IBM Cúram Social Program Management

Cúram Business Intelligence Reporting Developer Guide

Version 6.0.5

IBM Cúram Social Program Management

Cúram Business Intelligence Reporting Developer Guide

Version 6.0.5

Note
Before using this information and the product it supports, read the information in "Notices" on page 77

Revised: May 2013

This edition applies to IBM Cúram Social Program Management v6.0 5 and to all subsequent releases unless otherwise indicated in new editions.

Licensed Materials - Property of IBM.

Contents

Figures

Tables

Chapter 1. Introduction

1.1 Objective

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

1.2 Document Structure

Chapter 2 & 3, *Introduction and Overview,* begins this guide with an overview of BIA Reporting. The artifacts that are provided with BIA Reporting are also explained.

Chapter 4 & 5, *Runtime Architecture and Reporting Environment* provide an overview of the architecture and the various components that make up BIA Reporting. They also outline the development process.

Chapter 6 *Installation and Configuration,* describes how to create and run BIA Reporting for both Oracle Warehouse Builder and DB2® InfoSphere Warehouse Edition. This covers installation, populating the databases, and running the reports. By following the instructions in this chapter you will have a working system installed.

Chapter 7, *Customizations and Upgrades,* covers customizations and describes how to change or extend BIA Reporting using practical examples. This chapter includes a section on how to customize your implementation in a manner that facilitates the take on of future upgrades.

Chapter 8, *Troubleshooting,* details some common pitfalls and problems that may be encountered during installation or deployment together with the solution.

1.3 Intended Audience

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

1.4 Prerequisites

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

Chapter 2. Introduction to Reporting

2.1 Objective

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

2.2 What is Data Warehousing

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

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

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

2.3 Data Warehousing

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

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

2.4 BIRT Reporting Tool

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

2.5 Changed Data Capture

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

Chapter 3. System Overview

3.1 DB2 Environment

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

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

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

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

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

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

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

3.2 Oracle Environment

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

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

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

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

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

Chapter 4. Runtime Architecture and Development Process

4.1 Introduction

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

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

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

4.2 Architecture Overview

4.2.1 Runtime Architecture

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

Data Flow

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

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

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

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

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

4.2.2 Development Process And Modelling

The development architecture outlines how to extend and maintain the BIA Reporting solution. The design process starts with a report or analysis requirement and works back creating or adding to data models, once that is done the ETL processes are designed. The ETL processes specify how the data is moved through the system from OLTP to end report.

4.2.2.1 Data Models

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

Modeling the Data Marts

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

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

Modeling the CDW

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

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

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

Modeling the Staging Database

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

4.2.2.2 ETL Process Design

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

4.2.2.3 Java Code

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

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

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

4.3 Directory Structure & Artifacts

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

4.3.1 *bin*

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

4.3.2 *build*

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

4.3.3 components/BIBuildTools

This directory contains the code for the BIA Reporting Build Environment. These include the Apache Ant files that are used by all reporting components. These build files should not be changed by the client as to allow new versions to be delivered in future releases.

4.3.4 *components/BIBuildTools/source/curam/util/reporting/internal*

The *source/curam/util/reporting/internal* directory contains Java code necessary for the internal workings of the transformations in the ETL's for both Oracle and DB2. This code includes a connectivity library.

4.3.5 *components/core*

The *components/core* directory contains artifacts for the implementation of BIA Reporting. These artifacts include the build.xml file which loads DDLs and ETL's into the required database.

4.3.6 *components/core/data_manager*

The *components/core/data_manager* folder contains data manager files to populate the static tables during the implementation of BIA Reporting. These files are populated into the database by running the required ETL which populates the static data from these files into the tables.

4.3.7 *components/core/data_manager/demodata*

The *components/core/data_manager/demodata* folder contains sample demonstration data that is used to populate the datamarts database for the purpose of demonstration of the reports to users.

4.3.8 *components/core/ddl*

The *components/core/ddl* folder contains the data definition language (DDL) scripts for the staging, central, and datamarts databases for both Oracle and DB2 used during the implementation of BIA Reporting.

4.3.9 *components/core/etl/db2*

The *components/core/etl/db2* folder contains the DB2 version of the Data Warehouse ETL's and metadata, which was created by IBM's ETL tool InfoSphere Warehouse Edition.

4.3.10 *components/core/etl/oracle*

The *components/core/etl/oracle* folder contains the extract, transform, and load (ETL) metadata for the staging, central, and datamarts databases used during the implementation of BIA Reporting. Like the DDLs the ETL metadata in the Oracle folder can only be read by Oracles ETL tool (Oracle Warehouse Builder).

4.3.11 *components/core/run*

The *source* /run directory contains batch files for Oracle which are used during the build process to run ETL's. These are intended for development and testing use only and we do not recommend that there are used in a production environment.

4.3.12 *components/core/source/curam*

The *components/core/source/curam* directory contains the Java source code for the BIA Reporting ETL Transformations.

4.3.13 *components/custom*

The components/custom folder contains a directory structure but no files. This directory is used for the customization of artifacts. The directory structure consists of core, reports, and source folders and any artifacts that need to be customized are copied to this folder. This process is described further in the *Customizations and Upgrades* chapter.

