

IBM Cúram Social Program Management
Version 7.0.1

*Cúram Business Intelligence Reporting
Developer Guide*



Note

Before using this information and the product it supports, read the information in “Notices” on page 49

Edition

This edition applies to IBM Cúram Social Program Management v7.0.1 and to all subsequent releases unless otherwise indicated in new editions.

Licensed Materials - Property of IBM.

© **Copyright IBM Corporation 2012, 2017.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

© Cúram Software Limited. 2011. All rights reserved.

Contents

Figures	v
--------------------------	----------

Tables	vii
-------------------------	------------

Developing Business Intelligence

Reports	1
--------------------------	----------

Introduction	1
Objective	1
Intended Audience	1
Prerequisites	1
Introduction to Reporting	1
Objective	1
What is Data Warehousing.	1
Data Warehousing	2
BIRT Reporting Tool	2
Changed Data Capture	2
System Overview	2
DB2 Environment.	2
(Deprecated) Oracle Environment	3
Runtime Architecture	3
Introduction	3
Architecture Overview	4
Runtime Architecture	4
Development Process And Modelling	4
Directory Structure & Artifacts	6
<i>bin</i>	6
<i>build</i>	6
<i>logs</i>	6
The Reporting Environment Explained.	6
Overview	6
Initial Setup	6
ETL explained	7
(deprecated)Metadata explained	7
The Build Script	8
The Build Script explained.	8
Using the Build script	9
Running the build script	10
Change Data Capture	10
Sequence of running ETLs	11
Performance Tuning and Optimization	12
Creating custom Java transformations	12
Summary	13
Installation and Configuration	13
Overview	13
Ant setup	13
Java Platform, Standard Edition installation	14
Other Environment Variables	14
Common	14
DB2	15
(deprecated) Oracle.	15
Install the BIA Reporting Module	15
Setup BIApplication.Properties and BIBootstrap.properties files	15
Executing BIRT Reports against Demo-Data	16
DB2 Environment	16

Install DB2 Database and IBM DB2 Warehouse	16
Deploying DB2 Warehouse	17
DB2 BIApplication.properties, BIBootstrap.properties and Environment Variable Setup	17
DB2 Bootstrap Your Environment	17
Create DB2 target schema's	18
Test Configuration	19
Create Data Warehouse Databases	19
Set Up and Configure BIA Project in DB2 Warehouse.	20
Merging DB2 Warehouse Projects	21
Running DB2 Warehouse Control Flows	21
Customizations and Upgrades	22
Customizations	22
Upgrade Process.	22
Troubleshooting	23
Build fail errors when running the build script	23
'build configtest' is failing	23
Errors with permissions	25
The lastwritten field in source table is not populated	25
Error when running 'build transform.aggcasemonth'.	25
Cognos does not start	25
No Data in Tables after running owb.environment.tests.run	26
Error reimporting due to Matching Conflicts	26
Incorrect language code set	26
Import process running slowly	27
(deprecated) Error deploying the locators in Oracle because of Oracle version	27
(deprecated) Unwanted OWB Locations	28
(deprecated) Errors while importing the ETLs	29
(deprecated) Unable to log into Control Center	29
(deprecated) ETL cannot deploy due to incompatible database version	29
(deprecated) 'build database.all' is failing with JAR file error for DB2	30
Configuring the BIBootstrap.properties file.	30
Sample BIBootstrap property file for DB2	31
Sample BIBootstrap property file for Oracle	31
Sample BIApplication property file for Oracle	32
Configuring the BIApplication.properties file.	33
Build Script Commands.	34
Common DB2 and Oracle Commands	34
(deprecated) Scheduling of ETL Processes	36
(deprecated) DB2	36
(deprecated)Oracle	36
(deprecated)Design time platform	36
(deprecated)Runtime time platform	36
(deprecated)Create OWB Workspace Owner and Users using Repository Assistant	36
(deprecated)Remove OWB Workspace Owner and Users using the Repository Assistant	37
Granting Database Privileges	38

Capturing Changed Data	39
(deprecated)How to Add an Oracle ETL.	39
(deprecated)Known Issues	40
(deprecated)Build Environment.	40
(deprecated)Importing in OWB.	40
(deprecated)Installing Patches	40
(deprecated)Installing Oracle Patches.	40
(deprecated)Instructions for applying a once off patch	40
Globalization	41
(deprecated)Initial Oracle Schemas.	42

Security	42
(deprecated)Upgrading to Oracle 11gR2	43
(deprecated)Incremental Changed Data Extraction	44
Configuring the log4j.properties file.	44
Sample .profile property file for UNIX	45
Sample log4j property file for DB2.	47

Notices 49

Privacy Policy considerations	51
Trademarks	51

Figures

Tables

Developing Business Intelligence Reports

Use this information to learn how to develop Cúram Business Intelligence and Analytics Reporting. An overview of the architecture, design, development, and deployment of Business Intelligence and Analytics Reporting is provided.

Introduction

Objective

This document provides developers with an overview of the Cúram Business Intelligence and Analytics(BIA) Reporting development process. This includes the architecture, design, development and deployment of BIA Reporting.

Intended Audience

This guide is intended for developers who are working with BIA Reporting. It provides details on how to work within the BIA Reporting framework.

Prerequisites

A working knowledge of the database platform being used (Oracle or DB2) and ETL tool is required. The reader should also have an understanding of Data Warehousing concepts and techniques.

Introduction to Reporting

Objective

This document provides developers with a definition of warehousing and how it is implemented.

What is Data Warehousing

Data warehousing is the process of collecting data which is then organized in such a way that it can easily analyzed, extracted, reported on, and otherwise be used for the purposes of further understanding the data. This ability to understand any data can give organizations the power to make very useful management decisions.

This process consists of extracting data from one or more source Databases. This information is then transformed or cleaned up to remove all anomalies and brought into a central repository, a Data Warehouse. We then take the information from the data warehouse into Data Mart(s) which are specialized versions of the data warehouse which have been designed to suit the needs of the customer/target audience. It is then possible to view this data using a reporting tool such as Business Objects, Cognos BI in the form of graphs or charts etc. The Stages of the Data warehouse are:

1. Extracted into **Source System**
2. Transferred into **Staging** where it is transformed
3. Loaded into **Central Warehouse** where it is stored
4. Delivered into **Datamart** where it is queried

Data Warehousing

The Reporting schemas are comprised of: Staging, Central and Datamart schemas. Source database or the operational database from which we want to extract our data from.

1. Staging. The Staging ETL's are run, they pull any information from Source through to Staging
2. Central. Once staging is populated with data the Central ETL's are run. These ETL's pull data into the Central Data Warehouse changing data where necessary and applying any business logic required..
3. Datamarts. This is the final stage of the Reporting repository, data is transformed into dimensional format, and de-normalized to ensure ease of query, and to ensure cube builders/report builders find the data easier to model.

BIRT Reporting Tool

An open source reporting tool called Business Information and Reporting Tools (BIRT) is used by the source database. BIRT has been used to develop Dashboards and Reports, which display data that is stored in the Application and Data Warehouse Tables. Please see the BIRT Developer Guide for more information.

Changed Data Capture

When the Staging ETL processes are run, they rely on a lastwritten timestamp column on all source tables, this ensures that only changed data is extracted. It is important to understand that the reporting module requires that each table that it extracts data from must have a column of type timestamp and have a name of lastwritten, and that any application must update this column whenever a row is created or modified in the source database.

System Overview

DB2 Environment

This product contains components created using DB2 elements and is designed for use with IBM products, licenses for which may be obtained as necessary from IBM.

The installation and configuration section installs the environment shown below. Typically the BIA Reporting Module and DB2 Warehouse are installed on a standalone machine in a development environment. The application and associated operational database are installed on a separate machine called e.g. SourceTest. Connections are configured between the operational database and the data warehouse databases:

- A JDBC connection used by the BIA Reporting Module build environment

This environment typically represents the production environment in any project in that the data warehouse is located on separated machine(s) to the production machine(s).

Note that the Staging and Central tables are co-located in the same database called CuramDW. It is a requirement that the staging and central schemas are co-located in one DB2 database so that the Central ETL processes can be promoted and executed correctly.

The DB2 Warehouse Design Studio is the interface that provides a visual representation of the Data Warehouse. Use the Design Studio to import source objects such as tables and views, design ETL processes such as mappings, and ultimately design the target warehouse.

See the section "Customizations and Upgrades" for a description on how to make changes to Reporting and how to take on upgrades to Reporting.

(Deprecated) Oracle Environment

This product contains components created using Oracle Warehouse Builder elements and is designed for use with Oracle products, licenses for which may be obtained as necessary from Oracle.

Oracle Warehouse Builder is comprised of a set of graphical user interfaces to assist you in implementing complex data system designs. Your designs are saved as metadata in a centralized repository.

The centralized repository, known as the Warehouse Builder repository, is hosted on an Oracle Database. The Design Center is the interface that provides a visual representation of the Warehouse Builder repository. Use the Design Center to import source objects such as tables and views, design ETL processes such as mappings, and ultimately design the target warehouse.

A mapping is an object in which you define the flow of data from sources to targets. Based on a mapping design, Warehouse Builder generates the code required to implement the ETL logic. Warehouse Builder can generate PL/SQL, SQL*Loader, or ABAP code for mappings.

After you complete the design of a mapping, for example, and prompt Warehouse Builder to generate the code, the next step is to deploy the mapping. Deployment is the process of copying the relevant metadata and code you generated in the Design Center to a target schema. The target schema is generically defined as the database which will execute the ETL logic you designed in the Design Center. Specifically, in a traditional data warehousing implementation, the data warehouse is the target schema and the two terms are interchangeable.

Runtime Architecture

Introduction

This chapter provides an overview of the architecture and various components that make up BIA Reporting. Each of the artifacts provided with BIA Reporting are explained.

The BIA Reporting solution is a framework for providing specific reports. As a framework it can be customized to include new reporting requirements as they arise. The framework consists of the following deliverables:

- A set of *data models* both Relational and Dimensional. These models are optimized to support reporting needs.
- *Extract Transform and Load* (ETL) process definitions. A set of process definitions that define how the data is moved and transformed from the On-Line Transaction Processing (OLTP) data source to the BIA Reporting data sources. These are provided for both Oracle and DB2 platforms. The tools used to perform the ETL processes are Oracle Warehouse Builder (OWB) and IBM DB2 Warehouse.

Architecture Overview

Runtime Architecture

The runtime architecture for BIA reporting defines how the data flows from the transactional data source to the Reporting data sources and on to populate the reports. The data is moved from the Source database to the Staging database. From here it is moved to the Central Data Warehouse (CDW), and finally is pushed out to the Data Marts. Once the Data Marts are populated the reports are run.

Data Flow

The data is moved through the Reporting solution in a number of distinct steps.

1. **Extract to Staging Database:** The first step is the extract of the data of interest from the source database. The data is filtered on the LASTWRITTEN column in the source tables. Once the data of interest, e.g. all new entries in a particular table, is identified it is moved to the Staging area. The Staging area is a data storage area containing data from various data sources, it is essentially a copy of a subset of the source tables. This data movement is the first run of the ETL, and the data can be cleansed at this point to ensure there is no 'dirty' data that could lead to inaccurate reports.
2. **Staging to CDW:** After the required data is in the Staging area it is ready to be moved to the CDW. The CDW is the 'core' of the Reporting solution: it contains all the archived data stored in a normalized data structure. The physical location of this database is usually on the same database as the Staging database. The CDW is optimized for the efficient storage of large amounts of data. It does not prejudge how the data stored will be queried, it just stores the data that is required for analysis. It achieves this by serving all the current reporting needs and also attempting to capture the underlying business processes. Staging area data is not in the form that is required for reporting and has some gaps in it. Therefore when the data is moved from Staging to CDW any 'business logic' required is run on the data to fill in those gaps. This ensures it arrives in the CDW in a state that is useful for analysis.
3. **Load Data Marts:** Finally the data is moved from the CDW to the Data Marts, this is done by running another series of ETL processes. Data marts are de-normalized dimensional structures (Star schema). They are logical groupings of data that service a particular group of reports. ETL programs extract a subset of historical data from the CDW. This subset of data, a Data Mart, contains data that is tailored to and optimized for a specific reporting or analysis task.

An overview of the data flow between the main data sources is as follows.

1. Source Databases to Staging Area
 - Data is Cleansed
2. Staging Area to Central Data Warehouse
 - Business Logic applied from external business logic
3. Central Data Warehouse to Datamarts
 - Transformed to Dimensional

As is evident, there can be multiple data sources, only one Staging area, one CDW, and multiple Data Marts.

Development Process And Modelling

The development architecture outlines how to extend and maintain the BIA Reporting solution. The design process starts with a report or analysis requirement

and works back creating or adding to data models, once that is done the ETL processes are designed. The ETL processes specify how the data is moved through the system from OLTP to end report.

