Cúram 8.1.3

Business Intelligence Reporting Developer Guide

Note

Before using this information and the product it supports, read the information in

Edition

This edition applies to Cúram 8.1, 8.1.1, 8.1.2, and 8.1.3.

© Merative US L.P. 2012, 2024

Merative and the Merative Logo are trademarks of Merative US L.P. in the United States and other countries.

Contents

1 Developing business intelligence reports

Condition

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

1.1 Overview

Condition

This section 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.

Cause

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

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.

1.2 Introduction to Reporting

Condition

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

Cause

What is Data Warehousing

Data warehousing is the process of collecting data, which is then organized in such a way that it can be 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 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. You then take the information from the data warehouse into Data Mart(s), which are specialized versions of the data warehouse, which is designed to suit the needs of the customer/target audience. It is then possible to view this data by using a reporting tool such as

Business Objects, Cognos BI in the form of graphs or charts and other tools. 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.

The Reporting schemas consist of: Staging, Central and Datamart schemas. Source database or the operational database from which you want to extract the data from.

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

Figure 1: Data Warehousing

An open source reporting tool that is called Business Information and Reporting Tools (BIRT) is used by the source database. BIRT is used to develop Dashboards and Reports, which display data that is stored in the Application and Data Warehouse Tables. See the *Developing Business Intelligence with BIRT* section for more information.

Figure 2: BIRT Reporting Tool

When the Staging ETL processes are run, they rely on a lastwritten timestamp column on all source tables. The lastwritten timestamp column 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. Any application must update this column whenever a row is created or modified in the source database.

Figure 3: Changed Data Capture

1.3 System Overview

DB2 Environment

Condition

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.

Cause

The installation and configuration section installs the environment that is shown here. 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 that is called, for example, SourceTest. Connections are configured between the operational database and the data warehouse databases:

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

This environment typically represents the production environment in any project in that the data warehouse is on separated machines to the production machines.

The Staging and Central tables are colocated in the same database called CuramDW. It is a requirement that the staging and central schemas are colocated in one DB2 database so that the Central ETL processes can be promoted and run 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 change and upgrade Reporting.

1.4 Runtime Architecture

Condition

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

Cause

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 a set of data models both Relational and Dimensional. These models are optimized to support reporting needs.

Architecture Overview

Runtime Architecture

Condition

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.

Cause

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.

Remedy

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

Procedure

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, for example, all new entries in a particular table, is identified it is moved to the Staging area. The Staging area is a data storage area that contains data from various data sources, and 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 to ensure that there is no 'dirty' data that might 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 that is 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 is 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 complete 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, and 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

Condition

The development architecture outlines how to extend and maintain the BIA Reporting solution.

Cause

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

Condition

The starting point is a reporting requirement. This requirement is formalized by drawing a logical user model. A user model captures the 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 can also identify the level of granularity required. The granularity is important, as it determines what and how much data is captured.

Cause

Modeling the Data Marts:

To model the Data Mart, one must logically group the user models. One related group of user models forms 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', for example, 'sales', and dimensions are a way of narrowing the view of this fact, for example, 'sales for product x in the month of June'. It is 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 that is specified in the user models. The datamarts are de-normalized and consequently querying them is 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. The CDW design 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 can 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. The Staging area 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

Condition

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 that is 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 might also need to be added to the CDW if not already there. When the data addition 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

Condition

The BIA Reporting solution provides both Java transformation code and Java Connectivity code. The Java transformation code is used when the type of transformation that is 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 that are 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 (for example, 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

Condition

The section explains each of the artifacts and the directory structure that is provided with BIA Reporting.

Cause

bin

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

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 named directory mirroring the source folder. It also contains the *data_manager.csv* static data files.

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

Figure 5: logs

Figure 4: build

1.5 The Reporting Environment Explained

Condition

The section describes the environment, setup, and execution of runtime process for BIA Reporting.

Cause

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 the section is how to use these tools (when 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 are made in the BIA Reporting solution. The section is best read along with the section on *Customizations and Upgrades*.

Initial Setup

Condition

The section briefly describes the steps that are required to install and set up the reporting solution.

Cause

An assumption is made that the developer knows how to set up the tools for the database platform they want to run BIA Reporting on.

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

ETL explained

Condition

Before explaining how the reporting solution works, it is important to define an ETL and explain its function in BIA Reporting.

Cause

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 ETLs 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 you need a separate ETL. The following example helps to explain why you need these ETLs:

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 ETLs are needed:

- An ETL that takes all the data from the source *PERSON* table and populate the Staging database *S_PERSON* table.
- An ETL that takes the required data from the staging *S_PERSON* table and populate the *DW_PERSONHISTORY* table in the CDW. Some transformations need to be performed during the population in the CDW as you 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 takes the required data from the Central *DW_PERSONHISTORY* table and populate the *DM_FACTPERSONHISTORY* fact table and some dimension tables (for example, 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.

Based on the example you can see that you need a minimum of four separate ETLs to 'push' the necessary person data from source tables to a dimensional solution that the reports can run against.

(deprecated)Metadata explained

Condition

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

Cause

The metadata provided by BIA Reporting is stored on the hard disk in the folder structure `Reporting\Components\Core\ETL`. 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 a description of how to import the metadata by 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 example, 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, for example, 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 ETLs can be imported or exported to or from the ETL tool to disk multiple developers can work on separate ETLs 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*.

Remedy

Procedure

The Build Script

Condition

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. After the metadata is loaded, the ETLs can be run from the ETL tool to populate the reporting databases with data.

The Build Script explained

Condition

The build script is used to drive the Reporting environment.

Cause

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 when it is imported. Error messages that detail the problem are displayed if the build script does not successfully validate. The build script can also be used to deploy and run ETLs to populate the tables.