4.3.14 *logs*

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

4.3.15 *project/properties*

The *project/properties* directory contains the property files - BIBootstrap.properties and BIApplication.properties - for both the Data Warehousing databases and the operational source database. These files must be amended before implementing BIA Reporting to reflect the connection information for the Data Warehousing databases and the source database. Please refer to **Appendix A**, **Appendix B** and **Appendix N** for further details.

Chapter 5. The Reporting Environment Explained

5.1 Overview

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

5.2 Initial Setup

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

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

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

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

5.3 ETL explained

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

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

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

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

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

- An ETL that will take all the data from the source *PERSON* table and populate the Staging database *S_PERSON* table

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

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

5.4 Metadata explained

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

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

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

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

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

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

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

5.5 The Build Script

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

5.5.1 The Build Script explained

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

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

It is quicker to use the build script than manually importing each of the ETL's into the warehouse tool. This is because the build script can import all ETL's using one command. Also, because the database and warehouse tool connection information is stored in a properties file the developer does not need to repeatedly import meta data files.

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

5.5.2 Using the Build script

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

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

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

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

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

Some examples:

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

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

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

5.5.3 Running the build script

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

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

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

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

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

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

5.6 Change Data Capture

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

BIA Reporting 'Change Data Capture' techniques include using a control table which stores a last written date for each table that is being populated. When an ETL runs, the last written field for that table is also updated. The next time the ETL runs, it first reads from this control table and then extracts the data that has been updated since the previous ETL run.

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

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

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

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

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

5.7 Sequence of running ETL's

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

- **1. Staging database**
- **2. Central database (CDW)**
- **3. Datamarts**

Within each database the ETL's may need to be run in a particular order. The sequence of running ETL's is described below. The reasons why some ETL's 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 ETL's are run. This is because each ETL identifies and processes only the data that has been added or updated for each of the tables in a database. The ETL identifies this data through the last written date in the ETL Control tables.
- **Dependencies in the CDW** : foreign key dependencies exist on some of the tables in the CDW as this database is normalized. This means that some ETL's cannot be run until other ETL's are complete e.g DW_CASESTATUSHISTORY_ETL cannot be run until the DW_CASE_ETL has completed successfully as the prior ETL needs to retrieve the DWCASEID from the latter.
- **Datamart Dimensions and Facts** : dependencies in the datamarts mean that fact tables cannot be populated until all the dimensions that the fact table reference are populated.

The build script should be used to run ETL's in Oracle or DB2. The run commands allows a user to run control and operational ETL's in the Staging, CDW, or Datamarts. **Appendix B** details the build run targets, the sequencing of the ETL's in the Staging, CDW, and Datamarts and how to amend any run batch files.

5.8 Performance Tuning and Optimization

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

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

5.9 Summary

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

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

Chapter 6. Installation and Configuration

6.1 Overview

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

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

6.2 Ant setup

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

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

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

- set the System Environment variable **ANT_HOME** to the location you have just unzipped Ant to.
- add the entry %ANT_HOME%\bin; to the start of your **PATH** environment variable.
- create an environment variable **ANT_OPTS** and set this variable to "-Xmx512m" (do not include the quotes).
- Ensure the following are included in the components\BIBuildTools\lib-ext directory
  - ant.jar
  - ant-contrib-0.6.jar

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.

6.3 Java 2 Platform, Standard Edition (J2SE) installation

J2SE is a java-based, runtime platform that is used during the running of the reporting solution. J2SE should have been automatically installed and set up as part of the database installation for Oracle or DB2.

This section describes which version of Java is shipped with the supported third party tooling:

- DB2 9.1 and 9.5 both come with Java JDK 1.5
- Oracle 10.2 and Oracle 11g come with Java JDK 1.5

The BIA Reporting Build Environment requires a Java JDK 1.6 version that is compatible with the RDBMS vendor software (each version RDBMS has a JDK embedded in the installed footprint).

You will need to download and install a Java 1.6 JDK. You will then need to point the JAVA_HOME System Environment Variable at it, e.g. C:\Program Files\Java\jdk1.6.0

6.4 Other Environment Variables

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

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

6.4.1 Common

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

- **ANT_HOME, ANT_OPTS, PATH** - see Section ' *Ant setup* ' above
- **JAVA_HOME** - see Section ' *Java 2 Platform, Standard Edition (J2SE) installation* ' above
- **REPORTING_DIR** - this points to the Reporting directory that is installed in Section ' *Install the BIA Reporting Module* ' below, e.g. C:\Reporting
- **PATH** - the setting of this variable depends on the type of installation:
  - If you have DB2 on the machine then add %JAVA_HOME%\bin to the beginning of the Variable Value
  - If you have Oracle on the machine then add %JAVA_HOME%\bin to the beginning of the Variable Value
  - If the reporting build commands are running on a machine that only has OWB installed i.e. 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.