Data Models: The starting point is a reporting requirement. This requirement is formalized by drawing a logical user model. A user model captures the end user's model or view of the data required. User models help to determine the structure of the data marts. Drawing the user model involves picking out the measures or facts from the reporting requirement and the dimensions that the measure is sliced by. The user model should also identify the level of granularity required. The granularity is very important, as it determines what and how much data is captured.

Modeling the Data Marts:

In order to model the Data Mart, one must logically grouping the user models. One related group of user models will form the starting point for modeling a single Data Mart. Data Marts are defined as dimensional Star Schemas. The Data Marts contain the measures and the dimensions that are of interest. A measure is usually a numerical 'fact', e.g. 'sales', and dimensions are a way of narrowing the view of this fact, e.g. 'sales for product x in the month of June'. It should be possible to trace each item on a user model to a column in the Data Mart tables. The Data Mart must also provide the granularity specified in the user models. The datamarts are de-normalized and this makes querying them easier.

Models in the Data Mart conceptual model are engineered as star schema structure where one fact table references any number of dimension tables.

Modeling the CDW:

The next step is to design the CDW. The CDW is the main data storage area. This supports the data needs of the various Data Marts and also captures the underlying business processes.

The CDW model is engineered a normalized Entity-Relationship structure. It contains the lowest level of granularity that may be of interest and is a model of the business processes from a reporting viewpoint.

The following model is part of the CDW conceptual model, which is further discussed in the '*Directory Structure & Artifacts*' section.

Modeling the Staging Database:

The final data model is the Staging area. This is where the data of interest from the source OLTP data source is stored. It is derived by mapping the CDW to the source system. It contains a copy of every table of interest from the source system.

ETL Process Design: The ETL defines what data is taken from one database, how it is manipulated, and where it is stored. The aim is to have all data required for reports available in the Data Marts. Any required data which is not already in the Data Mart needs to be added to the Data Mart and may also need to be added to the CDW if not already there. When this happens data models are updated and the ETL processes are extended. The idea is to work back through the Reporting data sources until the required data is found. This change process is described further in the chapter '*Customizations and Upgrades*'.

Java Code: The BIA Reporting solution provides both Java transformation code and Java Connectivity code. The Java transformation code is used when the type of transformation required is not supported by the ETL tool. Java connectivity code is used to establish connections to different databases.

A framework exists for the java code and this framework contains a BIA Reporting connectivity library and BIA Reporting transformations used in any ETL processes. The directories containing the java code are described in the section '*Directory Structure & Artifacts*'.

This java code is extendable by developers of BIA Reporting. When a developer needs to create a custom transformation for an ETL (e.g. to apply some business logic to change data from the Staging to CDW) then a new transformation can be created. The transformation can then be called from within the ETL.

Directory Structure & Artifacts

In this section each of the artifacts and the directory structure provided with BIA Reporting is explained.

bin

The *bin* folder contains those files that are generated automatically from the data definition language (DDL) metadata during a build process, with the files for each script being held in its own appropriately named directory mirroring the *core/ddl* folder.

build

The *build* folder contains those files that are generated automatically from the java code during a build process, with each file being held in its own appropriately named directory mirroring the source folder. It will also contain the *data_manager.csv* static data files.

logs

The *logs* directory is the default directory to which log files are written.

The Reporting Environment Explained

Overview

This chapter describes the *environment, setup and execution of runtime process* for BIA Reporting. An assumption is made that the developer knows how to install and use the server and warehouse toolset for the database platform they are running Reporting on. What is outlined in this chapter is how to use these tools (once installed and configured) to get Reporting up and running. It is hoped that the developer also gets a clear understanding of why some of the design choices have been made in the BIA Reporting solution. This chapter is best read along with the following chapter on Customizations and Upgrades.

Initial Setup

This section briefly describes the steps required to install and setup the reporting solution. An assumption is made that the developer knows how to setup the tools for the database platform they want to run BIA Reporting on.

Please see the supported prerequisites in the Release Notes to see which **database platform versions** are supported by BIA Reporting.

To setup the database and ETL tools for Oracle and DB2 please refer to vendor documentation and Chapter 6 which details the following:

- Ant
- Enterprise Database Server (either Oracle or DB2)
- Warehouse development environment setup and initialization
- J2SE installation

ETL explained

Before explaining how the reporting solution works it is important to define an ETL and explain its function in BIA Reporting. ETL is short for *Extract, Transform* and *Load*: three functions that are combined to pull data from a source and place it in a destination database:

- **Extract** : the process of reading data from a source.
- **Transform** : the process of converting the extracted data from its previous form into the form it needs to be in so that it can be placed into another database. Transformation occurs by using business rules or lookup tables or by combining the data with other data.
- **Load** : the process of writing the data into the target database.

BIA Reporting uses ETL's to move and transform data from the On-Line Transaction Processing (OLTP) data source to the BIA Reporting data sources. As explained in the Overview the data is moved from Source database to the Staging data storage, on to the Central Data Warehouse (CDW), and then is pushed out to the Data Marts.

For the population of each of these tables in each database we will need a separate ETL. The following example will help explain why we need these ETL's:

A number of the Participant reports require Person information where the status is 'Active' between two dates. To obtain this information from the Person table in the source database and populate the datamarts the following ETL's are needed:

- An ETL that will take all the data from the source *PERSON* table and populate the Staging database *S_PERSON* table
- An ETL that will take the required data from the staging *S_PERSON* table and populate the *DW_PERSONHISTORY* table in the CDW. Some transformations will need to be performed during the population in the CDW as we need to keep a complete history of the Person so we know what their status is at a particular point in time.
- An ETL that will take the required data from the Central *DW_PERSONHISTORY* table and populate the *DM_FACTPERSONHISTORY* fact table and some dimension tables (e.g. gender ETL to populate the gender dimension) in the Datamarts database. This datamart contains data that is tailored to and optimized for running reports against.

From the above example we can see that we will need a minimum of four separate ETL's to 'push' the necessary person data from source tables to a dimensional solution that the reports can run against.

(deprecated)Metadata explained

Metadata is data that describes data and other structures, such as database objects, business rules, and processes. Each of the ETL's provided as part of the reporting solution is metadata; the database schemas are also metadata.

The metadata provided by BIA Reporting is stored on the hard disk in the folder structure described in Chapter 2. This metadata needs to be imported into the database and ETL tool. This is done by using the *build script* which is detailed in the next section.

Before detailing how to import the metadata using the build script, it is important to understand how the ETL metadata is broken into separate files and what these files are used for. This folder and file structure must be adhered to during the customization of the reporting solution also.

Within the ETL folder in the *Reporting\Components\Core\ETL* folder there are two separate folders: one for *Oracle* and one for *DB2*. The following files are contained in each of these folders:

- **database metadata files** which contain metadata for the ETL tool such as database table schema and database sequence information. These files are named after the database they contain the metadata for e.g. *central.mdl* (Oracle) or *curamdw.xml* (DB2).
- **ETL metadata files** which contains the ETL metadata for the ETL tool such as mappings and function calls. In Oracle, there is one separate file for each ETL. In DB2 there are 2. These ETL files are named after the destination table they are loading into and include the letters *ETL* at the end of the name as the build script will only load files with the name *ETL* in it e.g. *DW_PERSONHISTORY_ETL.xxxxxx*.

The reasons why the ETL metadata files are separated and not stored in one metadata file is for **concurrency**. As the ETL's can be imported or exported to or from the ETL tool to disk multiple developers can work on separate ETL's at the same time.

As already explained all BIA Reporting metadata is in the *Reporting\Components\Core\ETL* folder. With the use of the build script the metadata can be loaded into the database and ETL tool. However if the developer needs to customize this metadata the developers must copy the metadata to the *Reporting\Components\Custom* folder and make the changes here. The reasons for this are explained in the next chapter *Customizations and Upgrades*.

The Build Script

The next step is to load this metadata from disk into the database and ETL tool. This is done with the use of the build script. Once the metadata has been loaded the ETL's can be run from the ETL tool to populate the reporting databases with data.

The Build Script explained

The build script is used to drive the Reporting environment. Its main functions are to build the database schemas for Reporting and to import the ETL metadata into the ETL tools. It functions in the same manner for both supported database platforms.

In addition to importing the database schema and ETL metadata the build script validates the metadata once it is imported. Error messages detailing the problem are displayed if the build script does not successfully validate. The build script can also be used to deploy and run ETL's to populate the tables.

It is quicker to use the build script than manually importing each of the ETL's into the warehouse tool. This is because the build script can import all ETL's using one

command. Also, because the database and warehouse tool connection information is stored in a properties file the developer does not need to repeatedly import meta data files.

When customizing ETL's in the *custom* folder the build script will ensure that the customized ETL's and database schemas are being loaded and not the core metadata. Also, as the build script validates the metadata it will ensure that all objects (schema and ETL's) are consistent. It does this by attempting to validate and displaying messages if there are errors. An example of inconsistency includes a developer who changes a column name in a table in the database schema but does not change the corresponding ETL metadata. When the build script attempts to validate these objects a validation error will occur.

Using the Build script

Before using the build script it is important to set up the connection information for the databases and ETL tools. This is set up in the *BIBootstrap.properties* and *BIApplication.properties* files which are in the *Reporting\project\properties* folder (note: both of these files should be setup during the installation process but they should be checked before using the build script). When running the build script the connection information is obtained from the *BIBootstrap.properties* file.

A sample properties file is provided with the Reporting solution. To change the connection information please refer to section **Configuring the BIBootstrap.properties file**. Security details for the *BIBootstrap.properties* file is provided in the **Security** section.

Now that the databases, ETL tools, and 2 property files have been created and configured the developer can use the build script to populate the metadata.

The build script is used by calling **build.bat** in the *Reporting\components* folder. Executing build commands in the components folder will build all platform and all component artifacts. If you want to build all installed components then you must first set the component order property in the *BIApplication.properties* file.

The component.order property must be set listing the installed components, for example if you have a Cúram Child Services (CCS) component installed you must set this property accordingly component.order=core,childservices

Some examples:

To build all Cúram Enterprise Framework(CEF) and CCS artifacts the build commands must be executed from the components directory. For example if you needed to rebuild all tables, then run build database.all from the components directory.

However if you require to only build the CCS artifacts then you can run the build commands from the components\childservices directory. For example, if you only wanted to drop and re-create the CCS tables, then you would run the command build database.all from the components\childservices folder. To get a list to the build commands enter **buildhelp**.

After running this command the developer is presented with a list of commands with their description. These commands can be run by typing 'build x' where x is one of the commands that are listed and described in the **Configuring the BIApplication.properties file** section.

Running the build script

Before importing any of the metadata it is important to check that the database, ETL tool, and 2 property files have been setup correctly. Running the '**build configtest**' command checks that the environment is working correctly.

To build all the metadata (e.g. classes, schema metadata, ETL metadata) the developer can run the '**build all**' command. This command can be used to initially populate all metadata and compile all code for BIA Reporting.

Individual components within the build script can be isolated and run alone. A hierarchy exists when using the build script for building database schema and importing ETL metadata. An example of this is using the build script to import ETL metadata for the Staging database. Three options are available to the developer:

- '**build all**' command. This will build all metadata including the database schemas and the ETL metadata for all the databases. Also, in Oracle any metadata that had been imported previously is replaced (including modified metadata) in OWB.
- '**build owb.import.all**' command. This Oracle command will build all ETL metadata but will not build the database schemas. Any ETL metadata that had been imported previously is replaced (including metadata in the ETL tool which has been modified). There is no DB2 equivalent command as the metadata does not need to be imported into DB2 Warehouse.
- '**build owb.import.staging**' command. This Oracle command will build the Staging ETL metadata only. No other metadata will be replaced. There is no DB2 equivalent command as the metadata does not need to be imported into DB2 Warehouse.

The developer has the option to replace all metadata or replace isolated components. This is helpful during the customization process as the developer may not wish to replace amended ETL's in the ETL tool or amended schema in the database.

After the build runs for any command the results need to be checked. Messages will be displayed during the build run and a final message will indicate if the build ran successfully or failed ('*BUILD SUCCESSFUL*' or '*BUILD FAILED*'). Even if the build runs successfully it is important to scroll through the output checking for errors. Error and validation messages are also written to the *Reporting\logs* folder.

Change Data Capture

Data needs to be extracted periodically from the source system(s) and transformed to the data warehouse. This process is commonly referred to as refreshing the data warehouse. The most efficient refresh method is to extract and transform only the data that has changed since the last extraction. Change Data Capture identifies and processes only the data that has changed in each of the tables in a database and makes the changed data available to the Data Warehouse. BIA Reporting has been designed with the intention that the refresh will take place on a nightly basis. However, the implementation is flexible and it is possible to run the refresh at a different frequency.

