigate to **C:\Program Files\IBM\cognos\c10_64full\bin64**, and then delete **TEST.log** and **Test.api**.
- Edit **startup.bat** for both dispatcher installations (**C:\Program Files\IBM\cognos\c10_64full\bin64** and **C:\Program Files\IBM\cognos\c10_64DispCM\bin64**) to remove the setting of the environment variables.

The result appears as follows:



```
startup.bat - Notepad
File Edit Format View Help
@rem Licensed Materials - Property of IBM
@rem
@rem IBM Cognos Products: btsv
@rem
@rem (C) Copyright IBM Corp. 2003, 2014
@rem
@rem US Government Users Restricted Rights - Use, duplication or disclosure
restricted by GSA ADP Schedule Contract with IBM Corp.
@echo off

setlocal

rem Call webprofile Liberty Profile start up batch file
call startwlp.bat

endlocal
```

- Navigate to **C:\Program Files\IBM\cognos\c10_64full\configuration**, and delete **CQEconfig.xml**.
- Close all open applications.

Leave the IBM Cognos Full:9315 service and the IBM Cognos DispCM:9320 service stopped for the next workshop.

What is Perf.QFS?

- Purpose:
 - use to determine report running time allocation
- Description:
 - captures data within query framework (QFW) code
 - captures gesture and action times from studios

© 2015 IBM Corporation



If your report is slow, use Perf.QFS to determine where time is being spent when running a report. There may be external factors involved such as database loads, network latency, overloaded servers, and so on, but this will isolate the IBM Cognos BI components involved.

Perf.QFS provides a method to capture data within the QFW code, and is independent of the product or test tool that consumes the QFW code.

Perf.QFS allows gesture or action times from studios to be captured, rather than only the overall request run timing as in QFWTest.

Activate Perf.QFS

- Activate:
 - modify ipfPERFclientconfig.xml.sample and save as ipfclientconfig.xml

```
<category name="Perf.QFS" class="com.cognos.indications.LogTypedLogger">  
    <level value="debug"/>  
    <appender-ref ref="clientFlatFile"/>  
</category>
```

© 2015 IBM Corporation



To setup Perf.QFS, edit the ipfPERFclientconfig.xml.sample file, in the section below the string `<!--` QFS `-->`.

Describe Perf.QFS Output

- Output:
 - ..\logs directory in a file prefixed with performance
 - information by report and component

```
<RP n="Add Color" rN="0" st="2015-03-24T08:56:38.977" et="2015-03-24T09:15:56.609" status="SUCCESS">
  <RC><![CDATA[/content/package[@name='GO
Sales(Query)']/folder[@name='Documentation Report
Samples']/report[@name='Add Color'];asynchRun_Request;]]></RC>
  <RS et="6586">
    <C n="MDOperationProvider" et="227"></C>
    <C n="RelationalQueryProvider" et="317"></C>
    <C n="CoordinationPlanner" et="5522"></C>
```

© 2015 IBM Corporation



The information in the file will be presented by report and then by component.

The XML file can be analyzed as is, or it can be transformed into something more readable by applying a transformation to the XML output, such as perf.xslt, which will be used in the workshop on this topic.

The meaning of the XML elements are described at the top of the output file. The file may not be written immediately, as it depends on the flushing of master dataset. In most cases, running more than one report or running the same report twice will result in the file being created in the ..\logs directory.

perf.xslt will identify the areas that took more than 15 seconds in red; you can change the 15 seconds setting by editing perf.xslt. The code for perf.xslt has been provided for your reference at the end of this module, if you are interested in using such a file in your own environment.

Workshop 2: Perform Perf.QFS Logging

You need to identify the architectural components that are used to process a run report request and how much time spent in each component. This will help you with troubleshooting issues should they occur.

You will:

- Assign the IIS 7 IUSR user to have Write permission to the logs directory for the c10_64DispCM instance.
- Set the logging level to info for Perf.QFS logging in the ipfclientconfig.xml file for the c10_64DispCM instance.
- Using admin/Education1 credentials, create and save a list report and a crosstab report in Report Studio, and then execute each report twice.
- Stop the services and open the perf.xml file and briefly examine the contents. You can modify the xml file to use the perf.xslt stylesheet at C:\Edcognos\B5A19\08-Explore_Component_Logging\.
- Compare the components that each report used and compare the results with the IBM Cognos 10.2.2 Architecture diagram. Why is there an OlapQueryProvider as a component for the cross tab report?
- What component is the crosstab report spending most of its time in? Why?
- What component function is the most time spent in? Why?

For more information about where to work and the workshop results, refer to the Tasks and Results section that follows. If you need more information to complete a task, refer to earlier demos for detailed steps.