6.4.2 DB2

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

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

6.4.3 Oracle

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

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

6.5 Install the BIA Reporting Module

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

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

6.6 Setup BIApplication.Properties and BIBootstrap.properties files

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

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

- Open a command prompt from Reporting\components
- Run the following 'appbuild encrypt.password -Dpassword=<p>' where <p> is the assigned password to be encrypted
- Enter the full encrypted password returned, for example: qqnscP4c4+s== as the password in BIBootstrap.properties

Refer to **Appendix A, B & N** for samples and further information on the BIBootstrap.properties and BIApplication.properties files

6.7 Executing BIRT Reports against Demo-Data

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

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

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

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

1. Ensure the Demo-Data schema has been created on your database
2. Ensure the Demo-Data schema name matches the name specified in the BIBootstrap.properties file. Also ensure the connection details are specified correctly in the file.
3. Run the command init at..\Reporting\components in dos prompt (Refer Appendix C).
4. Run the command build database.datamarts.demodata at..\Reporting\components in dos prompt(Refer Appendix C).

The following steps describe how to execute a report:

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

6.8 DB2 Environment

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

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

6.8.1 Install DB2 Database and IBM InfoSphere Warehouse Client (IWE)

The DB2 Database should be installed first. Please ensure that the install user has administration rights to install on the machine.

IBM InfoSphere Warehouse Client should then be installed.

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

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

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

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

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

BIApplication.propertiessampledb2 and BIBootstrap.propertiessampledb2 have been provided for guidance in ..\Reporting\project\properties. These files can be copied and renamed as BIApplication.properties and BIBootstrap.properties as a start point. They must be kept in the same folder. Please refer to Appendix A and B for more information when setting the properties and following the below steps. Please refer to Appendix N for security details.

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

Ensure the following environment variables exist and are set correctly:

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

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

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

6.8.3 DB2 Bootstrap Your Environment

Follow these steps to test your environment setup:

1. Start a command prompt and navigate to ..\Reporting\components
2. Run the 'init' command
3. Run the command 'build jar'

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

6.8.4 Create DB2 target schema's

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

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

Follow these steps to create the required Databases:

1. In order to change the language used in localized property names which is defaulted to English, please modify the Component.locale.order.installedLanguage variable in BIApplication.properties and also modify the language code in the 3 initialdata.sql files (Please refer to Appendix L on Globalization for full details)

2. Update all Properties in BIBootstrap.properties, which can be found at ..\Reporting\project\ properties. Ensure the database names and user names match your project naming standards. Each name will be a database:
   - staging.db.name=CuramDW - this will contain the Staging Database Objects
   - central.db.name=CuramDW - this will contain the Central Database Objects
   - centraldm.db.name=CuramDM - this will contain the Datamart Database Objects
   - dmdemodata.db.name=CuramDMO - this will contain the Demo Data Database Objects
   - design.db.name=CuramDW - this will be used as the WAS execution database
3. All passwords are stored as encrypted strings in the BIBootstrap.properties file. Run this command to encrypt your passwords, replacing password with your own password:
   - build encrypt.password -Dpassword=password
4. Copy and paste the encrypted string that the command returns into the password properties in the BIBootstrap.properties file.
5. Start a command prompt and navigate to ..\Reporting\components. Run the build db2.create.database command to create the BI Data Warehouse Databases. Please note that the Staging and Central Tables will be deployed on to the same CuramDW Database, and the Datamart Tables will be deployed onto a separate CuramDM Database. This command takes these parameters:
   - Ddatabase.name - This specifies the name of the Database that will be created
   - Ddb2.drive - This specifies the drive
   - Ddb2.dir - This specifies the path to the SQLLIB folder
   - Ddatabase.userid - This specifies a Database Super User Name who has permission to create a Database
   - Ddatabase.userpassword - This specifies the Password for the User Name, which should be encrypted

   Here are sample commands to create the CuramDW and CuramDM Databases. These commands contain sample values – please set them as specified by your own project standards. If required, the commands can be used to create the Demo Data Database, and the Execution Database by changing the Database Names and Passwords:
   - build db2.create.database -Ddatabase.name=CuramDW -Ddb2.drive=c:\ -Ddb2.dir=C:\IBM\SQLLIB -Ddatabase.userid=db2admin -Ddatabase.userpassword="/B22j7ZBq+gjuWdxwts++A=="
   - build db2.create.database -Ddatabase.name=CuramDM -Ddb2.drive=c:\ -Ddb2.dir=C:\IBM\SQLLIB -Ddatabase.userid=db2admin -Ddatabase.userpassword="/B22j7ZBq+gjuWdxwts++A=="
6. Run the build db2.create.role command to create the role which contains the required permissions to use the above Databases. This command takes these parameters:
   - Ddatabase.name - This specifies the name of the Database that the role will be created for
   - Dcreate.db2role - This specifies that the role is to be created
   - Denvironment.db2.dba.userid - This specifies the User Name that will be connecting to the Database when running ETL's
   - Denvironment.db2.dba.password - This specifies the Password for the User Name, which should be encrypted

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

   Here are sample commands to create the roles for the Users of the CuramDW and CuramDM Databases. These commands contain sample values – please set them as specified by your own project standards. Also, the roles that they create are only intended as a quick starting point for a Development Environment – please apply your own Database Security Policies when configuring your Databases:
   - build db2.create.role -Ddatabase.name=CuramDW -Dcreate.db2role=true -Denvironment.db2.dba.userid=db2user -Denvironment.db2.dba.password="/B22j7ZBq+gjuWdxwts++A=="