BIA Reporting 'Change Data Capture' techniques include using a control table which stores a last written date for each table that is being populated. When an ETL runs, the last written field for that table is also updated. The next time the

ETL runs, it first reads from this control table and then extracts the data that has been updated since the previous ETL run.

It is important to note that for change data capture to work in BIA Reporting all the last written fields must be populated in the source tables that the reporting solution extract data from.

There are three control tables provided with BIA Reporting. Each of the control tables contain a list of all the tables that are populated in that database:

- **Staging ETL Control table** : This table is created in the Staging database and is used to extract data from tables in the source database to tables in the Staging database using the last written field to extract only the data that has changed since the last extraction. This table includes a truncate flag which, when set to 'Y', will truncate the destination table in the Staging database before running the ETL. When set to 'N' the table will not be truncated before the ETL runs. The default is 'Y' as there is no need to build up a history of changes in the Staging database. 'N' is used for a reference table, CODETABLEITEM, as some ETL's use this table to retrieve descriptions of codes.
- **CDW ETL Control table** : This table is in the CDW and is used to extract data from tables in the Staging database, using the last written field to extract only the data that has changed since the last extraction.
- **Datamart ETL Control table** : This table is in the Datamart and is used to extract data from tables in the CDW, using the last written field to extract only the data that has changed since the last extraction.

As already stated, a row in the ETL Control table is updated before and after every ETL run for the table which is being updated. This works by the ETL's calling a pre-mapping transformation to read the previous last written date and setting the extract time. After the ETL has run, a post-mapping transformation is called which updates the last written date to the current date. This functionality is not supported by the ETL tool. These transformations are custom BIA Reporting transformations written in java called from the ETL tool.

After the ETL Control table has been initially populated with data (see next section) the last written date is reset to a start date to ensure that the ETL's extract all data updated after this date. The developer can manually set the last written date or use the `resetetl.XXX` build command (where x is staging, central, or datamarts) which resets the last written date for all tables in that database to the 1st of January, 1934.

Sequence of running ETLs

The BIA Reporting databases can be populated by executing the ETLs. The databases must be populated in the following order:

- **Staging database**
- **Central database (CDW)**
- **Datamarts**

Within each database the ETLs may need to be run in a particular order. The sequence of running ETLs is described below. The reasons why some ETLs need to be run in a database before others are:

- **Control tables:** the ETL Control tables in the databases must be populated before the other ETLs are run. This is because each ETL identifies and processes

only the data that has been added or updated for each of the tables in a database. The ETL identifies this data through the last written date in the ETL Control tables.

- **Dependencies in the CDW:** foreign key dependencies exist on some of the tables in the CDW as this database is normalized. This means that some ETLs cannot be run until other ETLs are complete e.g DW_CASESTATUSHISTORY_ETL cannot be run until the DW_CASE_ETL has completed successfully as the prior ETL needs to retrieve the DWCASEID from the latter.
- **Datamart Dimensions and Facts:** dependencies in the datamarts mean that fact tables cannot be populated until all the dimensions that the fact table reference are populated.

The build script should be used to run ETLs in Oracle or DB2. The run commands allows a user to run control and operational ETLs in the Staging, CDW, or Datamarts. Section **Configuring the BIAApplication.properties** file details the build run targets, the sequencing of the ETLs in the Staging, CDW, and Datamarts and how to amend any run batch files.

Performance Tuning and Optimization

There are many ways to enhance the performance in the reporting environment and they are dependent on the database and environment that the reporting solution runs in. The following list some methods which can be used to optimize BIA Reporting:

- **Database Indexing** : Each of the database vendors include different indexing algorithms that can be used to increase the speed with which queries are serviced. BIA Reporting provides indexing but the optimal indexing strategy is dependent on the environment.
- **Database Partitioning** : Partitioning involves the task of breaking a single database table into sections which are stored in multiple files. The strategy used may increase performance.
- **Materialized Views** : A materialized view is a database object that contains the results of a query. As discussed in the section ' *Running Reports* ' materialized views can increase the performance when running reports.

Creating custom Java transformations

The Reporting module supports the creation of custom transformations to support or augment ETL processes. Transformations can be created by writing Java classes that are deployed into the database and accessed through the database engine as an SQL function or as an SQL procedure. Each transformation is deployed into a database schema, for example, staging, warehouse, or the data mart schema. The transformation can read, via JDBC, tables that are in the same schema as the transformation; write data back into tables in the same schema or return a result back up to the caller, typically an ETL process. SQL functions typically return values, SQL procedures typically write data back into Curam BI tables.

The following steps show how to create a new java transformation.

1. Within Eclipse, go to the project CuramBITransforms you can view and change your java code .
2. Create your Java code in the custom directory, for example custom/source. Using the standard build commands to build and package your classes in to a JAR file, for example build jar. Refresh the project CuramBITransforms in Eclipse.

3. From the file `ddl\oracle\staging\transforms.sql` add your SQL function or procedure by adding the SQL definition.
4. From the directory `Reporting\components\core`, build the database, for example if you created a transformation to be deployed with the staging schema, rebuilding the staging database creates the transformation as an SQL function or procedure.

If you have a requirement to create a new normalized table to store, for example `CreoleCaseDetermination` data, the following process describes the recommend process, see the `developerWork` article https://www.ibm.com/developerworks/community/blogs/5e15a5a7-d4d6-4880-bd9c-e6819061a832/entry/accessing_cer_determination_results_for_business_intelligence2?lang=en. Once you have a new normalized table you can develop new ETL processes that extract data from the new normalized table to the staging schema.

Modifying the class path for the project `CuramBITransforms` is a non-compliant action making future upgrades more difficult. In the case where you are considering new Java code within the Eclipse project `CuramBITransforms` that requires you to modify the projects class path. It is recommended that you create a new Eclipse project. The new project can use Jars from `CuramBITransforms` as well as a new library.

Summary

This chapter details the setup and running of BIA Reporting. The reader should have acquired a clear understanding of the design choices and the flexibility of the reporting framework provided. This chapter included:

- details on setting up and running the BIA Reporting solution.
- a section on the build script and how it is used in BIA Reporting.
- the change data capture method used in BIA Reporting.
- a section on the sequencing needed when running ETL's.
- a section on further possible optimization techniques.

Installation and Configuration

Overview

This chapter outlines the installation and configuration of the BIA Reporting Module and associated third party products.

An assumption is made that the developer knows how to use the setup tools for their database platform.

Ant setup

Apache Ant is a Java-based tool with XML-based configuration files. BIA Reporting uses this open-source software to deploy the metadata into the databases and ETL tools from files on disk.

To install Ant choose a directory and copy the distribution file there. This directory will be known as `ANT_HOME`. The Apache Ant distribution files can be found on the core installation disks or downloaded from ant.apache.org. Version 1.8.2 should be installed.

Once Ant has been installed some modifications need to be made:

- set the System Environment variable **ANT_HOME** to the location you have just unzipped Ant to.
- add the entry %ANT_HOME%\bin; to the start of your **PATH** environment variable.
- create an environment variable **ANT_OPTS** and set this variable to "-Xmx512m" (do not include the quotes).
- Ensure the following are included in the components\BIBuildTools\lib-ext directory
 - ant.jar
 - ant-contrib-0.6.jar
- To confirm that ANT was installed correctly, open a command prompt and type 'ant -version' and hit return. If ANT was installed correctly then the ANT version will be displayed

The Ant-Contrib project is a collection of user supplied tasks (like and <if> task) and a development playground for experimental tasks like a compilation task for different compilers. The Apache Ant distribution files can be found on the core installation disks or downloaded from ant-contrib.sourceforge.net. Ant-contrib-0.6 is the version to be installed.

Java Platform, Standard Edition installation

Java Platform, Standard Edition is a java-based, runtime platform that is used during the running of the reporting solution. It should have been automatically installed and set up as part of the database installation for Oracle or DB2, However a different version may be required in order to run the reporting solution.

The BIA Reporting Build Environment requires a Java JDK version that is compatible with the RDBMS vendor software (each version RDBMS has a JDK embedded in the installed footprint).

Download and install a Java JDK compatible with the RDBMS. Then, set the JAVA_HOME system environment variable to the Java JDK location. See the IBM® Cúram Social Program Management Version 6 Supported Prerequisites document for the required Java Platform, Standard Edition version.

Other Environment Variables

This Section outlines the other Environment Variables that need to be set.

Please note that you will only be able to set the Environment Variables *AFTER* you have installed the Reporting components in the following sections, as you will not know the directory paths beforehand.

Common

This Section outlines the other System Environment Variables that need to be set for both Oracle and DB2:

- *ANT_HOME*, *ANT_OPTS*, *PATH* - see the section *Ant setup*
- *JAVA_HOME* - see Section *Java Platform, Standard Edition installation*
- *REPORTING_DIR* - this points to the Reporting directory that is installed in Section ' *Install the BIA Reporting Module* for example, C:\Reporting
- *PATH* - the setting of this variable depends on the type of installation:

- If you have DB2 on the machine, then add %JAVA_HOME%\bin to the beginning of the Variable Value
- If you have Oracle on the machine, then add %JAVA_HOME%\bin to the beginning of the Variable Value
- If the reporting build commands are running on a machine that only has OWB installed that is, no Oracle instance, then ensure that %ORACLE_HOME%\bin is added to the path, but ensure the %JAVA_HOME%\bin is still at the beginning of the Variable Value.

DB2

This Section outlines the other System Environment Variables that need to be set for DB2:

- **DB2DIR** - This is set to the DB2 directory, e.g. C:\Program Files\IBM\SQLLIB
- **JAVA_HOME_RDBMS** - This is set to DB2 Java folder, e.g. **DB2DIR** \java\jdk

(deprecated) Oracle

This Section outlines the other System Environment Variables that need to be set for Oracle:

- **ORACLE_HOME** - this is set to the Oracle directory, e.g. C:\oracle\product\11.2.0\db_1. Note: If the reporting build commands are running on a machine that only has OWB installed i.e. no Oracle instance then set the ORACLE_HOME to the parent directory of OWB_HOME. This is to enable the build scripts to find the loadjava.bat and javac.exe files. (ORACLE_HOME dependencies:loadjava.bat & jdk).
- **OWB_HOME** - this is set to the OWB directory, e.g. ORACLE_HOME\owb. Ensure that when this environment variable is set that the path %OWB_HOME%\bin\win32 is correct. (OWB_HOME dependencies: OMBPLUS, loadjava.bat & jdk).
- **JAVA_HOME_RDBMS** - This is set to the Oracle Java folder,e.g. %ORACLE_HOME%\jdk

Install the BIA Reporting Module

The BIA Reporting Module can be installed by either installing the full development environment or simply copying the reporting directory in its entirety from a BIA development installation

After installing the BIA Reporting Module the BIA Reporting directories and files should be located at "..\Reporting".

Setup BIApplication.Properties and BIBootstrap.properties files

The BIApplication.properties and BIBootstrap.properties files need to be configured with the database connection details.

The passwords for Oracle, DB2, WLS and WAS need to be encrypted in the BIBootstrap.properties as follows:

- Open a command prompt from Reporting\components
- run the following 'init.bat'
- Navigate up one directory to the Reporting directory
- run the following 'bootstrap.bat'
- Navigate to Reporting\components

- Run the following 'build encrypt.password -Dpassword=<p>' where <p> is the assigned password to be encrypted
- Enter the full encrypted password returned, for example: qqnscP4c4+s== as the password in BIBootstrap.properties

Refer to sections **Configuring the BIBootstrap.properties file** and **Configuring the BIApplication.properties file & Security** for samples and further information on the BIBootstrap.properties and BIApplication.properties files

Executing BIRT Reports against Demo-Data

Demo data has been created for most of the Facts and Dimensions in the Datamart. This means that it can be used to test if the BIRT Reports are correctly displaying data without having to set up and load the Data Warehouse first.

You need to build the datamart demo data schema, populate it with our demo data, and then point the Reports at it before they can be executed.

Installation of a vendor database, i.e. DB2 or Oracle, is required before starting below steps.(Refer Sec - 6.8 for DB2 and Sec - 6.9 for Oracle Installations.)

The following steps describe how to setup our data-mart and demo-data.

1. Ensure the Demo-Data schema has been created on your database
2. Ensure the Demo-Data schema name matches the name specified in the BIBootstrap.properties file. Also ensure the connection details are specified correctly in the file.
3. Ensure the environment.demodata.disabled property is set to false in the BIApplication.properties file.
4. Run the command init at..\Reporting\components in dos prompt (Refer to Build Script Commands section).
5. Run the command build database.datamarts.demodata at..\Reporting\components in dos prompt(Refer to Build Script Commands section).

The following steps describe how to execute a report:

1. To execute BIRT reports against demo data you need to configure your data source to point to the datamart demo data schema. See the BIRT Developer Guide for the steps for creating a data source.