Workshop 2: Tasks and Results

At the beginning of this workshop, no dispatcher is running.

Task 1. Allow IIS 7 write permission to the logs directory.

In this environment, the Web server user (IUSR) requires permission to write in the `..\logs` directory. Before proceeding with the trace, you will enable this permission for the C10_64DispCM instance. You may be familiar with this process, as it has been done earlier, for the C10_64Full Dispatcher instance.

- In **Windows Explorer**, navigate to **C:\Program Files\IBM\cognos\c10_64DispCM**, right-click the **logs** directory, and then click **Properties**.
- On the **Security** tab, to change permissions, click **Edit**.
- Click **Add**.
- In the **Enter the object names to select (examples)** pane, type **IUSR**, and then click **Check Names**.

If the user is found, it will appear with an underscore in the pane.

- Click **OK**, and then in the **Group or user names** pane, click **IUSR**.
- In the **Permissions for IUSR** pane, scroll to the **Write** permission, and in the **Allow** column, click the **Write** check box to select it.
- Click **Apply**, and then click **OK** to close the **Permissions for logs** dialog box.
- Click **OK** to close the **logs Properties** dialog box, and then close **Windows Explorer**.

Task 2. Set the logging level to INFO for Perf.QFS logging in the ipfclientconfig.xml file.

- From the **Taskbar**, launch **Services**, and then ensure that the **IBM Cognos DispCM:9320** service is stopped.
- Ensure that the following services are started:
 - Apache Directory Server - default
 - DB2-DB2COPY1 - DB2
 - DB2 Remote Command Server (DB2COPY1)
 - DB2DAS - DB2DAS00
 - Lotus Domino Server (CProgramFilesx86IBMLotusDominodata)
 - World Wide Web Publishing Service
- Start the **IBM Cognos Full:9315** service.
- In **Windows Explorer**, navigate to **C:\Program Files\IBM\cognos\c10_64DispCM\configuration** and copy **ipfPERFclientconfig.xml.sample** to the same directory.

The DispCM instance is the instance in the server groups that runs the 32-bit ReportServer service queries. The Full instance is the instance in the server groups that runs the 64-bit ReportServer service queries, and therefore does not require this type of tracing.

- Rename **ipfPERFclientconfig.xml - Copy.sample** to **ipfclientconfig.xml**, and then open this **ipfclientconfig.xml** file in **Eclipse**.

Ensure that you do not keep PERF in the filename when you rename the file. If necessary, close any open tabs in the middle pane before opening the ipfclientconfig.xml file.

- Maximize the **ipfclientconfig.xml** tab, and then on the **Source** tab, locate the following section:

```
<!--category name="Perf.QFS" class="com.cognos.indications.
LogTypedLogger">
    <level value="info"/>
</category-->
```

Do not use the level value="warn" section.

- Replace this section with the following:

```
<category name="Perf.QFS" class="com.cognos.indications.
LogTypedLogger">
<level value="debug"/>
<appender-ref ref="clientFlatFile"/>
</category>
```

Hint: You can copy and paste the text from C:\Edcognos\B5A19\08-Explore_Component_Logging\Mod 8_Wkshp 2_Task 1.txt.

You have uncommented the section, by removing the `!`, the `--`, and have modified the setting of info to debug, and added an appender-ref setting. The results appear as follows:

```
*ipfclientconfig.xml
<!-- -->
<category name="Perf.QFS" class="com.cognos.indications.LogTypedLogger">
  <level value="debug"/>
  <appender-ref ref="clientFlatFile"/>
</category>
```

- Save the file, close the `ipfclientconfig.xml` tab, and then close **Eclipse**.

Task 3. Create a list report and a crosstab report in Report Studio and then execute the reports.

- Navigate to **C:\Program Files\IBM\cognos\c10_64DispCM\logs**, and then delete the files at the root of this folder.
Do not delete the XQE folder if there is one.
- Switch to the **Services** window, start the **IBM Cognos DispCM:9320** service, and wait for it to fully start before proceeding to the next step.
- Launch **Internet Explorer**, go to **http://vclassbase:88/C10Full**, and then log on to the **LDAP_Dev** namespace with **admin/Education1** credentials.