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

6.8.5 Test Configuration

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

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

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

6.8.6 Create Data Warehouse Databases

Follow these steps to create the Data Warehouse Databases:

1. Start a command prompt and navigate to ..\Reporting\components
2. Run the 'build database.all' command

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

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

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

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

6.8.7 Set Up and Configure BIA Project in IWE

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

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

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

1. In IWE Design Studio open the Data Source Explorer
2. Select the Import Icon - this will open the Import Connection Profiles window
3. Select Browse and navigate to ..\Reporting\components\core\etl\CuramBIWarehouse\database-connections
4. Import each of the 5 Connection Profiles in turn:

- curamdb.xml - this contains the connection details to connect to the Source Application Database
- curamst.xml - this contains the connection details to connect to the Staging Objects on the Central Database
- curamdw.xml - this contains the connection details to connect to the Central Database
- curamdm.xml - this contains the connection details to connect to the Datamart Database
- curamdmo.xml - this contains the connection details to connect to the Demo Data Database

5. After importing the Connection Profiles right click each of the Database Connections in IWE and select Properties to check that their properties are all correct
6. Enter the required password into each Database Connection
7. Right click on each of the Database Connections and select Connect - they should all successfully connect to each of their Databases

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

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

6.8.8 Merging IWE Projects

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

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

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

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

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

6.8.9 Running IWE Control Flows

Follow these steps to run the IWE Control Flows:

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

- CuramStagingCoreStaticFlow.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 should now be populated with the data that was in the Source Application Database Tables.

6.9 Oracle Environment

This section describes the steps required to install and setup the reporting solution for Oracle. An assumption is made that the developer is familiar with the tools for the Oracle database platform.

There are two possible configuration options when setting up Reporting for Oracle:

1. The Reporting component, OWB client and Oracle server (hosting the Repository owner and Reporting schema's) reside on one machine.
2. The Reporting component and OWB client reside on one machine, and the Oracle server (hosting the Repository owner and Reporting schema's) reside on a remote machine.

When setting up a Reporting sand box environment **Option 1** is recommended as it is a more straight forward and simple configuration. However, if you are constrained by your local configuration and must use an existing Oracle server then **Option 2** is possible.

Note: Regardless of which option is chosen the Staging and Central schemas must exist on the same Oracle instance. We also recommend that the Datamart schema is put on the same instance.

6.9.1 Install Oracle Database and Oracle Warehouse Builder

Installation of the Oracle Database follows that of a typical installation accepting all the defaults.OWB will automatically be installed along with the Database. Please ensure that the install user has administration rights to the install machine.

If installing across two machines, and installing Oracle 11g, it is recommended to install Oracle 11g onto the Server machine first. This will automatically install the Oracle Database and OWB.

Then install the Oracle Client onto the Client machine.

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

Once the Oracle database and OWB have been successfully installed the below setup steps must be completed.

The **BIApplication.properties** and **BIBootstrap.properties** files need to be created in **..\Reporting\project\properties.** These files will contain the database connection details and other variables. Please refer to **Appendix N** for security details

BIApplication.propertiessampleora and BIBootstrap.propertiessampleora have been provided for guidance. They can be found in..\Reporting\project\properties. These files can be copied and renamed as BIApplication.properties and BIBootstrap.properties as a start point. They must be kept in the same folder. Please refer to **Appendix A and B** for more information when setting the properties and following the below steps.

The BIApplication.properties, BIBootstrap.properties and Environment Variables setup depends on which type of configuration option you have chosen, i.e. OWB Client present on Oracle server or no OWB client on Oracle server. However there are a number of common steps which are outlined first.

Note: For each of the Reporting build tasks the equivalent manual steps that could be used are outlined; these may be useful if any of the build targets do not run successfully due to environment configuration issues.

6.9.2.1 Common Setup Steps

Please complete these steps, which are common to both types of configuration. Refer to Appendix B for full list of variables specified below:

1. In the BIApplication.properties file ensure the environment.owbconfig.remotedatamanagerdir variable is set correctly.

   If the Data Manager folder is on the local machine then leave it blank or it will cause an error.

   If the staging, central and datamart schema's are being created on a remote server then the Data Manager folder and contents will need to be copied to this server after the build staticdata command has been run, refer to Section **'Create OWB Repository and target schema's',** Step 14. You then need to set the environment.owbconfig.remotedatamanagerdir variable to the path of the Data Manager folder on the remote server, using the java convention for path separators (\\),e.g. Reporting\\bin\\data_manager\\. Please ensure that the trailing \\ is added or it will cause an error.
2. In the BIApplication.properties file ensure the environment.owb.oracleinstalled variable is correctly set. E.g. environment.owb.oracleinstalled=true. Only set this to false if there is no Oracle database on the server where OWB has been installed. Otherwise set this to true.