DB2 Environment

This section describes the steps required to install and configure the IBM Cúram Business Intelligence and Analytics (BIA) Data Warehouse solution for DB2.

An assumption is made that the developer is familiar with the tools for the DB2 database platform.

Install DB2 Database and IBM DB2 Warehouse

IBM DB2 Database needs to be installed. Please ensure that the install user has administration rights to install on the machine.

IBM DB2 Warehouse Client will need to be installed to develop ETLs.

It is also recommend to install IBM Data Studio, to make it easier to interact with the database objects and data.

If required, each of the above components can be installed on different machines.

Deploying DB2 Warehouse

- Deploy DB2 Warehouse projects to a Warehouse server (WebSphere/InfoSphere server).
- Having all dependencies in one warehouse application reduces runtime complexity as it avoids dependencies between warehouse applications on the server. Otherwise, you must manage dependencies between control flows in multiple deployed warehouse applications on the server.
- Create a single warehouse application (zip file) from the core directory.
- Use the build command `build.infosphere.merge` to merge all components into the Core component. Components are defined by the component order environment variables/properties.
- Create your control flows and create a warehouse application that contains all the ETL artifacts, when all ETL metadata is present in the core directory.
- Clear down any copied files by using the command `infosphere.clean`, if necessary.

DB2 BIApplication.properties, BIBootstrap.properties and Environment Variable Setup

Once the DB2 database and DB2 Warehouse have been successfully installed the below setup steps must be completed.

The `BIApplication.properties` and `BIBootstrap.properties` files need to be created in `..\Reporting\project\properties`. These files will contain the database connection details and other variables.

`BIApplication.properties.sampledb2` and `BIBootstrap.properties.sampledb2` have been provided for guidance in `..\Reporting\project\properties`. These files can be copied and renamed as `BIApplication.properties` and `BIBootstrap.properties` as a start point. They must be kept in the same folder. Please refer to sections [Configuring the BIBootstrap.properties file](#) and [Configuring the BIApplication.properties file](#) for more information when setting the properties and following the below steps. Please refer to [Security section](#) for security details.

Please note that the Staging and Central Tables will be deployed on to the same Database.

Ensure the following environment variables exist and are set correctly:

- `ANT_HOME` - e.g. `C:\apache-ant-1.8.2` or `C:\ant182`
- `JAVA_HOME` - ensure this is set to the jdk of the java location. e.g. `C:\jdk1.6.0`. This MUST be java 1.6 or higher.
- `INFOSPHERE_SQLLIB_DIR` - this is set to point at the IBM SQLLIB folder, e.g. `C:\IBM\SQLLIB\`

These variables also need to be set in `..\Reporting\setEnvironmentBI.bat`:

- `INFOSPHERE_SQLLIB_DIR` - this is set to point at the IBM SQLLIB folder, e.g. `C:\IBM\SQLLIB\`
- `INFOSPHERE_WAREHOUSECLIENT_DIR` - this needs to be set to the parent of the IBM `eclipse.exe` file, e.g. `C:\PROGRA~1\IBM\DS3.1.1`

DB2 Bootstrap Your Environment

Follow these steps to test your environment setup:

1. Create a log4j.properties file in the *Reporting\project\properties* directory. please refer to Configuring the log4j.properties file for further details.
2. Start a command prompt and navigate to *..\Reporting\components*
3. Run the 'init' command
4. Run the command 'build jar'

These commands bootstrap the environment and compile the source files. If they fail verify that the contents of the BIApplication.properties file are correct and refer to the troubleshooting section.

Create DB2 target schema's

This section outlines the steps for creating the BI Data Warehouse Databases, i.e. the target Databases for Staging, Central and Datamarts.

It is important to note that the Staging and Central tables should reside in the same database. For the purpose of this document, the Staging and Central tables are stored in the Central, i.e. CuramDW, database. Failure to have the Staging and Central tables co-located in the same database will result in errors when attempting to execute the Central ETL processes.

Follow these steps to create the required Databases:

1. In order to change the language used in localized property names which is defaulted to English, please modify the Component.locale.order.installedLanguage variable in BIApplication.properties and also modify the language code in the 3 initialdata.sql files (Please refer to Globalization section for full details)
2. Update all Properties in BIBootstrap.properties, which can be found at *..\Reporting\project\properties*. Ensure the database names and user names match your project naming standards. Each name will be a database:
 - staging.db.name=CuramDW - this will contain the Staging Database Objects
 - central.db.name=CuramDW - this will contain the Central Database Objects
 - centraldm.db.name=CuramDM - this will contain the Datamart Database Objects
 - dmdemodata.db.name=CuramDMO - this will contain the Demo Data Database Objects
 - design.db.name=CuramDW - this will be used as the WAS execution database
3. All passwords are stored as encrypted strings in the BIBootstrap.properties file. Run this command to encrypt your passwords, replacing password with your own password:
 - build encrypt.password -Dpassword=password
4. Copy and paste the encrypted string that the command returns into the password properties in the BIBootstrap.properties file.
5. Start a command prompt and navigate to *..\Reporting\components*. Run the build db2.create.database command to create the BI Data Warehouse Databases. Please note that the Staging and Central Tables will be deployed on to the same CuramDW Database, and the Datamart Tables will be deployed onto a separate CuramDM Database. This command takes these parameters:
 - Ddatabase.name - This specifies the name of the Database that will be created
 - Ddb2.drive - This specifies the drive
 - Ddb2.dir - This specifies the path to the SQLLIB folder

- Ddatabase.userid - This specifies a Database Super User Name who has permission to create a Database
- Ddatabase.userpassword - This specifies the Password for the User Name, which should be encrypted

Here are sample commands to create the CuramDW and CuramDM Databases. These commands contain sample values – please set them as specified by your own project standards. If required, the commands can be used to create the Demo Data Database, and the Execution Database by changing the Database Names and Passwords:

- build db2.create.database -Ddatabase.name=CuramDW -Ddb2.drive=c:\ -Ddb2.dir=C:\IBM\SQLLIB -Ddatabase.userid=db2admin -Ddatabase.userpassword="/B22j7ZBq+gjuWdxwts++A=="
 - build db2.create.database -Ddatabase.name=CuramDM -Ddb2.drive=c:\ -Ddb2.dir=C:\IBM\SQLLIB -Ddatabase.userid=db2admin -Ddatabase.userpassword="/B22j7ZBq+gjuWdxwts++A=="
6. Run the build db2.create.role command to create the role which contains the required permissions to use the above Databases. This command takes these parameters:
- Ddatabase.name - This specifies the name of the Database that the role will be created for
 - Dcreate.db2role - This specifies that the role is to be created
 - Denvironment.db2.dba.userid - This specifies the User Name that will be connecting to the Database when running ETL's
 - Denvironment.db2.dba.password - This specifies the Password for the User Name, which should be encrypted

Please note that a Database User cannot grant a role to itself.

Here are sample commands to create the roles for the Users of the CuramDW and CuramDM Databases. These commands contain sample values – please set them as specified by your own project standards. Also, the roles that they create are only intended as a quick starting point for a Development Environment – please apply your own Database Security Policies when configuring your Databases:

- build db2.create.role -Ddatabase.name=CuramDW -Dcreate.db2role=true -Denvironment.db2.dba.userid=db2user -Denvironment.db2.dba.password="/B22j7ZBq+gjuWdxwts++A=="
- build db2.create.role -Ddatabase.name=CuramDM -Dcreate.db2role=true -Denvironment.db2.dba.userid=db2user -Denvironment.db2.dba.password="/B22j7ZBq+gjuWdxwts++A=="

Test Configuration

Follow these steps to check if the environment has been set up correctly:

1. Start a command prompt and navigate to ..\Reporting\components
2. Run the 'build configtest' command
3. Resolve any errors

This command will also generate default connection profiles that will be imported into DB2 Warehouse in a subsequent section. They will be used to connect to the Databases that have been specified in the BIBootstrap.properties file.

Create Data Warehouse Databases

Follow these steps to create the Data Warehouse Databases:

1. Start a command prompt and navigate to ..\Reporting\components

2. Run the 'build database.all' command

This command will create all of the Data Warehouse Database objects, such as Tables, Views, Functions, Procedures and Sequences in the Central and Datamart Databases.

Please note that this command executed 'build staticdata'. This copies/merges the static data files and the control files to the ..\Reporting\components\core\etl\CuramBIWarehouse\package-misc folder. It also set the Flat Files Path Variable in DB2 Warehouse to point to this folder.

Note: If the DB2 Warehouse client is not installed on the DB2 server then this static data will need to be manually copied to the DB2 server and ensure that the Static Data File Variable path points to that location. This Variable will be set in a subsequent section.

The command also executed 'build database.staging', 'build database.central', 'build database.datamarts', which create the Staging, Central, and Datamart Database objects. These commands can be run individually, when a full Data Warehouse build is not required.

Set Up and Configure BIA Project in DB2 Warehouse

Follow these steps to import the BI Project into DB2 Warehouse Design Studio:

1. Open DB2 Warehouse Design Studio
2. Set the Workspace as required
3. Select File - Import
4. Select General - Existing Projects into Workspace then select Next
5. Select Browse and navigate to ..\Reporting\components\core\etl\CuramBIWarehouse
6. Select OK and Finish to import the Data Warehouse Project into DB2 Warehouse.

Follow these steps to import the Database Connection Profiles, which were created when running 'build configtest' above:

1. In DB2 Warehouse Design Studio open the Data Source Explorer
2. Select the Import Icon -to open the Import Connection Profiles window
3. Select Browse and navigate to ..\Reporting\components\core\etl\CuramBIWarehouse\database-connections
4. Import each of the 5 Connection Profiles in turn:
 - curamdb.xml - this contains the connection details to connect to the Source Application Database
 - curamst.xml - this contains the connection details to connect to the Staging Objects on the Central Database
 - curamdw.xml - this contains the connection details to connect to the Central Database
 - curamdm.xml - this contains the connection details to connect to the Datamart Database
 - curamdmo.xml - this contains the connection details to connect to the Demo Data Database
5. After importing the Connection Profiles right click each of the Database Connections in DB2 Warehouse and select Properties to check that their properties are all correct

6. Enter the required password into each Database Connection
7. Right click on each of the Database Connections and select Connect - they should all successfully connect to each of their Databases.

The Flat Files Path Variable should have been set when running 'build configtest' above. Follow these steps to check that the Flat Files Path Variable is correctly pointing to the Static Data Files:

1. In the DB2 Warehouse Design Studio Data Project Explorer open the CuramBIWarehouse folder
2. Right click on the Variables folder and select Manage Variables
3. In the Manage Variable window, select the CuramBIWarehouse Project and click Next
4. Select the FLATFILES_PATH Variable Group
5. Select the PATH_V Variable and click Edit
6. Ensure that the path is pointing to point at this folder `..\Reporting\components\core\etl\CuramBIWarehouse\package-misc`
7. Select OK and Finish.

To change the schema name from the default value of DB2ADMIN, follow these steps to specify the schema name for staging, warehouse and datamart tables. Also, ensure that the schema name matches what is in your BIBootstrap.properties file.

Merging DB2 Warehouse Projects

If you have purchased another IBM Curam BIA application module, such as BIA for Child Welfare (CCS) or BIA for Income Support (IS), then follow these steps to merge them into the main CuramBIWarehouse Project in DB2 Warehouse Design Studio:

1. Open a command prompt and navigate to `..\Reporting\components`
2. Run 'build infosphere.merge' - this will copy the CCS and IS Project meta data into the main project
3. In DB2 Warehouse Design Studio right click on the CuramBIWarehouse Project and select Refresh

The CCS and IS Project meta data should now be visible in the CuramBIWarehouse main project in DB2 Warehouse Design Studio.