- Launch **Report Studio** using the **Public Folders\Samples\Models\GO Sales (query)** package, and then create a list report and populate it from the **Sales (query)** namespace with the following items:

- **Products: Product line, Product type**
- **Sales: Product cost**

Product line	Product type	Product cost
<Product line>	<Product type>	<Product cost>
<Product line>	<Product type>	<Product cost>
<Product line>	<Product type>	<Product cost>

- Do not run the report at this time, but save the report as **perf.qfs_list** in the **GO Sales (query)** folder.
- Using the same **GO Sales (query)** package, create a **Crosstab** report (**File** menu**New**) and populate it from the **Sales (query)** namespace with the following items:
 - Columns: **Time: Year**
 - Rows: **Order method: Order method type**
 - Measures: **Sales: Quantity**

Quantity	<#Year#>	<#Year#>
<#Order method type#>	<#1234#>	<#1234#>
<#Order method type#>	<#1234#>	<#1234#>

- Do not run the report at this time, but save the report as **perf.qfs_crosstab** in the **GO Sales (query)** folder.
- Close **Report Studio**, and then navigate to **IBM Cognos Connection**.

- From **Public Folders\Samples\Models\GO Sales (query)**, run the **perf.qfs_list** report, on the toolbar click **Return**, run the same report again, and then click **Return**.

You may have to refresh the view in IBM Cognos Connection to see the new entries.

- Repeat to run the **perf.qfs_crosstab** report two times, and then close the browser window.

What do you notice about the run times after the first execution of the report?

Task 4. Modify the Performance_<timestamp>.xml file so that its contents can be examined in a stylesheet.

- In **Windows Explorer**, navigate to **C:\Program Files\IBM\cognos\c10_64DispCM\logs** to identify the presence of the **Performance_<timestamp>.xml** file.
- Stop the **IBM Cognos DispCM:9320** service and the **IBM Cognos Full:9315** service, and then open the **Performance_<timestamp>.xml** file in **Eclipse**.

At the top of the **Performance_<timestamp>.xml** file you'll find some descriptions of the elements (for example, CF=Component Function), and their attributes (for example, nc=NumCalls, rN=RunNumber, and so forth).

- Add the following as the second line:

```
<?xml-stylesheet type="text/xsl" href="C:\Edcognos\B5A19\08-Explore_Component_Logging\perf.xslt"?>
```

The **perf.xslt** code has been provided at the end of this workshop.
- Add the following as the last line (if necessary), **</PerformanceData>**.
- From the **File** menu, click **Save As**, navigate to **C:\Edcognos\B5A19\08-Explore_Component_Logging**.
- In the **Save as type** list, click ***.***, in the **File name** box, type **perf.xml**, and then click **Save**.

- Close the **perf.xml** tab, and then close **Eclipse**.
- In **Windows Explorer**, navigate to **C:\Edcognos\B5219\08-Explore_Component_Logging**, and then open **perf.xml** in Internet Explorer.

The result appears similar to the following, note that you may have to scroll through the results to see the perf.qfs_list and perf.qfs_crosstab outputs:

Report: perf.qfs_list	
Execution Start Time	2015-03-24T17:03:35.941
Execution End Time	2015-03-24T17:04:15.769
Status	SUCCESS
Request Summary	30712
QECL	396
CM_GetModelConnectInfo (1)	34
CM_LoadParameterMapFromCM (1)	0
CM_GetDatabaseAccessInfo (2)	361
RelationalQueryProvider	2064
QFSQuery::Prepare (1)	22

The file opens based on the formatting used in the perf.xlst file. This makes the file easier to read.

The numbers shown in the right column (such as 396) are milliseconds. Numbers in parentheses shown in the left column represent the number of calls. These values come directly from the original, unformatted Performance_<timestamp>.xml log file. You can refer to Performance_<timestamp>.xml to understand the values being presented. Near the top of this file is an explanation of the abbreviations for results.

- Review the information in the file, and answer the following:
 - Compare the components that each report used. Are there any differences?
 - Identify the components that spent the most time processing the request.
 - Compare the components used with the IBM_Cognos_10.2.2_architecture_diagram.jpg file located at C:\Edcognos\B5A19\01-Intro_and_SOA.
 - Why is there an OlapQueryProvider as a component for the crosstab report?
- Close **Internet Explorer**, and then delete **perf.xml** from C:\Edcognos\B5A19\08-Explore_Component_Logging.
- In **Windows Explorer**, delete the files at the root of C:\Program Files\IBM\cognos\c10_64DispCM\logs.
- Delete **ipfclientconfig.xml** from C:\Program Files\IBM\cognos\c10_64DispCM\configuration.
- To prepare for the first demo in the next module, start the **IBM Cognos Full:9315** service, and then start the **IBM Cognos DispCM:9320** service.
- Close all open windows.