3. If setting up a Development Environment, please ensure that the 2 BIApplication.properties autogrant variables are set to true. Please see Appendix G for more information, and also for information on setting up the privileges on non-development environments.
4. If setting up a Development Environment, please ensure that the variable environment.databases.curam.updatenulls.autorun is set to true. Please see Appendix H for further information on last-written columns and setting up privileges on non-development environments.
5. Ensure that the following are included in the components\BIBuildTools\lib-ext directory:

   ant.jar

   ant-contrib-0.6.jar
6. Ensure the following environment variables exist, are correctly set and it is very important that they are spelled correctly:

   REPORTING_DIR - should be set to the location of the Reporting directory, e.g. C:\IBM\Curam\development\Reporting

   ANT_HOME - e.g. C:\apache-ant-1.8.2 or C:\ant182

   OWB_HOME - e.g. C:\oracle\product\11.2.0\db_1\owb for Oracle 11gR2. Ensure that when this environment variable is set that the path %OWB_HOME%\bin\win32 is correct. (OWB_HOME dependencies: OMBPLUS, loadjava.bat & jdk).

   JAVA_HOME - ensure this is set to the jdk of the java location. e.g. C:\jdk1.6.0. This MUST be java 1.6 or higher.

   JAVA_HOME_RDBMS - should be set to %ORACLE_HOME%\jdk.

   CURAMSDEJ - should be set to C:\IBM\Curam\development\CuramSDEJ

6.9.2.2 OWB client installed on the Oracle server

If Oracle and OWB are installed on the same machine then:

1. Set ORACLE_HOME Environment Variable - E.g.:
   - D:\app\product\11.2.0\dbhome_1 for Oracle 11gR2

6.9.2.3 OWB client is not installed on the Oracle server

If the Oracle Server and OWB Client are not on a single machine please read the Oracle Warehouse Builder Installation and Administration Guide to ensure a valid Oracle/OWB configuration exists.

When the reporting build commands are running on a machine that only has the OWB Client installed i.e. no Oracle instance, then:

1. Set ORACLE_HOME to the parent directory of OWB_HOME. This is to enable the build scripts to find the loadjava.bat and javac.exe files. (ORACLE_HOME dependencies: loadjava.bat & jdk).
2. PATH - ensure %ORACLE_HOME%/bin is at the beginning of the Path Environment Variable, with %JAVA_HOME%\bin next.
3. In the BIApplication.Properties file set the environment.jdbc.jar to the path of the jdbc driver that contains the driver defined in the driver properties =
   - (Path in OWB machine)\ojdbc5.jar for Oracle 11gR2

   E.g.
   - C:\\oracle\\product\\11.2.0\\db_1\\jdbc\\lib\\ojdbc5.jar

6.9.2.4 Bootstrap your environment

All of the remaining properties contained in the files BIApplication.properties and BIBootstrap.properties need to be configured. Please see **Appendices A and B** for information about each property and **Appendix N** for security details.

Once the properties have been fully set, please follow these steps to test your setup:

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

3. Run the 'init' command

This compiles the source files. If this fails verify that the contents of the BIApplication.properties file are correct and refer to the troubleshooting section.

6.9.3 Create OWB Repository and target schema's

This section outlines the steps for creating the Reporting schema's i.e. Repository Owner and the target schema's for Staging, Central and Datamarts that need to be created.

1. In order to change the language used in localized property names which is defaulted to English, then please modify the **Component.locale.order.installedLanguage** variable in BIApplication.properties and also modify the language code in the 3 initialdata.sql files (Please refer to Appendix L on Globalization for full details)
2. Update all Properties in BIBootstrap.properties, which can be found at..\Reporting\project\ properties. Ensure schema names match the project naming standards. Each user name is a schema:

   staging.db.username=CuramST

   central.db.username=CuramDW

   centraldm.db.username=CuramDM
3. Additional usernames for Oracle 11gR2:

   design.db.username=CuramBI

   runtime.db.username=CuramBI

   design.db.workspacename=CuramBI

   The variables design.db.username and runtime.db.username need to be the same name. Please ensure that they match. These have been named to keep in accordance with other usernames but we highly recommend that you standardize the names to your own preference.
4. In the BIBootstrap.properties file please ensure that all the xxx.db.name properties are set to be the Oracle Net Service Name, e.g.

   staging.db.name=orcl

   The Oracle Net Service Name is found in the tnsnames.ora file (%ORACLE_HOME%\NETWORK\ ADMIN) and the Net Service Name is in bold:

   ORCL = (DESCRIPTION =(ADDRESS = (PROTOCOL = TCP)(HOST = server name)(PORT = 1521))(CONNECT_DATA =(SERVER = DEDICATED) (SERVICE_NAME = **ORCL**)))
5. In the case that Database User OWBSYS is locked, the Database User OWBSYS can be unlocked using either SQL Developer or by entering the following commands in a command prompt opened in **..\Reporting\components**:
   - 'sqlplus'
   - 'sys as sysdba'
   - password (will not appear, just click click enter once complete)
   - 'alter user owbsys identified by `password` account unlock'
6. Prior to creating the Workspace, Workspace owner and Reporting BI schemas, if the users local naming conventions are different to the standard naming conventions which are pre-set in \Reporting\components\project\properties\BIBootstrap.properties then manually update the username and/or workspace names in the BIBootstrap.properties file. Please refer to Appendix M for more advanced updates/additions which can be made to the the Reporting BI schemas other than the names.
7. Start a command prompt and navigate to.. **.\Reporting\components.** Run the 'build database.create.bischemas' command to create the 3 Reporting BI Schemas. This will create the Staging, Central and Datamart Schemas on the Database and connects to the Database using the User Name and Password specified in the file..\Reporting\components\BIBuildTools\scripts\ rep_oraschemas.properties. They are defaulted to SYS as SYSDBA and p. Please set them as required before running the command.