These other commands are available to aid the process of merging the other Project meta data into the main Project:

1. build infosphere.police - this highlights any file name collisions
2. build infosphere.clean - this cleans the "main" project of merged artifacts

Running DB2 Warehouse Control Flows

Follow these steps to run the DB2 Warehouse Control Flows:

1. In the DB2 Warehouse Design Studio Data Source Explorer window ensure that these 3 Database Connections are connected to their Databases:
 - curamdb
 - curamdw
 - curamdm
2. In the Data Project Explorer open the CuramBIWarehouse\Control Flows folder
3. Run the Core Control Flows in this order - they will in turn run all of the Data Flows:

- CuramStagingCoreStaticFlow.cflowxml
 - CuramStagingCoreOperationalFlow.cflowxml
 - CuramWarehouseCoreStaticFlow.cflowxml
 - CuramWarehouseCoreLookupFlow.cflowxml
 - CuramWarehouseCoreOperationalFlow.cflowxml
 - CuramDatamartCoreStaticFlow.cflowxml
 - CuramDatamartCoreLookupFlow.cflowxml
 - CuramDatamartCoreOperationalFlow.cflowxml
 - CuramDatamartCoreMonthlyAggregateFlow.cflowxml
4. If you have CCS and IS then run the Control Flows in this order:
- CuramStagingCoreStaticFlow.cflowxml
 - CuramStagingCoreOperationalFlow.cflowxml
 - CuramStagingCCSOperationalFlow.cflowxml
 - CuramStagingISOperationalFlow.cflowxml
 - CuramWarehouseCoreStaticFlow.cflowxml
 - CuramWarehouseISStaticFlow.cflowxml
 - CuramWarehouseCoreLookupFlow.cflowxml
 - CuramWarehouseCCSLookupFlow.cflowxml
 - CuramWarehouseISLookupFlow.cflowxml
 - CuramWarehouseCoreOperationalFlow.cflowxml
 - CuramWarehouseCCSOperationalFlow.cflowxml
 - CuramWarehouseISOperationalFlow.cflowxml
 - CuramDatamartCoreStaticFlow.cflowxml
 - CuramDatamartCoreLookupFlow.cflowxml
 - CuramDatamartCCSLookupFlow.cflowxml
 - CuramDatamartISLookupFlow.cflowxml
 - CuramDatamartCoreOperationalFlow.cflowxml
 - CuramDatamartCCSOperationalFlow.cflowxml
 - CuramDatamartISOperationalFlow.cflowxml
 - CuramDatamartCoreMonthlyAggregateFlow.cflowxml

The Data Warehouse Tables should now be populated with the data that was in the Source Application Database Tables.

Customizations and Upgrades

Customizations

When you develop Cúram applications, you must comply with certain guidelines to ensure that you can easily upgrade to future versions without affecting your custom functions. Please refer to the Cúram Business Intelligence Development Compliancy guide on how to achieve this.

Upgrade Process

Please refer to the Cúram Upgrade guide for details on the processes, recommended by IBM, which must be considered when upgrading an IBM Business Intelligence and Analytics installation.

Troubleshooting

This information details possible errors and then troubleshooting tips and fixes that the user may find helpful.

Build fail errors when running the build script

Problem	Solution
<p>When using the build script to create a database schema or import metadata into the ETL tool the error 'BUILD FAILED' occurs with a description.</p> <p>The build script may have failed for a number of reasons such as the applications.properties\ BIbootstrap.properties files not set up correctly or the database environment variables are not set up correctly.</p>	<p>The user needs to isolate the error and fix the problem. Check the error messages from the build script output to isolate the problem.</p> <p>Running a 'build configtest' in the command prompt from the directory that the build script resides will test to check if the reporting environment is setup correctly. The results of running this command are displayed including a message indicating if the environment has been setup correctly (BUILD SUCCESSFUL) or not (BUILD FAILED). The exact error message for failing is also displayed. Should a command not run successfully, then prior to the 'BUILD FAILED' message, an error message report would specify where the issue needs to be fixed. An example of one of these error message reports is as follows:</p> <pre>----Report Start----- you should investigate the following issues: no informational issues Error(s), you MUST fix the following issues: Error: please set JAVA_HOME= ----Report End----- In this case, the user has not set an environment variable correctly.</pre>

'build configtest' is failing

First of all, if the build configtest fails for any reason, verify that the contents of the application.properties file is correct.

build configtest can fail with these errors:

Problem	Solution
<ul style="list-style-type: none"> C:\Curam\development\Reporting\components\core>build configtest Unable to locate tools.jar. Expected to find it in C:\ProgramFiles\Java\jre1.5.0_09\lib\tools.jarBuildfile: C:\Curam\development\Reporting\components\core\build.xml BUILD FAILED C:\Curam\development\Reporting\components\core\build.xml:12: taskdef class curam.util.reporting.internal.tasks.AntReadBuildFileName cannot be found 	<ul style="list-style-type: none"> Verify that the contents of the application.properties file is correct and copy a tools.jar file into the C:\ProgramFiles\Java\jre1.5.0_09\lib location Ensure JAVA_HOME is set to the JDK of Java 1.5 or higher.
<ul style="list-style-type: none"> Info(s), you should fix the following: Info, is not a file:REPORTING_DIR=REPORTING_DIR Info, is not a file:REPORTING_ENV=REPORTING_ENV Info, is not a file:COGNOS_HOME=COGNOS_HOME Info, is not a file:ANT_HOME=ANT_HOME Info, is not a file:ORACLE_HOME=ORACLE_HOME Info, is not a file:OWB_HOME=OWB_HOME 	<ul style="list-style-type: none"> Ensure JAVA_HOME is set to the JDK of Java 1.5 or higher.
<ul style="list-style-type: none"> [echo] info:Compiling class using RDBMS compiler, using \${java.path}\javac.exe 	<ul style="list-style-type: none"> Ensure JAVA_HOME points to a JDK home and not a JRE home.
<ul style="list-style-type: none"> BUILD FAILED C:\Curam\development\Reporting\devenvironment\scripts\oraclebuild.xml:328: The following error occurred while executing this line: C:\Curam\development\Reporting\devenvironment\scripts\oraclebuild.xml:410: The directory D:\oracle\owb\bin\win32 does not exist 	<ul style="list-style-type: none"> Ensure the OWB_HOME var is set correctly, for example,\oracle\owb
<ul style="list-style-type: none"> jar: [echo] info: Compiling class using RDBMS compiler, using \${java.path}\javac.exe [javac] Compiling 87 source files to %REPORTING_DIR%\devenvironment\build\rdbms 	<ul style="list-style-type: none"> Ensure the application.properties file exists and is correctly configured. Specifically the database type property.

Errors with permissions

Problem	Solution
In the OWB Control Center an ETL fails to deploy and gives this error message: ORA-06550: line 222, column 11: PL/SQL: ORA-00942: table or view does not exist	Running this command may resolve the problem - 'build grant.all'. This command grants permissions from staging, central, and datamarts to public. It should only be used in a Development Environment.

The lastwritten field in source table is not populated

Problem	Solution
When attempting to extract data from a source table to the Staging database using an ETL no data is being extracted. The problem is that the last written field in the source database is not being populated.	All source tables being used in BIAReporting need the last written field to be populated. The developers of the source system must ensure that this is a mandatory field and always updated. The last written field in the source table needs to be populated, the date in the control table will need to be reset and the ETL run again.

Error when running 'build transform.aggcasemonth'

Problem	Solution
If the transform.aggcasemonth build task fails run the build configtest task to ensure the project properties are correct. If the build configtest fails with the below error: <ul style="list-style-type: none">Error, ant property <environment.jdbc.jar> can't find file C:\oracle10\product\10.2.0\db_1\jdbc\lib\classes12.zip	Then navigate to the application.properties file (... \Reporting\project\properties) and ensure the environment.jdbc.jar is correctly set to the Oracle home directory. (Can vary slightly from different machines) <ul style="list-style-type: none">environment.jdbc.jar=C:\oracle\product\10.2.0\db_1\jdbc\lib\classes12.zipRerun build configtest, this should now run successfully and then run build transform.aggcasemonth which should also successfully complete.

Cognos does not start

Problem	Solution
Errors thrown in Cognos Configuration preventing Cognos from starting.	Cognos requires a Sun JRE which must be 1.5. Ensure the JAVA_HOME points to the Cognos JRE or a compatible JRE.

No Data in Tables after running `owb.environment.tests.run`

Problem	Solution
<p>There is no data in the Tables after running <code>owb.environment.tests.run</code>. There may be a number of reasons for this issue, but if this error message is in the screen output then please see the below solution:</p> <ul style="list-style-type: none">• <code>run.etl.execute</code>: [exec] ERROR: [exec] ORA-01017: invalid username/password; logon denied	<p><code>build owb.environment.tests.run</code> tries to connect to the database using <code>runtime.db.username</code>.</p> <p>The variables <code>design.db.username</code> and <code>runtime.db.username</code> in <code>..\Reporting\project\properties\BIBootstrap.properties</code> need to match.</p> <p>Please ensure that they match and that they are correct.</p>

Error reimporting due to Matching Conflicts

A number of oracle files are failing to reimport due to matching conflicts when reimporting data (using the command `build owb.import.all` when reporting folder has not been deleted) into OWB.

Problem	Solution
<p>A number of oracle files are failing to reimport due to matching conflicts when reimporting data (using the command <code>build owb.import.all</code> when reporting folder has not been deleted) into OWB.</p>	<p>In order to permanently fix this bug, clients must install patch 10195667 which can be found by logging into https://support.oracle.com/CSP/ui/flash.html. The instructions in the <code>Readme.txt</code> file provided with this patch are complicated so a more clearer version is provided in the section <code>Installing Patches</code></p>

Incorrect language code set

If the workspace is created with the incorrect language code, then the language cannot be changed. Instead, the `owbsys` user must be dropped and recreated.

Problem	Solution
If the workspace is created with the incorrect language code, then the language cannot be changed. Instead, the owbsys user must be dropped and recreated.	<p>Remove OWB workspace users and owners using the Repository Assistant (follow the steps in the section called Remove OWB Workspace Owner and Users using the Repository Assistant).</p> <p>Open an SQLPLUS command prompt, i.e. open a command prompt and run "sqlplus". Login as owbsys and then run the following:</p> <ul style="list-style-type: none"> • To clean out the OWBSYS schema, run the script in the clean_owbsys.sql file located in the \$ORACLE_HOME/owb/unifiedRepos directory. This will drop the OWBSYS user and all roles associated with it. • To recreate the OWBSYS schema and seed all the required objects, run the script located in the cat_owb.sql file. This may lock the OWBSYS account, in which case it must be unlocked using either SQL developer, or by running the following in SQLPLUS: 'alter user owbsys identified by password account unlock' where password is the assigned password to OWBSYS • Reset the OWBSYS schema by running the script located in the reset_owbcc_home.sql file. This may ask for the full path to \$ORACLE_HOME to be entered.

Import process running slowly

The OWB import process has become slow over time and is taking longer than expected.

Problem	Solution
The OWB import process has become slow over time and is taking longer than expected.	<p>To optimise your OWB repository:</p> <ol style="list-style-type: none"> 1. Open the OWB Design centre 2. Execute the "Optimise Repository" option from the Tools menu <p>The following Oracle publication provides further tips describing which features OWB provides to optimize a repository automatically: https://blogs.oracle.com/warehousebuilder/entry/introducing_optimize_repository</p>

(deprecated) Error deploying the locators in Oracle because of Oracle version

When attempting to run the *owb.deploy.runtimesetup* or the *owb.deploy.all* with Oracle an error occurs saying that there is a version difference between the database and the locators in the warehouse.

Problem	Solution
When attempting to run the <i>owb.deploy.runtimesetup</i> or the <i>owb.deploy.all</i> with Oracle an error occurs saying that there is a version difference between the database and the locators in the warehouse.	The user needs to change the database version in the locators in Oracle Warehouse Builder (OWB). In the Reporting project in OWB the user needs to select the <i>database/oracle/locations</i> folder and open the properties of each location and change the version. If the version looks to be correct then change to an older version and commit and then change back and commit again.

(deprecated) Unwanted OWB Locations

Problem	Solution
The names of the location objects in OWB are related to the schema names in the application.properties file. Every client applies their own naming conventions when creating the schemas for staging, central and datamarts. It is therefore necessary to perform some clean up operations on the OWB location objects created.	<ol style="list-style-type: none"> 1. Open the OWB Design Center and log in as the Runtime Owner. 2. In the Location Explorer, expand the Databases and Oracle options. It may be necessary to delete the following locations: Any duplicate locations prefixed with "curstaging". 3. Please do not delete the following: <ul style="list-style-type: none"> • OWB locations which are prefixed with "OWB" or "REPOSITORY" • Reporting locations which are SOURCE_LOCATION, STAGING_LOCATION, CDW_LOCATION, DATAMARTS_LOCATIONS and Static Data.

Necessary Locations

The locations which should remain are:

- Locations
 - Databases
 - Oracle
 - CDW_LOCATION
 - DATAMARTS_LOCATION
 - OWB_REPOSITORY_LOCATION
 - SOURCE_LOCATION
 - STAGING_LOCATION
 - DB2

Please note that the locations will be registered correctly when the command *md.deploy.runtimesetup* is run.

(deprecated) Errors while importing the ETLs

(deprecated) Problem	(deprecated) Solution
Errors thrown when importing the ETLs from the command line.	If there are issues importing the ETLs, check the owbimport.tcl script located here (%REPORTING_DIR%\bin\etl) and verify the OMBCONNECT string on the first line is correct.
Cannot create connection to control center	Oracle>Warehouse Builder>Administration>Start Control Center
OWBSYS is locked (This relates to any user)	In SQL developer: Sys>Other Users>OWBSYS . Then right click and choose edit user. Un-check 'Account is locked. Click Apply and close.