It is quicker to use the build script than manually importing each of the ETLs into the warehouse tool. The reason is because the build script can import all ETLs through 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 metadata files.

When customizing ETLs in the *custom* folder the build script ensures that the customized ETLs and database schemas are being loaded and not the core metadata. Also, as the build script validates the metadata it ensures that all objects (schema and ETLs) are consistent. It achieves consistency by attempting to validate and displaying messages if errors exist. 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 occurs.

Using the Build script

Condition

Before you use the build script, it is important to set up the connection information for the databases and ETL tools.

Cause

The information is set up in the *BIBootstrap.properties* and *BIApplication.properties* files, which are in the *Reporting\project \properties* folder. Note that both of these files are set up during the installation process but they are checked before you use the build script. When you run 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, see *Configuring the BIBootstrap.properties file*. Security details for the *BIBootstrap.properties* file are provided in the *Security* section.

Now that the databases, ETL tools, and 2 property files are 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. Running build commands in the components folder builds 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 Child Services (CCS) component installed you must set the property to component.order=core,childservices.

Some examples:

To build all Cúram Platform and CCS artifacts the build commands must be run 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 build only 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**.

The developer runs the command and a list of commands is presented with and 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 you import any of the metadata it is important to check that the database, ETL tool, and 2 property files are set up correctly. Running the **build configtest** command checks that the environment is working correctly.

To build all the metadata (for example, 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 you use the build script for building database schema and importing ETL metadata. An example of component use is using the build script to import ETL metadata for the Staging database. Three options are available to the developer:

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

The developer can replace all metadata or replace isolated components. The customization process is assisted as the developer might not want to replace amended ETLs in the ETL tool or amended schema in the database.

After the build runs for any command, the results need to be checked. Messages are displayed during the build run and a final message indicates whether 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

Condition

Change Data Capture identifies and processes only the data that changed in each of the tables in a database and makes the changed data available to the Data Warehouse.

Cause

Data needs to be extracted periodically from one or more source systems 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 changed since the last extraction. BIA Reporting is designed with the intention that the refresh takes 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 is 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.

Three control tables that are provided with BIA Reporting. Each of the control tables contains 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 is changed since the last extraction. This table includes a truncate flag which, when set to 'Y', truncates the destination table in the Staging database before you run the ETL. When set to 'N' the table is 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 ETLs 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, by using the last written field to extract only the data that is 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 is 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. The ETLs work by calling a pre-mapping transformation to read the previous last written date and setting the extract time. After the ETL is run, a post-mapping transformation is called which updates the last written date to the current date. The

transformations are not supported by the ETL tool. The transformations are custom BIA Reporting transformations that are written in java called from the ETL tool.

After the ETL Control table is initially populated with data, the last written date is reset to a start date to ensure that the ETLs extract all data that is 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

Condition

The BIA Reporting databases can be populated by running the ETLs.

Cause

The databases must be populated in the following order:

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

Within each database, the ETLs might need to be run in a particular order. The sequence of running ETLs is described here. 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. Each ETL identifies and processes only the data that is 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. Therefore, some ETLs cannot be run until other ETLs are complete. For example, DW_CASESTATUSHISTORY_ETL cannot be run until the DW_CASE_ETL is 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 references are populated.

    The build script can be used to run ETLs in Oracle or DB2. The run command allows a user to run control and operational ETLs in the Staging, CDW, or Datamarts. Section *Configuring the BIApplication.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

Condition

To enhance the performance in the reporting environment depends on the database and environment that the reporting solution runs in.

Cause

These methods can be used to optimize BIA Reporting:

- **Database Indexing** : Each of the database vendors includes 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 depends on the environment.
- **Database Partitioning** : Partitioning involves the task of breaking a single database table into sections that are stored in multiple files. The strategy that is used can 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 reports are run.

Empty tab container on page

Condition

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.

Cause

Each transformation is deployed into a database schema, for example, staging, warehouse, or the data mart schema. The transformation can read, through 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.

Remedy

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

Procedure

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 into 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 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 that makes 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, create a new Eclipse project. The new project can use the Jars code from the *CuramBITransforms* project as well as a new library.

Summary

Condition

The section details the setup and running of BIA Reporting. The reader can acquire a clear understanding of the design choices and the flexibility of the reporting framework provided. The section 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 that is used in BIA Reporting.
- A section on the sequencing that is needed when running ETLs.
- A section on further possible optimization techniques.

1.6 Installation and Configuration

Condition

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

Cause

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

Ant setup

Condition

Apache Ant is a tool based on Java 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.

Cause

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.

Remedy

Procedure

To install Ant, choose a directory and copy the distribution file there. This directory is 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 can be installed.

After Ant is installed, some modifications need to be made:

1. Set the System Environment variable *ANT_HOME* to the location you just unzipped Ant to.
2. Add the entry %ANT_HOME%\bin; to the start of your *PATH* environment variable.
3. Create an environment variable *ANT_OPTS* and set this variable to `'-Xmx1400m -Dcmp.maxmemory=1400m'`.
4. Ensure that the following programs are included in the `components\BIBuildTools\lib-ext` directory:

   • ant.jar
   • ant-contrib-0.6.jar
5. To confirm that ANT was installed correctly, at a command line, type **ant -version** and press Enter. If ANT was installed correctly, then the ANT version is displayed.

Java Platform, Standard Edition installation

Condition

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

Cause

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

Download and install a Java developer kit compatible with the RDBMS. Then, set the JAVA_HOME system environment variable to the Java developer kit location. For more information, see Cúram for the required Java Platform, Standard Edition version.

Other Environment Variables

Condition

The section outlines the other Environment Variables that need to be set.