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

   Oracle Version 11gR2

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

   Step 2:Operation - Select Manage Warehouse Builder Workspace' option and click next

   Step 3:Workspace Operations - Select 'Create a new Warehouse Builder workspace' option and click next

   Step 4:New or Existing User - Select 'Create a workspace with a new user as workspace owner' option and click next

   Step 5:DBA Information - Enter username (e.g. sys) and password and click next

   Step 6:Workspace Owner - Enter workspace owner's username, e.g. CuramBI, password and workspace name, e.g. CuramBI (Workspace details should match the workspace details in BIBootstrap.properties file) and click next

   Step 7: In the OWBSYS information window, enter the OWBSYS username and password (This will not appear in subsequent runs of the Repository Assistant)

   Step 8: In the Select Languages window, accept the defaults and click Next (This will not appear in subsequent runs of the Repository Assistant)

   Step 9: Select Workspace Users - Just click next here.

   Step 10:Summary - Review the Summary and click finish to build the Workspace and register the Target Schemas.
9. If setting up a Development Environment, please ensure that the 2 BIApplication.properties autogrant variables are set as follows:
   a. Environment.databses.curam.privaleges.autogrant should be set to true if the Curam source data is from a database that resides on the same database instance as the staging database. Otherwise it should be set to false.
   b. Environment.databases.bi.privilages.autogrant should be set to true

   Please see Appendix G for more information, and also for information on setting up the privileges on non-development environments.
10. From your local environment, add \product\11.2.0\dbhome_1\jdbc\lib\ojdbc5.jar file to Reporting\components\BIBuildTools\drivers directory
11. Open a command prompt and navigate to...\Reporting\components. Run 'build configtest'. This will test if everything is installed correctly.
12. Run 'build owb.environment.tests.import' - this imports the Locations, meta data and 6 ETL's to test if they run correctly.
13. Log into the OWB Design Center and expand the Reporting Project in the Project Explorer / Projects Navigator panel.
14. For Oracle 11gR2: In the Locations Navigator window, double click on the DEFAULT_CONTROL_CENTER panel in Control Centers folder. Click on Data locations and ensure all the following locations:
    - DEFAULT_AGENT
    - SOURCE_LOCATION
    - STAGING_LOCATION
    - STATIC_DATA_LOCATION
    - DATAMARTS_LOCATION
    - CDW_LOCATION

    are in the 'Selected Locations' with both Source and Target box's ticked for all except STATIC_DATA_LOCATION which should have just the Source box ticked.

15. Open the Control Center in Tools-Control Center Manager, to check that all 5 of the Locations are listed.
16. Log out of OWB and click Yes to Save the changes.
17. Open Oracle - Application Development - SQL Developer. On first use, you'll be asked to chose connection for java.exe. This should be located in the java bin folder. e.g.: C:\Program Files\Java\jdk1.5.0_06\bin. Right click on Connections, choose new connection and fill in the following details for Source, Staging, Central and Datamarts:
    - Connection Name
    - Username
    - Password
    - Hostname
    - Port
    - Service name
18. Open a command prompt and navigate to..\Reporting\components. Run 'build owb.environment.tests.run'.

    This builds the database for each schema, and deploys 2 ETL's from each schema and tests the environment has been correctly configured. When this step executes successfully check the following tables and ensure they are populated with data:
    - S_ETLCONTROL
    - DW_ETLCONTROL
    - DM_ETLCONTROL

    If there is data in the source database then the following tables should also be populated:
    - DM_DIMCASETYPES
    - S_CODETABLEITEM
    - DW_CASETYPE

    This confirms that the environment has been correctly configured. This step may between 5 and 20 minutes to execute.

    Please note that the previous command executed an 'build staticdata'. This copies/merges the static data files and the control files to the...\Reporting\bin\data_manager directory. Note: IF the OWB client is not installed on the Oracle server then this static data will need to be manually copied to the Oracle server and ensure the Static Data File path points to that location.

    It also executed an 'build grant.all' command. This grant.all command grants permissions to the staging, central and datamart schema's in order for the ETL's to run successfully. In order for this grant.all to execute successfully the source, staging and central users must have the correct grant authority. Please see Appendix G for more information, and also for information on setting up the privileges on non-development environments.

    It also executed an 'build database.source.updatenulls' command. This command updates the last written column values in the Application Tables if they are null. This ensures that all of the records are extracted from the Source Tables into the Warehouse. If the last written columns are null then the records will not get loaded as they will not pass the join to the Control Table in each ETL.
19. If the above tables have data in them then your Data Warehouse is set up correctly!
20. Please follow the Steps in the next Sections to Import the Meta Data into OWB, Deploy the ETL's and Run them.