(deprecated) Unable to log into Control Center

Problem	Solution
The following error is thrown when trying to log into the control center: RTC-5260: Failed to connect to runtime platform, please check you have provided the correct Host,	Navigate to All Programs->Warehouse Builder->Admin and click on Start control center service. Go back and log into control center again.

(deprecated) ETL cannot deploy due to incompatible database version

The following error is returned when an ETL is deployed and the oracle version number applied to a database in the control centre is incorrect. RPE-01011 Cannot deploy to target database location because it does not have a compatible version.

The oracle version number applied to each database in Oracle Warehouse Builder should match the environment.owbconfig.version number set in the BIApplication.properties file.

(deprecated)Solution

In the Oracle Warehouse Builder Control Centre:

1. Unregister the database
2. Update the version number
3. Re-register the database

(deprecated) 'build database.all' is failing with JAR file error for DB2

Problem	Solution
<p>build database.all' is failing with this error:</p> <ul style="list-style-type: none">call sqlj.remove_jar('coreTransforms.jarInternalName') SQL20201N The install, replace or remove of of "CORETRANSFORMS.JARINTERNALNAME" failed as the jar name is invalid. SQLSTATE=46002 <p>CALL sqlj.install_jar('file:C:\Curam\development\Reporting\components\core\jar\coreTransforms.jar', 'coreTransforms.jarInternalName',0) SQL1646N A routine failed because the fenced user ID cannot access required files in the sqllib directory or other instance or database directories.</p>	<p>The DB2 server process 'Log on' must be set, via the control panel, to log on as the db2admin user account and not the 'Local system account'. Restart the service once this change has been made.</p>

Configuring the BIBootstrap.properties file.

Before using the build script it is important to set up the connection information for the databases and ETL tools. This is set up in the **BIBootstrap.properties** file which needs to be created in the *Reporting\project\properties* folder. When running the build script the connection information is obtained from this properties file.

A sample file, called *BIBootstrap.properties.samplexxx*, has been provided for guidance. It can be found in the *..\Reporting\project\properties* folder. This file can be copied and renamed as *BIBootstrap.properties* as a start point. It must be kept in the same folder.

The following lists the main components in the properties file:

- **Database Type** - Set this for Oracle (db.type=ORA) or DB2 (db.type=DB2)
- **DB2 Source Type** - Only set for DB2. The type must be set to UDB if using DB2 Universal Database (db2.source.type=UDB)
- **Connection information for the target databases** (Source, Staging, Central, Datamarts). The server, database name, username, SID, port number (if applicable) and password must be set. We will use the target database (staging, central, or datamarts) set up in the previous section as the username. Note: For DB2 Central and Staging should be set up on the same database.
- **Connection information for the Demo Data database** (curamdm demo). The server, database name, SID, username, port number (if applicable) and password must be set. The Demo Data schema is set up by running the appropriate Build Environment command.
- **Design time repository connection information.** The server, port (if applicable), database name, username and password and service name (if applicable) should be set. For Oracle we will use the design time repository user previously set up as the username. For DB2 we will use the control database automatically set up as part of the install as the username.
- **Run time repository connection information (Oracle Only).** The server, port (if applicable), database name, username and password and service name (if

applicable) should be set. For Oracle we will use the runtime repository owner previously set up as the username. For DB2 we will use the control database automatically set up as part of the install as the username.

- **Password - passwords for Oracle, DB2, WLS and WAS need to be encrypted in the BIBootstrap.properties as follows:**
 - Open a command prompt from Reporting\components
 - Run **build encrypt.password -Dpassword=<p>** where <p> is the assigned password to be encrypted
 - Enter the full encrypted password returned, for example: qqnscP4c4+s== as the password in BIBootstrap.properties

Sample BIBootstrap property file for DB2

Note: These database usernames are samples. We highly recommend that you use you standardize the names to your own conventionThe passwords below have been encrypted. Please refer to section 'Setup BIApplication.Properties and BIBootstrap.properties files' for further information

```
# DB2 connections properties for the Application Database
curamsource.db.server=kingston
curamsource.db.port=50000
curamsource.db.name=database
curamsource.db.username=db2admin
curamsource.db.password=qqnscP4c4+s==
# DB2 connections properties for the staging area
staging.db.server=kingston
staging.db.port=50000
staging.db.name=curamdw
staging.db.username=db2admin
staging.db.password=qqnscP4c4+s==
# DB2 connections properties for the central area
central.db.server=kingston
central.db.port=50000
central.db.name=curamdw
central.db.username=db2admin
central.db.password=qqnscP4c4+s==
# DB2 connections properties for the data mart
centraldm.db.server=kingston
centraldm.db.port=50000
centraldm.db.name=datamart
centraldm.db.username=db2admin
centraldm.db.password=qqnscP4c4+s==
# DB2 connection properties for the Control Database
design.db.server=kingston
design.db.port=50000
design.db.name=DWCTRLDB
design.db.servicename=
design.db.username=db2admin
design.db.password=qqnscP4c4+s==
# DB2 connections properties for the demo data data mart
# Allow demo data to be loaded in isolation to real data
dmdemodata.db.server=kingston
dmdemodata.db.port=50000
dmdemodata.db.name=demodata
dmdemodata.db.username=db2admin
dmdemodata.db.password=qqnscP4c4+s==
```

Sample BIBootstrap property file for Oracle

#Note: These database usernames are samples. We highly recommend that you use you standardize the The passwords below have been encrypted. Please refer to section 'Setup BIApplication.Properties a
Oracle connections properties for the Application Database
curamsource.db.port=1521

```

curamsource.db.name=orcl
curamsource.db.username=curam
curamsource.db.password=qqnscP4c4+s==
curamsource.db.server=kingston
# Oracle connections properties for the staging area
staging.db.port=1521
staging.db.name=orcl
staging.db.username=CuramST
staging.db.password=qqnscP4c4+s==
staging.db.server=localhost
# Oracle connections properties for the central area
central.db.port=1521
central.db.name=orcl
central.db.username=CuramDW
central.db.password=qqnscP4c4+s==
central.db.server=localhost
# Oracle connections properties for the datamart
centraldm.db.port=1521
centraldm.db.name=orcl
centraldm.db.username=CuramDM
centraldm.db.SID=orcl
centraldm.db.password=qqnscP4c4+s==
centraldm.db.server=localhost
# Oracle connections properties for the demo data schema
dmdemodata.db.port=1521
dmdemodata.db.name=orcl
dmdemodata.db.username=curamdmo
dmdemodata.db.SID=orcl
dmdemodata.db.password=qqnscP4c4+s==
dmdemodata.db.server=localhost
# Oracle connection properties for the oracle design time repository
The variables design.db.username and runtime.db.username need to be the same name. Please ensure that
design.db.server=localhost
design.db.port=1521
design.db.name=orcl
design.db.servicename=ORCL
design.db.username=curambi
design.db.password=qqnscP4c4+s==
design.db.workspacename=curambi
# Oracle connection properties for the oracle runtime time repository
runtime.db.server=localhost
runtime.db.port=1521
runtime.db.name=orcl
runtime.db.servicename=ORCL
runtime.db.username=curambi
runtime.db.password=qqnscP4c4+s==

```

Sample BIApplication property file for Oracle

```

bi.bootstrap.id=local development #1
component.order.warninglevel=error
component.order=core,childservices
environment.iexplorer.url=file:///C:/Program Files/Internet Explorer/iexplore.exe
environment.variables=REPORTING_DIR,REPORTING_ENV,COGNOS_HOME,ANT_HOME,
DB2DIR
environment.resetetl.date=01/01/1934:00:00:00environment.resetetl.dateformat=dd/mm/yyyy:hh:mm:ssenvironment
environment.datamart.aggmonth.end=30/06/2009
environment.datamart.aggmonth.dateformat=dd/mm/yyyy
environment.demodata.dateformat=dd/mm/yyyyenvironment.jdbc.jars=F:\\app\\dwtesting\\product\\11.2.0\\
environment.owbconfig.remotedatamanagerdir.failonwarnings=falseenvironment.owb.oracleinstalled=true
environment.owbconfig.version=11.2environment.owbconfig.version.sourceDB=11.2
environment.databases.curam.privileges.autogrant=false
environment.databases.bi.privileges.autogrant=trueenvironment.databases.curam.updatenulls.autorun=false

```

Configuring the BIApplication.properties file.

Before using the build script it is important to set the variables for the Build Environment and ETL tools. They are set in the **BIApplication.properties** file which needs to be created in the *Reporting\project\properties* folder.

A sample file, called *BIApplication.properties.samplexxx*, has been provided for guidance. It can be found in the *..\Reporting\project\properties* folder. This file can be copied and renamed as *BIApplication.properties* as a start point. It must be kept in the same folder.

The following lists the main components in the properties file:

- **Component Order Warning Level** - this is set to Error which means that if the Build Environment gets an Error it will stop processing the last command. If it is changed to Warning then it will carry on processing after getting an error.
- **Component Order** - If you are just using the CEF Framework then please delete *childservices* and *cgis*, leaving *core*, e.g. *component.order=core*. If you are using the Child Services Module then set *component.order=core*, *childservices*. If you are using any of the other application modules *CGIS* then please refer to the Application Properties section of their Reporting Developer Guides.
- **Component.locale.order.installedLanguage** - This is the language code for choosing the correct localizations and translations, see Globalisation section
- **Environment IExplorer URL** - This is the path to Microsoft Internet Explorer.
- **Environment Variables** - This lists the Environment Variable that are used by the Build Environment.
- **Environment ResetETL Date** - This is used by the Build Environment Targets *resetetl.staging*, *resetetl.central* and *resetetl.datamarts*. It resets all of the Last ETL Dates in each schema Control Table. It should only be used in a Development Environment and not in Production.
- **Environment ResetETL Dateformat** - this specifies the date format that *environment.resetetl.date* expects
- **Aggmonth** - these 3 properties are used by the Build Environment Target *transform.aggmonth*, they specify the Start Date, End Date and Date Format to be used to load the *DM_AGGCASEMONTH* fact
- **Environment.jdbc.drivers** - This is the Oracle driver that BIA Reporting uses, i.e. *oracle.jdbc.driver.OracleDriver*.
- **Environment.jdbc.jars** - Change the path to point to the specified JAR files, e.g. *environment.jdbc.jars=C:\\Oracle\\product\\11.2.0\\db_1\\jdbc\\lib\\ojdbc5.jar*. Please note that the double back slashes are required to fix a bug. This bug will be removed in the next release.
- **Environment.owb.oracleinstalled** - Only set this to false if there is no Oracle database on the server where OWB has been installed. Otherwise set this to true.
- **Environment.owbconfig.validate.failonwarnings** - This will check the oracle ETL's for successful validation. This will fail even if the ETL's validates with warnings.
- **Environment.owbconfig.remotedatamanagerdir** - If the Data Manager folder is on the local machine then leave it blank or it will cause an error.

If the staging, central and datamart schemas are being created on a remote server then the Data Manager folder and contents will need to be copied to this server after the build *staticdata* command has been run, refer to Section 6.8.3, Step 12. You then need to set the variable to the path of the Data Manager folder

on the remote server, using the java convention for path separators (\\),e.g. Reporting\\bin\\data_manager\\. Please ensure that the trailing \\ is added or it will cause an error.

- **Environment.owbconfig.version** - Set this to the version of Oracle you are using as the target database, e.g. 11.2, 12.1.
- **Environment.owbconfig.version.sourceDB** - Set this to the version of Oracle you are using as the source database, e.g. 11.2, 12.1.
- **Environment.owbconfig.version.scripts** - Set this as 11.2 or 12.1 which both supports 11.2 and 12.
- **Environment.owbconfig.exectemplate** - Set this to be sqlplus_exec_template.sql.
- **Environment.databases.curam.privileges.autogrant** - This grants select on all Source Tables to the Staging Schema. See Granting Database Privileges section. This should be set to false.
- **Environment.databases.bi.privileges.autogrant** - This grants select on all Staging Tables to the Central Schema. See Granting Database Privileges section. This should be set to false.
- **Environment.databases.curam.updatenulls.autorun** - This turns on/off the privilege of granting from Source Applications to Reporting Staging Schemas. See Capturing Changed Data section. Set this to be false.
- **Environment.demodata.disabled** - Set this to true to stop the Demo Data Datamart from being created. It will also disable the validating of the Demo Data Datamart during configtest. Set it to false to enable the Demo Data Datamart to be created, and to also be validated during configtest.

Build Script Commands.

The following table lists a sample of the build targets and their descriptions when using the build script. The full list of commands can also be seen from the command line by typing **buildhelp**.

Common DB2 and Oracle Commands

Build command	Description
Control ETL Processes	
all	Initializes the environment and then compiles all code and builds all the metadata for the database and ETL tool.
compile	Compiles the BIA Reporting classes.
clean	Removes datawarehouse ddl, scripts, classes and jars.
database.all	Builds the staging, central, and datamarts schema objects into the databases specified in the applications.properties file.
database.central	Builds the central data warehouse schema objects into the central database specified in the applications.properties file.
database.central.transforms	Loads central transformations into the central database.
database.datamarts	Builds the datamarts schema objects into the datamarts database specified in the applications.properties file.