Cause

You can set the Environment Variables only *AFTER* you installed the Reporting components in the following sections, as you do not know the directory paths beforehand.

Common

Remedy

The other system environment variables that need to be set for both Oracle and DB2 are:

Procedure

1. *ANT_HOME, ANT_OPTS, PATH*. See the section *Ant setup*

    1. *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.

2. *JAVA_HOME*. See Section *Java Platform, Standard Edition installation*.
3. *REPORTING_DIR*. This environment variable points to the Reporting directory that is installed in the section *Install the BIA Reporting Module*, for example, `C:\Reporting`.
4. *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

Condition

The other system environment variables that need to be set for DB2:

- *DB2DIR*. This environment variable is set to the DB2 directory, for example, `C:\Program Files\IBM\SQLLIB`.
- *JAVA_HOME_RDBMS*. This environment variable is set to DB2 Java folder, for example, `DB2DIR \java\jdk`.

Install the BIA Reporting Module

Condition

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

Cause

After you install the BIA Reporting Module, the BIA Reporting directories and files are found at *..\Reporting.*

Setup BIApplication.Properties and BIBootstrap.properties files

Condition

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

Cause

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

Remedy

The passwords for Oracle, DB2, WLS, and WebSphere Application Server need to be encrypted in the *BIBootstrap.properties* file as follows:

Procedure

1. Go to the *..\Reporting\components* directory from a command line.
2. Run the following **init.bat**.
3. Go up one directory to the *Reporting* directory.
4. Run the following **bootstrap.bat.**.
5. Go to the *Reporting\components* directory.
6. Run the following `build encrypt.password -Dpassword=<p>` where *<p>* is the assigned password to be encrypted.
7. Enter the full encrypted password returned, for example: qqnscP4c4+s== as the password in *BIBootstrap.properties* file.

Executing BIRT Reports against Demo-Data

Condition

Demonstration data is created for most of the Facts and Dimensions in the Datamart. The demonstration data can be used to test if the BIRT Reports are correctly displaying data without having to set up and load the Data Warehouse first.

Cause

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

Remedy

Installation of a vendor database, that is, DB2 or Oracle, is required before you start.

Procedure

1. Ensure that the Demo-Data schema is created on your database.
2. Ensure that the Demo-Data schema name matches the name that is specified in the `BIBootstrap.properties` file. Also ensure that 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** from the `..\Reporting\components` directory in a DOS session. (Refer to the *Build Script Commands* section).
5. Run the command **build database.datamarts.demodata** at `..\Reporting \components` in a DOS session. (Refer to the *Build Script Commands* section).

   The following steps describe how to run a report:

   1. To run BIRT reports against demo data, you need to configure your data source to point to the datamart demonstration data schema. See the *BIRT Developer* section for the steps for creating a data source.

DB2 Environment

Condition

The section describes the steps that are required to install and configure the Cúram Business Intelligence and Analytics Data Warehouse solution for DB2.

Cause

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

Install IBM® Db2® and IBM® Db2® Warehouse

Condition

Db2® needs to be installed. Ensure that the install user has administration rights to install on the machine.

Cause

The IBM® Db2® Warehouse client needs to be installed to develop ETLs.

Also install IBM® Data Studio to make it easier to interact with the database objects and data.

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

Deploying DB2 Warehouse

Condition

Remedy

Procedure

1. Deploy DB2 Warehouse projects to a Warehouse server (WebSphere/InfoSphere server).
2. 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.
3. Create a single warehouse application (zip file) from the core directory.
4. 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.
5. 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.
6. Clear down any copied files by using the command **infosphere.clean**, if necessary.

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

Condition

After the DB2 database and DB2 Warehouse are successfully installed the setup steps must be completed.

Cause

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

Remedy

The *BIApplication.propertiessampledb2* and *BIBootstrap.propertiessampledb2* files are 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. Refer to sections *Configuring the BIBootstrap.properties file* and *Configuring the BIApplication.properties file* for more information when setting the properties and following the steps. Refer to *Security* section for security details.

Procedure

1. The Staging and Central Tables are deployed on to the same database.

    Ensure that the following environment variables exist and are set correctly:

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

2. These variables also need to be set in the *..\Reporting\setEnvironmentBI.bat* file:

    -   *INFOSPHERE_SQLLIB_DIR*, this environment variable is set to point at the IBM SQLLIB folder, for example, *C:\IBM\SQLLIB*.
    -   *INFOSPHERE_WAREEHOUSECLIENT_DIR*, this environment variable needs to be set to the parent of the IBM *eclipse.exe* file, for example, *C:\PROGRA~1\IBM\DS3.1.1*.

IBM® Db2® bootstrap your environment

Condition

Test your environment setup.

Remedy

Follow the steps to test your environment setup:

Procedure

1. Create a *log4j2.properties* file in the *Reporting\project\properties* directory. For more information, see [Configuring the log4j2.properties file](#).
2. Go to the *..\Reporting\components* directory from a command line.
3. Run the **init** command.
4. Run the command **build jar**.

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

Create DB2 target schemas

Condition

Study the steps for creating the BI Data Warehouse Databases, that is, the target databases for Staging, Central and Datamarts.

Cause

It is important to note that the Staging and Central tables can reside in the same database. For this document, the Staging and Central tables are stored in the Central, that is, CuramDW, database. Failure to have the Staging and Central tables that are colocated in the same database results in errors when you attempt to run the Central ETL processes.

Remedy

Procedure

1. To change the language that is used in localized property names, which is defaulted to English, modify the *Component.locale.order.installedLanguage* variable in the `BIApplication.properties` file and also modify the language code in the 3 `initialdata.sql` files. Refer to *Globalization* section for full details.