6.9.4 Importing the Meta Data

The source code is imported into OWB using the build commands below.

1. Open a command prompt and navigate to...\Reporting\components. Run "build owb.import.all". It may take up to an hour for these commands to complete.

    Please note that this command in turn runs each of these commands:

- "build owb.import.common"
- "build owb.import.source"
- "build owb.import.staging"
- "build owb.import.central"
- "build owb.import.datamarts"

6.9.5 Deploying the Meta Data

The source code is deployed from OWB to the Database using the build commands below.

1. Open a command prompt and navigate to...\Reporting\components. Run "build owb.deploy.all". It may take up to an hour for these commands to complete.

   Please note that this command in turn runs each of these commands:

   - "build owb.deploy.runtimesetup"
   - "build owb.deploy.staging"
   - "build owb.deploy.central"
   - "build owb.deploy.datamarts"
2. It is necessary to verify via the control center that these commands ran successfully.
3. Open the OWB Design Center, go to the tools menu and click Control Center manager. Note it is necessary to expand the Reporting project otherwise this option will be grayed out. The right-hand pane displays the deployment details with the deployment status. Ensure that the green tick is present for all the ETL's that have been deployed.

   Please ignore the Drop Errors - these occur because OWB tries to drop the ETL's before deploying them and they do not exist on the database yet.
4. In order to check that the deploy commands have executed successfully carry out the following for each schema.
5. Expand the mapping sections for the Staging schema; right click on the S_ETLCONTROL_ETL and start. If this executes successfully the S_ETLCONTROL table will be populated.

   Please see the Troubleshooting Section for issues on the deploy command.

6.9.6 Loading Data

It should be noted that the execution scripts shipped for the Oracle environment are present to help developers and testers executing ETL processes. These are not intended for production environments.

1. Open a command prompt and navigate to...\Reporting\components. Run "build run.staging.all". Check that the staging tables have data. Specifically the S_ETLCONTROL table should have date/time stamps.
2. Open a command prompt and navigate to...\Reporting\components. Run "build run.central.all". Check that the central tables have data. Specifically the DW_ETLCONTROL table should have date/time stamps.
3. Open a command prompt and navigate to...\Reporting\components. Run "build run.datamarts.all". Check that the datamart tables have data. Specifically the DM_ETLCONTROL table should have date time stamps.
4. Open a command prompt and navigate to...\Reporting\components\Core. Run "build run.datamarts.operational. aggregates". Note that the purpose of this command is to run the aggregate ETL's on a monthly basis. As a result this command will only execute on the last day of a month.

   This command executes the following ETL's:

   - DM_AGGCASEMONTH_ETL
   - DM_AGGCASEDAY_ETL
   - DM_AGGFUNDS_ETL
   - DM_AGGPAYMENTS_ETL

Before running them, please set the environment.datamart.aggmonth.start/end dates to the required dates in the application.properties file, otherwise the ETL's will run for every month from 1934 till now (Default date set in Application.properties file).

5. The command "build run.all" aggregates the commands in point 1 to 3(above). Use this command on a regular basis (e.g. nightly).

Note : If the ETL processes were not deployed successfully they cannot execute correctly. Please note that any messages from the build targets stating "deployed successfully" may be misleading. OWB does not report errors in deployment when deploying from the command line, use the control center to verify all ETL processes are successfully deployed.

Appendix C "Build Script Commands" contains an overview of the build commands available with regards to executing ETL processes. The command "run.all" runs all ETL processes, please note that ETL processes are divided into 2 groups:

1. **Control ETL Processes:** These are ETL processes that do not need to be executed with each batch run, and possibly should not be executed with each batch run (depending on your requirements). The control ETL process populate control tables which only need to be populated once, and thereafter the tables only need to be populated when new ETL processes are added to the system, other control ETL process only update table which contain data that changes infrequently
2. **Operational ETL processes:**These are the ETL process that must be executed for each batch run as the extract from the transactional tables within the Curam database schema.

If your primary requirement is to verify operational data extracts, we recommend that the following jobs be executed as part of your batch run:

- run.staging.operational
- run.central.operational
- run.datamarts.operational

Chapter 7. Customizations and Upgrades

7.1 Introduction

Customers who take on BIA Reporting will customize the system in some way. As it is extremely important that customizations continue to work when an upgrade is applied, guidelines are provided in the manner in which customizations should be made. The upgrade approach takes these guidelines into account and does not overwrite functionality which has been properly customized.

This chapter details how to make changes to Reporting and how to take on upgrades to Reporting. The customization and upgrade process is described with the use of examples. The upgrading process clarifies why BIA Reporting is customized in the recommend fashion, so it helps to read the customization and upgrade sections together.

7.2 Customizations

There are two primary reasons why customers will have a requirement to customize the Reporting Module. The first is that customers may want to include fields and entities in Reporting that are not included in the 'out of the box' functionality and the second is that they may have already extended the Core Data Model and they want to include the additional data in Reporting:

- **Include additional core data in BIA Reporting**. The source Data Model consists of hundreds of entities. Only a subset of these entities and a subset of the fields within these entities are extracted to the Data Warehouse. In the event that a Customer requires additional data in the Warehouse they will have to customize some of the reporting artifacts to include the data in the Warehouse.
- **Include additional extended data in BIA Reporting**. Customers may extend the core data model to cater for additional information that they need to capture. The data in the extensions will not be automatically propagated through to the warehouse and customers will have to customize many of the Reporting artifacts to include the data in the Warehouse.