Build command	Description
database.datamarts.demodata	Loads demo data into the datamart schema if present.
database.datamarts.transforms	Loads datamart transformations into the datamarts database.
database.source.updatenulls	Updates source last written column values in the Application tables if null.
database.staging	Builds the staging schema objects into the staging database specified in the applications.properties file.
database.staging.transforms	loads staging transformations into the staging database.
encrypt.password -Dpassword=<p>	Returns the encrypted version of a password <p>
export.control	Writes the contents of all control tables to a log file
jar	Packages multiple java file into reporting classes.
resetetl.central	Sets the Last ETL Date in the Control Table for the Central ETL's.
resetetl.datamarts	Sets the Last ETL Date in the Control Table for the Datamart ETL's.
resetetl.staging	Sets the Last ETL Date in the Control Table for the Staging ETL's.
staticdata	Copies/merge static data files, merge control files into..\Reporting\bin\data_manager
transform.address	A Developer can use this Build Target to help debug the Post Process in DW_ADDRESS_SP.
transform.aggday	A Developer can use this Build Target to help debug the transform in DM_AGGCASEDAY_SP.
transform.aggmonth	A Developer can use this Build Target to help debug the transform in DM_AGGCASEMONTH_SP.
transform.staticdata	A Developer can use this Build Target to help debug the -1 records that are loaded in DW_STATICDATA_SP
Operational ETL Processes	
clean	Removes datawarehouse ddl, scripts, classes and jars.
configtest	Checks that the reporting environment is set up correctly.
database.all	Builds the staging, central, and datamarts schema objects into the databases specified in the applications.properties file.
owb.import.all	Imports source,staging, central and datamart
owb.deploy.all	Deploys staging, central and datamart ETL's
run.all	Runs all the ETL's

(deprecated) Scheduling of ETL Processes

This section outlines how the execution of our reporting ETL processes can be scheduled.

A production quality scheduling and execution system should be used.

(deprecated) DB2

The ETL's can be executed in the DB2 Warehouse Design Studio and also in the Warehouse Administration Console.

The Warehouse Administration Console can be used to schedule the execution of the Control Flows.

Schedules can be created in the Warehouse Administration Console by navigating to DB2 Warehouse - SQL Warehousing - Processes- Run Processes.

(deprecated)Oracle

(deprecated)Design time platform

Process flows can be designed using Oracle Warehouse Builder. Designing process flows is described in detail in the OWB User Guide.

(deprecated)Runtime time platform

Once you create your process flows you deploy your flows to Oracle Workflow or any compliant XPDL work flow engine.

You can also execute and schedule flows using Oracle Enterprise Manager. A quote from "OWB User Guide" states:

With Warehouse Builder, you have two main options for executing process flows: you can execute them from within Warehouse Builder using the Deployment Manager as described earlier, or you can execute them from Oracle Workflow. In addition, you can use Warehouse Builder to integrate with Oracle Enterprise Manager to schedule these process flow executions. For information about Oracle Workflow, see the Oracle Workflow Guide. For information about Oracle Enterprise Manager, see the Oracle Enterprise Manager Administrator's Guide.

(deprecated)Create OWB Workspace Owner and Users using Repository Assistant

Follow these steps to create the OWB Workspace Owner and Users using the Repository Assistant:

- Ensure the Database User OWBSYS is unlocked and the Password is set using either SQL Developer or Oracle Enterprise Manager.
- Open the Warehouse Repository Assistant which can be found here: Start - All Programs - Oracle_Home - Warehouse Builder - Administration - Repository Assistant

Step 1: Enter the Host Name, Port Number and Oracle Service Name, which can all be found in your tnsnames.ora file.

Step 2: Select Manage Warehouse Builder Workspace

Step 3: Select Create a new Warehouse Builder workspace

Step 4: Select Create a workspace with a new user as workspace owner

Step 5: Enter the SYS user name and password

Step 6: Enter the Workspace Owner's User Name, e.g. ORCL, and Password. Also enter the Workspace Name, e.g. CURAMBI

Step 7: Enter the OWBSYS user name and password

Step 8: Accept the defaults or else change the Tablespaces

Step 9: Accept the default or else change the Language

Step 10: Create 5 new schemas - CuramST, CuramDW, CuramDM, CuramDMO and CuramBIROLE. These will be created as Schemas on your database so please name them as Project Standards require.

Review the Summary and click finish to build the Repository and create the Target Schemas on the Database.

- By running database.create.bischemas the CuramBIROLE is dropped, recreated and granted each of the required access rights in areas such as drop, recreate & debug. CuramST, CuramDW, CuramDM & CuramDMO are also dropped, but on recreation they are simply granted access to CuramBIROLE, giving each of them the same level of access.
- The **grant.all** command, which gets called by database.all, grants table/view permissions to the 5 schemas (see **Granting Database Privileges section**)

(deprecated)Remove OWB Workspace Owner and Users using the Repository Assistant

Before removing any owb workspace owners or users verify that these are no longer required. The OWB workspace contains the source code for your warehouse project; it is strongly recommended that a backup of the repository is taken before deletion.

Deletion of the OWB schema's through the standard Oracle tools like SQLPLUS does not achieve the same result as the following steps. When removing an OWB workspace or target schema complete the following steps.

Follow these steps to drop the **OWB Workspace Users** using the Repository Assistant:

- Open the Warehouse Repository Assistant which can be found here: Start - All Programs - Oracle_Home - Warehouse Builder - Administration - Repository Assistant

Step 1: Enter the Host Name, Port Number and Oracle Service Name, which can all be found in your tnsnames.ora file.

Step 2: Select Manage Warehouse Builder workspace users

Step 3: Enter the Workspace Owner's User Name, e.g. CuramBI and Password.

Step 4: Select the Workspace Name

Step 5: Unregister Warehouse Builder workspace users

Step 6: Select all users, e.g. CuramST, CuramDW, CuramDM, CuramDMO and CuramBIROLE

Review the Summary and click finish to unregister the Warehouse Builder workspace users.

- If required, manually drop the CuramST, CuramDW, CuramDM, CuramDMO and CuramBIROLE schemas from the database using Oracle Enterprise Manager or SQL Developer.

Follow these steps to drop the **OWB Workspace Owner** using the Repository Assistant:

- Open the Warehouse Repository Assistant which can be found here: Start - All Programs - Oracle_Home - Warehouse Builder - Administration - Repository Assistant
 - Step 1:** Enter the Host Name, Port Number and Oracle Service Name, which can all be found in your tnsnames.ora file.
 - Step 2:** Select Manage Warehouse Builder Workspace
 - Step 3:** Select Drop an existing Warehouse Builder workspace
 - Step 4:** Enter the Workspace Owner's User Name, e.g. CuramBI, and Password.
 - Step 5:** Select the Workspace Name
- Review the Summary and click finish to drop the OWB Workspace Owner.
- If required, manually drop the OWB Workspace Owner schema from the database using Oracle Enterprise Manager or SQL Developer.

Granting Database Privileges

The Reporting Build Environment provides a quick and easy way to grant the required privileges that the Data Warehouse needs. However, it should only be used in a Development Environment, when the source data is from a database that resides on a different database instance to the staging database, and not in an environment where there are stricter security priorities, like a Production Environment.

The BIApplication.properties file contains these variables which, when set to True, automatically grant the required privileges:

- **environment.databases.curam.privileges.autogrant** - grants select on all Source Source Tables to the Staging Schema
- **environment.databases.bi.privileges.autogrant** - grants select on all Staging Tables to the Central Schema, and on all Central Tables to the Datamart Schema

The **grant.all** command, which gets called by database.all, grants the permissions from tables/views in Source, Staging, Central to the Staging, Central, Datamart databases respectively:

- Source tables/views granted to Staging
- Staging tables/views granted to Central
- Central tables/views granted to Datamart

each and everyone of which need to be specified in the relevant grant file for all components:

- curam_grant
- s_grant
- dw_grant
- dm_grant

which are located in the Reporting\Components\<Specific Component>\Run\ Oracle folder for each individual component

In order for the grant.all command to execute successfully please ensure that the Source, Staging and Central users have the correct grant authority.

In a **Development Environment**, if the source data is coming from an external database different to the staging database, please ensure that:

- The environment.databases.curam.privileges.autogrant is set to false
- The environment.databases.bi.privileges.autogrant is set to true

If the source data is from the same database as that of the staging database, ensure both are set to true.

In a **Production Environment**, please ensure that these two autogrant variables are set to false, which is the default. Relevant production environment to meet personal security requirements will then need to be set up.

Capturing Changed Data

The source database is provided with default data, however from the perspective for the BIA Reporting module, some of the columns of interest are set to a null value.

BIA Reporting only extracts data from tables where a column named "LASTWRITTEN" has a non-null value. If these columns are not updated to a non-null value, then the warehouse cannot be populated with data correctly.

In order to allow the Reporting Schemas to be populated with data as quickly as possible, we provide a mechanism to update these last written columns to a non-null value. However, it should only be used in a Development Environment, and not in an environment where there are stricter security priorities, like a Production Environment.

It is possible to turn on/turn off the command to update the last written columns by setting:

- environment.databases.curam.updatenulls.autorun - This turns on/off the privilege of granting from Source Applications to Reporting Staging Schemas by setting to true/false respectively

You can update the last written columns manually by executing the target "database.source.updatenulls". Alternately, this command is also called automatically when building the database via the "database.all" command.

(deprecated)How to Add an Oracle ETL

Please follow these steps to add an ETL:

- Log into OWB Design Center
- Import any new required Tables into OWB
- Create a New Mapping under the appropriate Reporting\Databases\Oracle\Schema Name folder
- Add the Source and Target Tables
- Add the appropriate Control Table and join it to the Source Table to extract the updated records
- Add any other required Joins or Transformations
- Map the Source Columns over to the Target Columns
- Set the Load Properties on the Target Table and its Columns
- Add the Pre and Post Mapping Transformations to update the Control Table
- Validate and Deploy the ETL

- Add a record into the ETLCONTROL Table for the ETL. This can be done manually, as a once off, or by adding a new entry into the appropriate ETLCONTROL csv file and building the database
- Run the ETL to test via the OWB Control Center
- Add an entry into the appropriate run.bat file, which can be found in..\\Reporting\\components\\core\\run\\oracle. Any new ETL's should be added to a run.bat file in the custom folder, e.g...\\Reporting\\components\\core\\custom\\run\\oracle. This will ensure the new ETL gets run when the ETL's are executed through the build environment
- The ETL and updated Table metadata can be exported from OWB as .mdo files and stored on a file system

(deprecated)Known Issues

This section contains a list of all known issues and suggested workarounds:

(deprecated)Build Environment

The build command "build owb.deploy.xxx" uses the OWB TCL toolkit called OMBPLUS to deploy all ETL process in the repository. OMBPLUS does not return an error if a mapping does not deploy successfully, you must review the deployment results using the OWB Control center to verify that mappings deployed successfully.

(deprecated)Importing in OWB

Oracle Warehouse Builder ETL Loading Properties are incorrect after a meta data import. See the Oracle Note 754763 for the description of the issue and fix. Oracle recommends migrating to Oracle 11gR2 to resolve the issue. We have not seen the issue on 11gR2, but we cannot guarantee that it will not reoccur.

(deprecated)Installing Patches

(deprecated)Installing Oracle Patches

This section gives a less complicated set of steps for installing patches than those provided in the Oracle README.txt file provided with each patch. Only the installation steps are given so please refer to the README.txt file, provided with the patch for:

1. The list of files provided with the patch
2. The list of bugs fixed by the patch
3. The software required for the patch to work

(deprecated)Instructions for applying a once off patch

The instructions for applying a one off patch are as follows:

1. Once the patch is downloaded, extract the contents to a directory, for example E:\stage\12345678 where 12345678 is the no. of the patch
2. Add an environment variable OPatch with value %ORACLE_HOME%\OPatch to the system advanced properties
3. Add %ORACLE_HOME%\OPatch; to the Path variable
4. Stop the runtime service by doing the following:
 - Run a command prompt from folder %OWB_HOME%\rtp\sql
 - Login to sqlplus, i.e run "sqlplus"

- Logon as "sys as sysdba" OR "owbsys" and enter password
 - Execute "@stop_service.sql"
5. Go to Oracle>WarehouseBuilder>Administration>Stop Control Center Service, log on to the Control Center and choose Stop.
 6. Ensure all Oracle programs are closed.
 7. Run the OPatch Utility in a command prompt from the folder in which it was saved in Step 1:
 - Execute "opatch apply"

The patch will ask to rollback any other conflicting patches, when asked enter in 'y' to proceed.