2. Update all Properties in the `BIBootstrap.properties` file, which can be found in the `..\Reporting\project\properties` directory. Ensure the database names and user names match your project naming standards. Each name is a database:

   - staging.db.name=CuramDW - this property contains the Staging Database Objects.
   - central.db.name=CuramDW - this property contains the Central Database Objects.
   - centraldm.db.name=CuramDM - this property contains the Datamart Database Objects.
   - dmdemodata.db.name=CuramDMO - this property contains the Demonstration Data Database Objects.
   - design.db.name=CuramDW - this property is used as the WebSphere Application Server 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. Go to the `..\Reporting\components` directory from a command line. Run the build **db2.create.database** command to create the BI Data Warehouse Databases. The Staging and Central Tables are deployed on to the same CuramDW database, and the Datamart Tables are deployed onto a separate CuramDM database. This command takes these parameters:

   - Ddatabase.name - This parameter specifies the name of the database that is created.
   - Ddb2.drive - This parameter specifies the drive.
   - Ddb2.dir - This parameter specifies the path to the `SQLLIB` folder.

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

  Here are sample commands to create the CuramDW and CuramDM databases. These commands contain sample values. 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 permissions to use the databases. This command takes these parameters:

   - Ddatabase.name - This parameter specifies the name of the Database that the role is created for.
   - Dcreate.db2role - This parameter specifies that the role is to be created.
   - Denvironment.db2.dba.userid - This parameter specifies the User Name that is connecting to the database when running ETLs.
   - Denvironment.db2.dba.password - This parameter specifies the Password for the User Name, which can be encrypted.

   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. 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. Apply your own Database Security Policies when you configure 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

Remedy

Procedure

Follow these steps to check whether the environment is set up correctly:

1. From a command line, go to the `..\Reporting\components` directory.

2. Run the **build configtest** command.

3. Resolve any errors.

   The **build configtest** command also generates default connection profiles that are imported into DB2 Warehouse in a subsequent section. The profiles are used to connect to the databases that are specified in the `BIBootstrap.properties` file.

Create Data Warehouse Databases

Condition

The **build database.all** command creates all of the Data Warehouse database objects, such as Tables, Views, Functions, Procedures, and Sequences in the Central and Datamart databases.

Cause

This command ran **build staticdata**. The command 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.

If the DB2 Warehouse client is not installed on the DB2 server, then this static data can be manually copied to the DB2 server. Ensure that the Static Data File Variable path points to that location. This variable is set in a subsequent section.

The command also ran **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.

Remedy

Procedure

Follow these steps to create the Data Warehouse databases:

1. Go to the `..\Reporting\components` directory from a command line.

2. Run the **build database.all** command.

Set Up and Configure BIA Project in DB2 Warehouse

Remedy

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

Procedure

1. Open DB2 Warehouse Design Studio.

2. Set the Workspace as required.

3. Select File - Import.

4. Select General - Existing Projects into Workspace.

5. Select Next.
6. Select Browse.
7. Go to the `..\Reporting\components\core\etl\CuramBIWarehouse` folder.
8. Select OK.
9. Select Finish to import the Data Warehouse Project into DB2 Warehouse.

Follow these steps to import the Database Connection Profiles, which were created when you run the **build configtest** command:

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.
4. Go to the `..\Reporting\components\core\etl\CuramBIWarehouse\database-connections` folder.
5. Import each of the 5 Connection Profiles in turn:
   - curamdb.xml - this profile contains the connection details to connect to the Source Application Database.
   - curamst.xml - this profile contains the connection details to connect to the Staging Objects on the Central Database.
   - curamdw.xml - this profile contains the connection details to connect to the Central Database.
   - curamdm.xml - this profile contains the connection details to connect to the Datamart Database.
   - curamdmo.xml - this profile contains the connection details to connect to the Demo Data Database.
6. 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.
7. Enter the required password into each Database Connection.
8. Right-click on each of the Database Connections and select Connect to connect successfully to each database.

The Flat Files Path Variable might be set when you run the **build configtest** command. 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.
4. Click Next.
5. Select the FLATFILES_PATH Variable Group.
6. Select the PATH_V Variable and click Edit.
7. Ensure that the path is pointing to point at this folder `..\Reporting\components\core\etl\CuramBIWarehouse\package-misc`.
8. Select OK and Finish.

To change the schema name from the default value of DB2ADMIN, follow the 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

Condition

If you purchased another Cúram Business Intelligence and Analytics application module, such as BIA for Child Welfare (CCS) or BIA for Income Support (IS).

Cause

Follow these steps to merge them into the main CuramBIWarehouse Project in DB2 Warehouse Design Studio:

1. From a command line, go to the *..\Reporting\components* directory.
2. Run **build infosphere.merge** command. The command copies the CCS and IS Project metadata into the main project.
3. In DB2 Warehouse Design Studio, right-click on the CuramBIWarehouse Project.
4. Select Refresh.

Remedy

Procedure

The CCS and IS Project metadata is now visible in the CuramBIWarehouse main project in DB2 Warehouse Design Studio.