Additional information for workshop: perf.xslt code

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright (C) 2007 Cognos Incorporated. All Rights Reserved. Cognos (R) is a trademark of Cognos
Incorporated. -->
<!-- Cognos and the Cognos logo are trademarks of Cognos Incorporated. -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" encoding="UTF-8"/>
  <xsl:variable name="result" select="/PerformanceData"/>
  <!--Match document root-->
  <xsl:template match="/">
    <html>
      <head>
        <title>Performance</title>
        <meta http-equiv="pragma" content="no-cache"/>
        <meta http-equiv="cache" content="0"/>
      </head>
      <body bgcolor="#ffffff">
        <font size="5" face="arial">Performance Information</font>
        <!--Match all elements under 'PerformanceData' -->
        <xsl:apply-templates select="$result/RP"/>
      </body>
    </html>
  </xsl:template>
  <!--Execute this block for every matching File element-->
  <xsl:template match="RP">
    <table border="1" cellpadding="10">
      <!--Output Name attribute from element File-->
      <tr>
        <td bgcolor="#8EB6F0" colspan="4">
          <font size="4" face="arial">
            <b>Report: </b>
          </font>
          <xsl:value-of select="@n"/>
        </td>
      </tr>
      <tr>
        <td bgcolor="#E3E9F3">
          <font size="2" face="arial">
            <b>Execution Start Time</b>
          </font>
        </td>
        <td bgcolor="#E3E9F3">
          <xsl:value-of select="@st"/>
        </td>
      </tr>
      <tr>
        <td bgcolor="#E3E9F3">
          <font size="2" face="arial">
            <b>Execution End Time</b>
          </font>
        </td>
        <td bgcolor="#E3E9F3">
          <xsl:value-of select="@et"/>
        </td>
      </tr>
      <tr>
        <td bgcolor="#E3E9F3">
          <font size="2" face="arial">
            <b>Status</b>
          </font>
        </td>
        <td bgcolor="#E3E9F3">
          <xsl:value-of select="@status"/>
        </td>
      </tr>
      <tr>
        <td bgcolor="#E3E9F3">
          <font size="2" face="arial">
            <b>Status</b>
          </font>
        </td>
        <td bgcolor="#E3E9F3">
          <xsl:value-of select="@status"/>
        </td>
      </tr>
    </table>
    <!--Match all Customer elements under the RS element-->
    <xsl:apply-templates select="RS"/>
    <xsl:apply-templates select="RQ"/>
  </xsl:template>
  <!--Execute this block for every matching RS element-->
```

```

<xsl:template match="RS">
  <tr>
    <td bgcolor="#BFD2E2">
      <font size="3" face="arial">
        <b>Request Summary </b>
      </font>
    </td>
    <td bgcolor="#BFD2E2">
      <xsl:value-of select="@et"/>
    </td>
  </tr>
  <xsl:apply-templates select="C"/>
</xsl:template>
<xsl:template match="RQ">
  <tr>
    <td bgcolor="#BFD2E2">
      <font size="3" face="arial">
        <b>Request Query: </b>
        <xsl:value-of select="@n"/>
      </font>
    </td>
    <td bgcolor="#BFD2E2">
      <xsl:value-of select="@et"/>
    </td>
  </tr>
  <xsl:apply-templates select="C"/>
</xsl:template>
<xsl:template match="C">
  <tr>
    <td bgcolor="#E3E9F3">
      <font size="2" face="arial">
        <xsl:value-of select="@n"/>
      </font>
    </td>
    <td bgcolor="#E3E9F3">
      <!--Highlight name when elapse time exceeds n milliseconds'-->
      <xsl:if test="@et > 15000">
        <xsl:attribute name="bgcolor">#FF0000</xsl:attribute>
      </xsl:if>
      <font size="2" face="arial">
        <xsl:value-of select="@et"/>
      </font>
    </td>
  </tr>
  <xsl:apply-templates select="CF"/>
</xsl:template>
<xsl:template match="CF">
  <tr>
    <td bgcolor="#E3E9F3">
      <font size="2" face="arial">
        &#160;&#160;&#160;&#160;
        <xsl:value-of select="@n"/>
        ( <xsl:value-of select="@nc"/> )
      </font>
    </td>
    <td bgcolor="#E3E9F3">
      <!--Highlight name when elapse time exceeds n milliseconds'-->
      <xsl:if test="@et > 15000">
        <xsl:attribute name="bgcolor">#FF0000</xsl:attribute>
      </xsl:if>
      <font size="2" face="arial">
        <xsl:value-of select="@et"/>
      </font>
    </td>
  </tr>
</xsl:template>
</xsl:stylesheet>

```

Summary

- At the end of this module, you should be able to:
 - explore component logging for Gateway, Dispatcher, Report Server, and Universal Data Access layer