If you encounter any errors - enter in 'y' to proceed and continue the installation, i.e. duplicate files exist or file cannot be found
 8. Restart the runtime service by following all the steps in step 4 except the final one which will be
 - Execute "@start_service.sql"
 9. The patch should now be fully applied. If any problems are experienced after installing the patch, it can be removed by following steps 4-8 above, only in step 7 run the following in command prompt:
 - Execute "opatch rollback -id 12345678" where 12345678 is the no. of the patch
 10. Please refer to the README.txt files for a more detailed version

Globalization

BIA Reports are defaulted in English but are also now supported in multiple languages. For languages other than English, in order to build the database in the language specific to your BIA Reporting module then set the **Component.locale.order.installedLanguage** variable in Reporting\project\properties\BIApplication.properties to the relevant language code, prior to building your database. The language codes are:

- **en** - English
- **es** - Spanish
- **pt_BR** - Portuguese (Brazil)
- **fr** - French
- **ko** - Korean
- **it** - Italian
- **zh_CN** - Chinese (PRC)
- **zh_TW** - Chinese (Taiwan)

The 3 localized property files also need to be manually updated. Located in Reporting\components\BIBuildTools\data_manager\initialdata, for each of the following sql files:

- st_initialdata.sql
- dw_initialdata.sql
- dm_initialdata.sql

change the language code (BI_PROP_VALUE) for the insert statements which contain a BIPROP_NAME = 'BI.BILOCALE'.

For example, when switching to Korean, change the code in the st_initialdata.sql file from

```
INSERT INTO ST_PROPERTIES (BIPROPERTYID, BICATEGORY,  
BIPROP_NAME,BIPROP_VALUE,BIPROP_TYPE,DEFAULTVALUE,LOCALE,LASTWRITTEN)  
VALUES (stpropertiesseq.nextval, 'CONFIG','BI.BILOCALE', 'en',  
'STRING',NULL,getDateime());
```

to

```
INSERT INTO DW_PROPERTIES (BIPROPERTYID, BICATEGORY,  
BIPROP_NAME,BIPROP_VALUE,BIPROP_TYPE,DEFAULTVALUE,LOCALE,LASTWRITTEN)  
VALUES (dwpropertiesseq.nextval, 'CONFIG','BI.BILOCALE', 'ko',  
'STRING',NULL,getDateime());
```

and make similar changes to dw_initialdata.sql and dm_initialdata.sql

(deprecated)Initial Oracle Schemas

The earlier section on 'Create OWB Repository and target schema's provides instructions on how to create the initial oracle roles and schemas with the option of renaming them.

The full list of these schemas is primarily set in the file Reporting\build\scripts\initoracleschemas.sql with recommended access granted between them and the Repositories. In order to make further changes other than the naming conventions, i.e. add extra schemas or alter the preset access granted to these schemas then we recommend the following:

- Copy the initoracleschemas.sql file to another location.
- Make the relevant changes/additions required to this new initoracleschemas.sql file
- Open a command prompt from Reporting\components
- Run the command 'database.create.bischemas' using the -D option to point to the updated copy of the initoracleschemas.sql file.
E.g. >database.create.bischemas -Dschema.createscript=[full path of updated file]\initoracleschemas.sql
to run the new script.
- Modifications to naming conventions need only be applied in the BIBootstrap.properties file

Security

BIA ETL programs require credentials to authenticate to a database and read/write data. The credentials for this connection are supplied in a configuration file (BIBootstrap.properties).

When an ETL program is ready to be used in production and/or when database builds are being prepared, the person who configures it will provide a BIBootstrap.properties file containing the batch programs production credentials. Since this is sensitive information, the production copy of this file (or the folder it is contained in) should be access-controlled to your satisfaction, e.g. so that the file can be accessed only by administrators and the batch program itself.

Recommendation:

For best practices, we advise you review your production copies of the following files, to ensure that they are access-controlled in line with your security policies:

- Reporting/project/properties/BIBootstrap.properties

You may also have automated packaging and/or deployment tasks for your application. If this automation is used to package or deploy production releases, you should consider the configuration files that support those automated processes.

BIA Reporting also has a sample create schema creation script (initoracleschemas.sql and rep_oraschemas.properties), this is only intended for use within development environments as a quick-start mechanism to getting a Reporting sandbox created and running. If these files are used for any other purposes other than within development environments then you must ensure the these files are access controlled and secure.

Please ensure any default values are replaced with secure values in line with your local security policies. We advise that you create your own secure copy of any scripts that create the BI schemas.

We also highly recommend the use of encrypted passwords in the BIBootstrap.properties file. Please refer to the password section in the Configuring the BIBootstrap.properties file section.

(deprecated)Upgrading to Oracle 11gR2

If you are moving to 11gR2, in which Oracle introduced meta-data format changes with OWB version 11g R2, all ETL processes need to be upgraded to 11g R2 format. All OOTB, OWB artifacts have been upgraded, there is no further action required for OOTB content. All custom OWB 10g ETL development artifacts needs to be upgraded to Oracle 11g R2 format. If this is not done, then OWB will upgrade the OWB content on the fly, each and every time when importing. This will result in longer build times.

How to Upgrade: Importing OWB content into OWB upgrades the ETL process, writing a new upgraded file to disk (the new file will have a postfix appended to its name). For example "test.mdo" when imported will be upgraded with a new upgraded file written to disk by OWB, the file name will be "test_11_2.mdo".

Note:

If your OWB content is subject to source control, please note that the original file is the file that is source controlled. As such you will be required to overwrite the original file with the upgraded file, ensuring that the Original file name is maintained. If you have many custom artifacts, it can be time consuming to manually rename all upgraded ETL artifacts back to their original name. 2. Please also ensure that the upgraded file is removed from the directory structure otherwise it may be imported by the build process, thus resulting in longer build times.

For points 1 & 2; there is support within the build environment to automate the copying and removing of upgraded files. Please see the build command "buildowb.upgrade.11gR2.collectfiles

(deprecated)Incremental Changed Data Extraction

Changed data is extracted during daily/nightly batch runs of the Business Intelligence ETL processes. To ensure that only changed data is extracted, ETL processes must run successfully with zero warnings/errors.

DB2 Warehouse is not configurable and enforces zero tolerance for warnings/errors with all ETL's, hence any errors/warnings will result in the ETL process being marked as failed

Oracle Warehouse Builder is configurable, allowing ETL processes to be flagged as failing. This is controlled by the 'Maximum number of errors' property value.

Default runtime behavior in Oracle Warehouse Builder:

1. When a critical SQL exception occurs within an ETL process, then the ETL is marked as failed and the changed Data Capture date is not updated.
2. When the number of errors/warnings exceeds the 'Maximum number of errors' threshold (the Oracle Warehouse Builder default is 50) within an ETL process, then the ETL will be marked as failed.
3. If the number of warnings is less than the 'Maximum number of errors' threshold, then the ETL will be marked as completed successfully but the Changed Data Capture dates will NOT be updated.

To ensure that all ETL processes are correctly flagged as either successful or failed, a zero tolerance to warnings/errors is enforced by the build process which sets the 'Maximum number of errors' property to zero for all ETL's. This is set during the import process.

Two new build commands have been introduced which are embedded in the import process but can be run individually:

1. build owb.etl.runtime.check
 - This command reports an error if the max warnings threshold is not zero. The base value can be overridden at build time using `-Denvironment.owbconfig.maxnumberoferrors=0`
2. build owb.etl.runtime.set
 - This command sets the value of the max warnings threshold to zero. The base value can be overridden at build time using `-Denvironment.owbconfig.maxnumberoferrors=0`

Configuring the log4j.properties file.

Before using the build script it is important to configure the logging properties for the Build Environment and ETL tools. These are set in the **log4j.properties** file which needs to be created in the *Reporting\project\properties* folder.

Sample contents for this file are provided below.

Sample .profile property file for UNIX

#Note: These environment variable values are samples. Please update them to your own values#

```
if [ -s "$MAIL" ]
```

```
then echo "$MAILMSG"
```

```
fi
```

#This is at Shell startup. In normal operation, the Shell checks periodically.#

```
##### WLS #####
```

```
WLS_HOME=/oracle/wls/wlserver_10.3/server
```

```
export WLS_HOME
```

```
#####
```

```
##### WAS #####
```

```
WAS_HOME=/ibm/was/AppServer
```

```
export WAS_HOME
```

```
#####
```

```
##### WAS JAVA #####
```

```
JAVA_HOME=${WAS_HOME}/java
```

```
export JAVA_HOME
```

```
JAVA_HOME_RDBMS=/oracle/database/oracle112/jdk
```

```
export JAVA_HOME_RDBMS
```

```
J2EE_JAR=${WAS_HOME}/lib/j2ee.jar
```

```
export J2EE_JAR
```

```
#####
```

```
##### ANT #####
```

```
ANT_HOME=/opt/JavaTools/apache-ant-1.8.2
```

```
export ANT_HOME
```

```
OMBPLUS=/oracle/database/oracle112/owb/bin/unix
```

```
export OMBPLUS
```

```
ANT_OPTS="-Xmx1024m -Djava.awt.headless=true"
```

```

export ANT_OPTS

#####

#####Oracle 11gR2 ####

ORACLE_HOME=/oracle/database/oracle112

export ORACLE_HOME

OWB_HOME=/oracle/database/oracle112/owb

export OWB_HOME

ORACLE_SID=CURAM112

export ORACLE_SID

LD_LIBRARY_PATH=$ORACLE_HOME/lib32

export LD_LIBRARY_PATH

LIBPATH=$ORACLE_HOME/lib:

export LIBPATH

SQLPLUS=/oracle/database/oracle112/bin

export SQLPLUS

#####

#####Reporting#####

REPORTING_DIR=/home/dwtesting/Curam/Development/Reporting

export REPORTING_DIR

REPORTING_ENV=$REPORTING_DIR/components/BIBuildTools

export REPORTING_ENV

#####

#####Components####

BI_COMPONENT_ORDER=BIBuildTools,core,childservices

export BI_COMPONENT_ORDER

BI_COMPONENT_LOCALE_ORDER=en

export BI_COMPONENT_LOCALE_ORDER

#####

```

```
#####PATH#####

PATH=/usr/sbin:/usr/bin:/usr/dt/bin:/usr/openwin/bin:/bin:/usr/ucb:/usr/
local/bin:/home/dwtesting/bin:

PATH=$PATH:$HOME/bin:${JAVA_HOME}/bin:${PATH}:${OMBPLUS}

export PATH

CLASSPATH=${CLASSPATH}:${ANT_HOME}/lib/jakarta-oro.jar:${ANT_HOME}/
lib/xalan.jar

export CLASSPATH

#####

ADBUILD_DIR=scripts

export ADBUILD_DIR

rLANG=en_US.ISO-8859-1

umask 002

CURAMSDEJ=/home/dwtesting/Curam/Development/CuramSDEJ

export CURAMSDEJ

ENV_JDBC_DRIVERS=/oracle/database/oracle112/jdbc/lib/ojdbc5.jar

export ENV_JDBC_DRIVERS
```

Sample log4j property file for DB2

```
# patterns explained at http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/PatternLayout
log4j.rootLogger=DEBUG,A1
log4j.appender.A1=org.apache.log4j.ConsoleAppender
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
# Print the date in ISO 8601 format, e.g. %d{dd MMM yyyy HH:mm:ss,SSS}
# %t Name of the thread making the log request
# %c Name of the logger associated with the log request
#
%-60c Left-justify the logger name within 60 spaces minimum
# %r Number of milliseconds elapsed since start of the application
# %p Level of the log statement
# %m
log4j.appender.A1.layout.ConversionPattern=%d{dd MMM yyyy HH:mm:ss} %-5p %m
# Print only messages of level WARN or above in the package com.foo
# Loggers may be assigned levels. The set of possible levels, that is:
#
# TRACE,
# DEBUG,
# INFO,
# WARN,
# ERROR
# FATAL
log4j.logger.curam.util.reporting=INFO
```

Notices

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

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 US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing Legal and Intellectual Property Law IBM Japan Ltd. 19-21, Nihonbashi-Hakozakicho, Chuo-ku Tokyo 103-8510, Japan

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 jurisdictions 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 provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created

programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

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

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Privacy Policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies or other similar technologies that collect each user’s name, user name, password, and/or other personally identifiable information for purposes of session management, authentication, enhanced user usability, single sign-on configuration and/or other usage tracking and/or functional purposes. These cookies or other similar technologies cannot be disabled.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at <http://www.ibm.com/privacy> and IBM’s Online Privacy Statement at <http://www.ibm.com/privacy/details> the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at <http://www.ibm.com/software/info/product-privacy>.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.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 <http://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.

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

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.

Other names may be trademarks of their respective owners. Other company, product, and service names may be trademarks or service marks of others.



Printed in USA