Other commands are available to aid the process of merging the other Project metadata into the main Project:

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

Running DB2 Warehouse Control Flows

Remedy

Follow these steps to run the DB2 Warehouse Control Flows:

Procedure

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 in turn run all of the Data Flows:

- CuramStagingCoreStaticFlow.cflowxmi
- CuramStagingCoreOperationalFlow.cflowxmi
- CuramWarehouseCoreStaticFlow.cflowxmi
- CuramWarehouseCoreLookupFlow.cflowxmi
- CuramWarehouseCoreOperationalFlow.cflowxmi
- CuramDatamartCoreStaticFlow.cflowxmi
- CuramDatamartCoreLookupFlow.cflowxmi
- CuramDatamartCoreOperationalFlow.cflowxmi
- CuramDatamartCoreMonthlyAggregateFlow.cflowxmi

4. If you have CCS and IS, then run the Control Flows in this order:

- CuramStagingCoreStaticFlow.cflowxmi
- CuramStagingCoreOperationalFlow.cflowxmi
- CuramStagingCCSOperationalFlow.cflowxmi
- CuramStagingISOperationalFlow.cflowxmi
- CuramWarehouseCoreStaticFlow.cflowxmi
- CuramWarehouseISStaticFlow.cflowxmi
- CuramWarehouseCoreLookupFlow.cflowxmi
- CuramWarehouseCCSLookupFlow.cflowxmi
- CuramWarehouseISLookupFlow.cflowxmi
- CuramWarehouseCoreOperationalFlow.cflowxmi
- CuramWarehouseCCSOperationalFlow.cflowxmi
- CuramWarehouseISOperationalFlow.cflowxmi
- CuramDatamartCoreStaticFlow.cflowxmi
- CuramDatamartCoreLookupFlow.cflowxmi
- CuramDatamartCCSLookupFlow.cflowxmi
- CuramDatamartISLookupFlow.cflowxmi
- CuramDatamartCoreOperationalFlow.cflowxmi
- CuramDatamartCCSOperationalFlow.cflowxmi
- CuramDatamartISOperationalFlow.cflowxmi
- CuramDatamartCoreMonthlyAggregateFlow.cflowxmi
- The Data Warehouse Tables are now be populated with the data that was in the Source Application Database Tables.

1.7 Customizations and Upgrades

Condition

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. Refer to the *Developing Compliantly with Cúram Business Intelligence* section on how to achieve effective customization and compliancy.

Cause

Refer to the *Upgrading* section for details on the processes, recommended by Merative ™, which must be considered when you upgrade a Business Intelligence and Analytics installation.

1.8 Troubleshooting

Condition

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

Condition

The build script creates a database schema or imports metadata into the ETL tool.

Cause

The error `BUILD FAILED` can occur with a description.

The build script can fail 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.

Optional. Clearly state the steps necessary to resolve the problem. Optionally, include appropriate response role elements. Alternatively, imbed a task topic or conref that provides the steps to resolve the problem.

The user needs to isolate the error and fix the problem. Check the error messages from the build script output to isolate the problem. Running a **build configtest** command on the command line from the directory that the build script resides, tests to check whether the reporting environment is set up correctly. The results of running this command are displayed including a message that indicates whether the environment is set up correctly (`BUILD SUCCESSFUL`) or not (`BUILD FAILED`). The exact error message for failing is also displayed. If a command does not run successfully, then before 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 provided here:

```
----Report Start-------------------------
you should investigate the following issues:
no informational issues Error(s),
you MUST fix the following issues:
Error: please set JAVA_HOME=
```

In this case, the user did not set an environment variable correctly.

build configtest is failing

Condition

If the **build configtest** command fails for any reason, verify that the contents of the *application.properties* file is correct.

The **build configtest** command can fail with the following errors:

Cause

Table 1: Problems and solutions

Problem	Solution

``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``
	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 the *JAVA_HOME* variable is set to the JDK of Java 1.5 or higher.

``[echo] info:Compiling class using RDBMS compiler, using ${java.path}\javac.exe``	Ensure the *JAVA_HOME* variable points to a JDK home and not a JRE home.

``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``
	Ensure the *OWB_HOME* variable is set correctly, for example, *…\oracle\owb* .

``jar: [echo]``
 	
``info: Compiling class using RDBMS compiler, using ${java.path}\javac.exe [javac] Compiling 87 source files to %REPORTING_DIR%\devenvironment\build\rdbms``
	Ensure the *application.properties* file exists and is correctly configured, specifically the database type property.

Problem	Solution

```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```
	Ensure the *JAVA_HOME* variable is set to the JDK of Java 1.5 or higher.

Errors with Permissions

Condition

In the OWB Control Center, an ETL fails to deploy.

Cause

Required. Clearly state the symptoms of the problem.

You receive this error message:

ORA-06550: line 222, column 11: PL/SQL: ORA-00942: table or view does not exist

<tsCauses>Optional. Clearly state the causes of the problem.</tsCauses><tsEnvironment>Optional. Describe any environmental details that are not already in the title or short description.</tsEnvironment><tsDiagnose>Optional. Clearly state the steps necessary to diagnose the problem. Optionally, include appropriate response role elements. Alternatively, imbed a task topic or conref that provides the steps to diagnose the problem.<tsUserResponse>Optional. When you have particular actions that are performed by particular users, use one or more of the ts*Response elements.</tsUserResponse></tsDiagnose>

Optional. Clearly state the steps necessary to resolve the problem. Optionally, include appropriate response role elements. Alternatively, imbed a task topic or conref that provides the steps to resolve the problem

You can run the **build grant.all** command to resolve the problem.

The command grants permissions from staging, central, and datamarts to public. It can be used only in a development environment.

<tsUserResponse>Optional. When you have particular actions that are performed by particular users, use one or more of the ts*Response elements. </tsUserResponse>

The lastwritten field in source table is not populated

Condition

<shortdesc>Recommended. The shortdesc element is optional and can be either on its own or inside an <abstract> element..</shortdesc>Required. Clearly state the symptoms of the problem.

Cause

When you extract data from a source table to the Staging database through an ETL, no data is extracted.

The problem is that the last written field in the source database is not being populated.

<tsCauses>Optional. Clearly state the causes of the problem.</tsCauses><tsEnvironment>Optional. Describe any environmental details that are not already in the title or short description.</tsEnvironment><tsDiagnose>Optional. Clearly state the steps necessary to diagnose the problem. Optionally, include appropriate response role elements. Alternatively, imbed a task topic or conref that provides the steps to diagnose the problem.<tsUserResponse>Optional. When you have particular actions that are performed by particular users, use one or more of the ts*Response elements.</tsUserResponse></tsDiagnose>

All source tables that are used in BIAReporting need the last written field to be populated. The developers of the source system must ensure that the last written field 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 needs to be reset and the ETL run again.

Error when running 'build transform.aggcasemonth'

Condition

The **build transform.aggcasemonth** command does not complete successfully.

Cause

If the **build transform.aggcasemonth** build command fails, run the **build configtest** build command to ensure that the project properties are correct.

The **build configtest** build command can fail with the following error:

```
Error, ant property <environment.jdbc.jars> cannot find file C:
\oracle10\product\10.2.0\db_1\jdbc\lib\classes12.zip
```

<tsCauses>Optional. Clearly state the causes of the problem.</tsCauses><tsEnvironment>Optional. Describe any environmental details that are not already in the title or short description.</tsEnvironment><tsDiagnose>Optional. Clearly state the steps necessary to diagnose the problem. Optionally, include appropriate response role elements. Alternatively, imbed a task topic or conref that provides the steps to diagnose the problem.<tsUserResponse>Optional. When you have particular actions that are performed by particular users, use one or more of the ts*Response elements.</tsUserResponse></tsDiagnose>

Optional. Clearly state the steps necessary to resolve the problem. Optionally, include appropriate response role elements. Alternatively, imbed a task topic or conref that provides the steps to resolve the problem.

Go to the *application.properties* file, in the *...\Reporting\project\properties* directory.

Ensure that the *environment.jdbc.jars* variable is correctly set to the Oracle home directory. This setting can vary slightly on different machines.

environment.jdbc.jars=C:\\oracle\\product\\10.2.0\\db_1\\jdbc\\lib\\classes12.zip

Rerun the **build configtest** command. If successful, then run the **build transform.aggcasemonth** command, which should also successfully complete.

Cognos does not start

Condition

Cause

Errors that are produced in Cognos Configuration prevent Cognos from starting.

Cognos requires a Sun Java Runtime Environment (JRE) which must be 1.5. Ensure that the *JAVA_HOME* variable points to the Cognos Java Runtime Environment (JRE) or a compatible JRE.

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

Condition

There is no data in the Tables after you run **owb.environment.tests.run**.

Cause

A number of reasons exist for this issue. If the following error message is in the screen output, then, see *Resolving the problem*:

```
run.etl.execute:

[exec] ERROR:

[exec] ORA-01017: invalid username/password; logon denied
```

<tsCauses>Optional. Clearly state the causes of the problem.</tsCauses><tsEnvironment>Optional. Describe any environmental details that are not already in the title or short description.</tsEnvironment><tsDiagnose>Optional. Clearly state the steps necessary to diagnose the problem. Optionally, include appropriate response role elements. Alternatively, imbed a task topic or conref that provides the steps to diagnose the

problem.<tsUserResponse>Optional. When you have particular actions that are performed by particular users, use one or more of the ts*Response elements.</tsUserResponse></tsDiagnose>

Optional. Clearly state the steps necessary to resolve the problem. Optionally, include appropriate response role elements. Alternatively, imbed a task topic or conref that provides the steps to resolve the problem.

The build **owb.environment.tests.run** tries to connect to the database using **runtime.db.username**.

The variables *design.db.username* and *runtime.db.username* in `..\Reporting\project \properties\BIBootstrap.properties` need to match. Ensure that they match and that they are correct.

<tsUserResponse>Optional. When you have particular actions that are performed by particular users, use one or more of the ts*Response elements.</tsUserResponse>

Error reimporting due to Matching Conflicts

Condition

Recommended. The shortdesc element is optional and can be either on its own or inside an element..

Cause

Required. Clearly state the symptoms of the problem.

A number of Oracle files fail to reimport due to matching conflicts if you reimport data into OWB by using the command **build owb.import.all** (and if the reporting folder is not deleted).

<tsCauses>Optional. Clearly state the causes of the problem.</tsCauses><tsEnvironment>Optional. Describe any environmental details that are not already in the title or short description.</tsEnvironment><tsDiagnose>Optional. Clearly state the steps necessary to diagnose the problem. Optionally, include appropriate response role elements. Alternatively, imbed a task topic or conref that provides the steps to diagnose the problem.<tsUserResponse>Optional. When you have particular actions that are performed by particular users, use one or more of the ts*Response elements.</tsUserResponse></tsDiagnose>

To permanently fix this bug, clients must install patch 10195667, which can be found by logging in to https://support.oracle.com/CSP/ui/flash.html. See *Related Information*. The instructions in the *Readme.txt* file that is provided with this patch are complicated so a clearer version is provided in the section *Installing Patches*.

Related information
My ORACLE Support

Incorrect language code set

Condition

Recommended. The shortdesc element is optional and can be either on its own or inside an element..

Cause

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

<tsCauses>Optional. Clearly state the causes of the problem.</tsCauses><tsEnvironment>Optional. Describe any environmental details that are not already in the title or short description.</tsEnvironment><tsDiagnose>Optional. Clearly state the steps necessary to diagnose the problem. Optionally, include appropriate response role elements. Alternatively, imbed a task topic or conref that provides the steps to diagnose the problem.<tsUserResponse>Optional. When you have particular actions that are performed by particular users, use one or more of the ts*Response elements.</tsUserResponse></tsDiagnose>

Optional. Clearly state the steps necessary to resolve the problem. Optionally, include appropriate response role elements. Alternatively, imbed a task topic or conref that provides the steps to resolve the problem.

Remove OWB workspace users and owners by using the Repository Assistant (follow the steps in the section called *Remove OWB Workspace Owner and Users using the Repository Assistant*).

From an SQLPLUS command line, run the **sqlplus** command. Log in as owbsys and then run the following scripts:

- To clean out the OWBSYS schema, run the script in the `clean_owbsys.sql` file that is located in the `$ORACLE_HOME/owb/unifiedRepos` directory. This script drops the OWBSYS user and all roles that are associated with it.
- To re-create the OWBSYS schema and seed all the required objects, run the script that is located in the `cat_owb.sql` file. This script might lock the OWBSYS account, in which case it must be unlocked by using either SQL developer, or by running the following command in SQPLUS: **alter user owbsys identified by password account unlock** where password is the assigned password to OWBSYS.
- Reset the OWBSYS schema by running the script that is located in the `reset_owbcc_home.sql` file. This script might ask for the full path to `$ORACLE_HOME` to be entered.
- <tsUserResponse>Optional. When you have particular actions that are performed by particular users, use one or more of the ts*Response elements.</tsUserResponse>

Remedy

Procedure

sss

Error reimporting due to Matching Conflicts

Condition

Cause

Required. Clearly state the symptoms of the problem.

Remedy

Procedure

sss

Related information
[Introducing Optimize Repository](#)

(deprecated)Incremental Changed Data Extraction

Condition

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 ETLs, hence any errors/warnings result in the ETL process being marked as failed.

Cause

Oracle Warehouse Builder is configurable, allowing ETL processes to be flagged as failing. This configuration 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 is marked as failed.
3. If the number of warnings is less than the 'Maximum number of errors' threshold, then the ETL is marked as completed successfully but the Changed Data Capture dates is NOT updated.

Remedy

Procedure

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 ETLs. This is set during the import process.

Two new build commands are 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

1.9 Configuring the BIBootstrap.properties file.

Condition

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

Cause

A sample file, called *BIBootstrap.propertiessamplexxx*, is 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.

Remedy

Procedure

The following lists the main components in the properties file:

• **Database Type** - Set this component 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. You use the target database (staging, central, or datamarts) set-up in the previous section as the username. Note: For DB2 Central and Staging can be set up on the same database.
• **Connection information for the Demo Data database** (curamdmdemo). 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) can be set. For Oracle you use the design time repository user who is previously set up as the username. For DB2 you use the control database that is 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) can be set. For Oracle you use the runtime repository owner that is previously set up as the username. For DB2 you use the control database that is 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

Condition

```
# 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

Condition

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

Condition

```
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.datamart.aggmonth.start=30/04/2009
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\\dbhome_1\\jdbc\\lib\\ojdbc5.jar
environment.owbconfig.remotedatamanagerdir.failonwarnings=falseenvironment.owb.oracleinstalled=true
environment.owbconfig.version=11.2environment.owbconfig.version.sourceDB=11.2
environment.databases.curam.privilages.autogrant=false
environment.databases.bi.privilages.autogrant=trueenvironment.databases.curam.updatenulls.autorun=fal
```

1.10 Empty tab container on page

Condition

Before you use 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.

Cause

A sample file, called *BIApplication.propertiessamplexxx*, is 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.

Remedy

Procedure

The following lists the main components in the properties file:

- **Component Order Warning Level** - This component is set to Error, which means that if the Build Environment gets an Error it stops processing the last command. If it is changed to Warning then it carries on processing after getting an error.
- **Component Order** - If you are just using the Cúram Platform then delete childservices and cgis, leaving core, for example, 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 refer to the Application Properties section of their Reporting Developer Guides.
- **Component.locale.order.installedLanguage** - This component is the language code for choosing the correct localizations and translations, see Globalisation section.

- **Environment Variables** - This component lists the Environment Variables that are used by the Build Environment.
- **Environment ResetETL Date** - This component 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 is only used in a Development Environment and not in Production.
- **Environment ResetETL Dateformat** - This component 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 component is the Oracle driver that BIA Reporting uses, that is, *oracle.jdbc.driver.OracleDriver*.
- **Environment.jdbc.jars** Change the path to point to the specified JAR files, for example, *environment.jdbc.jars=C:\\Oracle\\product\\11.2.0\\db_1\\jdbc\\lib\\ojdbc5.jar* . Note that the double backslashes are required to fix a bug. This bug will be removed in the next release.
- **Environment.owb.oracleinstalled** - Only set this component to false if there is no Oracle database on the server where OWB is installed. Otherwise set this component to true.
- **Environment.owbconfig.validate.failonwarnings** - This component checks the oracle ETLs for successful validation. This fails even if the ETLs validates with warnings.
- **Environment.owbconfig.remotedatamanagerdir** - If the Data Manager folder is on the local machine then leave it blank or it causes an error.

  If the staging, central and datamart schemas are being created on a remote server then the Data Manager folder and contents are copied to this server after the **staticdata** build command is run,. You then need to set the variable to the path of the Data Manager folder on the remote server, by using the java convention for path separators (\\), for example, `Reporting\\bin\\data_manager\\`. Ensure that the trailing \\ is added or it causes an error.
- **Environment.owbconfig.version** - Set this component to the version of Oracle you are using as the target database, for example, 11.2, 12.1.
- **Environment.owbconfig.version.sourceDB** - Set this component to the version of Oracle you are using as the source database, for example, 11.2, 12.1.
- **Environment.owbconfig.version.scripts** - Set this component as 11.2 or 12.1 which both supports 11.2 and 12.
- **Environment.owbconfig.exectemplate** - Set this component to be sqlplus_exec_template.sql.
- **Environment.databases.curam.privilages.autogrant** - This component grants select on all Source Tables to the Staging Schema. See Granting Database Privileges section. This component is set to false.
- **Environment.databases.bi.privilages.autogrant** - This component grants select on all Staging Tables to the Central Schema. See Granting Database Privileges section. This component is set to false.
- **Environment.databases.curam.updatenulls.autorun** - This component turns on/off the privilege of granting from Source Applications to Reporting Staging Schemas. See Capturing Changed Data section. Set this component to be false.
- **Environment.demodata.disabled** - Set this component to true to stop the Demo Data Datamart from being created. It also disables 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** command run.

1.11 Build Script Commands.

Condition

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**.

Cause

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.
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.

Build command	Description
jar	Packages multiple java file into reporting classes.
resetetl.central	Sets the Last ETL Date in the Control Table for the Central ETLs.
resetetl.datamarts	Sets the Last ETL Date in the Control Table for the Datamart ETLs.
resetetl.staging	Sets the Last ETL Date in the Control Table for the Staging ETLs.
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 ETLs.
run.all	Runs all the ETLs.

1.12 Granting Database Privileges

Condition

The Reporting Build Environment provides a quick and easy way to grant the required privileges that the Data Warehouse needs. However, it can 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.

Cause

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

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

Remedy

Procedure

1. 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.

2. 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

3. For the **grant.all** command to run successfully, ensure that the Source, Staging and Central users have the correct grant authority

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

   - The *environment.databases.curam.privileges.autogrant* is set to false
   - The *environment.databases.bi.privilages.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**, ensure that these two *autogrant* variables are set to false, which is the default. Relevant production environment to meet personal security requirements then needs to be set up.

1.13 Capturing Changed Data

Condition

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.

Cause

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.

Remedy

Procedure

1. 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

2. 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.

1.14 (deprecated)Installing Patches

Condition

Cause

Remedy

Procedure

(deprecated)Installing Oracle Patches

Condition

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.

Remedy

Procedure

Only the installation steps are given so please refer to 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

Remedy

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

Procedure

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. Pleas refer to the README.txt files for a more detailed version

1.15 Globalization

Condition

BIA Reports are defaulted in English but are also now supported in multiple languages.

Cause

For languages other than English, to build the database in the language specific to your BIA Reporting module, set the *Component.locale.order.installedLanguage* variable in `Reporting\project\properties\BIApplication.properties` to the relevant language code, before you build the 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, if you switch 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,'',getDateTime());
```

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,'',getDateTime());
```

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

1.16 Security

Condition

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.*

Cause

When an ETL program is ready to be used in production or when database builds are being prepared (or both), the person who configured it provides a *BIBootstrap.properties* file containing the batch programs production credentials. As the file contains sensitive information, the production copy of this file (or the folder it is contained in) can be access-controlled to your satisfaction, for example, so that the file can be accessed only by administrators and the batch program.

For best practices, you can review your production copies of the *Reporting/project/ properties/BIBootstrap.properties* files, to ensure that they are access-controlled in line with your security policies.

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.

Remedy

Procedure

1. BIA Reporting also has a sample create schema creation script (*initoracleschemas.sql* and *rep_oraschemas.properties*). This script 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 files are access controlled and secure.

2. Ensure any default values are replaced with secure values in line with your local security policies. Create your own secure copy of any scripts that create the BI schemas.

3. Use encrypted passwords in the BIBootstrap.properties file. Refer to the password section in the Configuring the BIBootstrap.properties file section. Pub Caret -1 Pub *0000002260

1.17 Configuring the `log4j2.properties` file

Condition

Before you run the build script, configure the logging properties for the build environment and ETL tools.

The properties are set in the `log4j2.properties` file. Create the file in the `Reporting \project\properties` folder.

Sample contents for this file are as follows:

Cause

Sample 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=${WAS_HOME}/java
export JAVA_HOMEJAVA_HOME_RDBMS=/oracle/database/oracle112/jdk
export JAVA_HOME_RDBMS
J2EE_JAR=${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='-Xmx1400m -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 property file for IBM® Db2®

```
# patterns explained at https://logging.apache.org/log4j/2.x/manual/layouts.html
appenders=A1
rootLogger.level=DEBUG
rootLogger.appenderRefs=A1_reference
rootLogger.appenderRef.A1_reference.ref=ConsoleAppender
appender.A1.type = Console
appender.A1.name = ConsoleAppender
appender.A1.layout.type = 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
appender.A1.layout.pattern=%d{dd MMM yyyy HH:mm:ss} %-5p %m
# Loggers may be assigned levels. The set of possible levels, that is:
#
# TRACE,
# DEBUG,
# INFO,
# WARN,
# ERROR
#
logger.curam.util.reporting.level=INFO
```

Notices

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the Merative website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of Merative

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of Merative.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

Merative reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by Merative, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

MERATIVE MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Merative or its licensors may have patents or pending patent applications covering subject matter described in this document. The furnishing of this documentation does not grant you any license to these patents.

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

Any references in this information to non-Merative 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 Merative product and use of those websites is at your own risk.

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.

The licensed program described in this document and all licensed material available for it are provided by Merative under terms of the Merative Client Agreement.

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 Merative, 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. Merative, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. Merative shall not be liable for any damages arising out of your use of the sample programs.

Privacy policy

The Merative privacy policy is available at https://www.merative.com/privacy.

Trademarks

Merative ™ and the Merative ™ logo are trademarks of Merative US L.P. in the United States and other countries.

IBM®, the IBM® logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

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.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft™, Windows™, 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 company, product, and service names may be trademarks or service marks of others.