The folder structure provided with BIA Reporting consists of both a **core** folder and a **custom** folder. When a customer site wants to modify metadata such as an ETL or DDL they should copy the core file, place it in the custom folder and make modifications in this custom folder. Build scripts will then give precedence to the contents of the custom folder.

The core folder should never contain artifacts modified by the customer. This is necessary for future upgrades (see separate section below on *Upgrades*) of BIA Reporting.

The following steps give the modifiable artifacts delivered with Reporting together with the customization guidelines. Example scenarios in **Appendix C** detail typical customizations that customers will be faced with. The three examples provided detail the user report change required, the methodology used for determining the required changes, and finally a listing of the changes required.

7.2.1 Customizing Data Models and ETL Designs

The Data Models are a pictorial representation of each of the databases. As a customer makes modifications to the physical database they should also make changes to the data model so that they are kept in sync with the physical database.

The ETL process flows are pictorial representations of the activities within an ETL process. As a customer makes modifications to the physical ETL processes they should also make changes to the process flows so that they are kept in sync with the actual ETL process.

The data models and ETL designs are delivered as part of the documentation center. Please refer to the Documentation Center guide for customization guidelines.

7.2.2 Customizing DDLs

One set of DDLs are delivered per database and each consists of the code necessary to setup the tables, indexes, views, constraints and sequences. The DDLs are modifiable. When customizing the customer will make modifications in the folder named *custom\core\ddl*. The build script will give precedence to the contents of the custom folder over the core folder.

7.2.3 Customizing ETL Meta Data

The ETL meta data is data about the ETL process. The ETL meta data is modifiable. When a customer needs to make an alteration to an ETL they do this by altering the ETL in the ETL tool and saving the ETL to the *custom\core\etl* folder. All modified ETL's are saved to the *custom\core\etl* folder instead of the *core\etl* folder. The build script will give precedence to the contents of the custom folder. When exporting from the ETL tool to the custom\core\etl folder the customer will separate each ETL into a separate file instead of exporting to one, single file. The purpose of this is to allow concurrent development and to make it easier to import/export individual ETL's to the ETL tool.

7.2.4 Customizing the Build Script

The Build scripts are delivered as Ant scripts and are in xml format. A customer can extend the build scripts to add in their own functionality e.g. the customer may wish to copy files to a backup folder after creating the schemas.

If a customer wants to amend or create additional tasks we recommend the following process:

1. the customer creates a new ant file, e.g. components\core\mybuild.xml file
2. the customer adds any new tasks into mybuild.xml
3. call the new build commands through the build.bat by importing the new build file, e.g. by editing component\core\build.xml adding the line <import file="mybuild.xml"/>
4. or by creating a new batch file to execute the new build commands, e.g. copy our component\core\build.bat to mybuild.bat and ensure <mybuild.xml> is referenced

7.2.5 Customizing the Java Transformation Code

Java transformation code is written when the type of transformation required is not supported by the ETL tools. The java transformation code is configurable but not modifiable from the *core* directory. When configuring existing transformation code the customer will make changes in the folder named *custom\source*. Customers may add their own transformation logic in this folder also and this code will be then referenced from an ETL. The build script will give precedence to the contents of the custom folder.

7.2.6 Customizing the Static Data Files

Static data files are used to populate entities such as the time dimension in the datamarts. These are delivered to the customer as comma separated files -.csv format. Any modifications will be made in the folder *custom\core\data_manager* and not in the *core* folder. For example, the customer may need to change the months in the fiscal calendar for the time dimension or change the fund data to match existing products. The build scripts will give precedence to the custom folder.

7.3 Upgrade Process

This section describes the process when a customer upgrades to a newer version of BIA Reporting. This process is described with some examples. This section should be read in conjunction with the customization section as upgrades will be easier if the customization process is followed.

As described in the customization section no modifications by customers to BIA Reporting are made to the core artifacts; instead any modifications are made in the *custom* folder by copying the original artifact to this folder. This means that when artifacts in the core folder are replaced with the newer upgraded metadata no customized artifacts are replaced. The build script will always give precedence to the contents of the custom folder.

When the original BIA Reporting artifacts in the core folder have been replaced with the upgraded artifacts, the customer must then begin the process of upgrading the customized artifacts. The first step is to isolate an upgrade strategy for each artefact; once complete the customer can make the necessary changes to artifacts that need to be updated. This is described in the upgrade strategy section.

7.3.1 Upgrade Strategy

When the customer upgrades the Reporting Solution the first step is to review the **Reporting Upgrade Documentation** to review the upgrade and get a list of all artifacts that have changed. Once the customer has a complete list of the changed artifacts a review of each artefact is needed to decide if a change is required. Once the review is complete any required changes are made. The following scenarios illustrate the different strategies which can be taken during an upgrade:

7.3.1.1 Scenario 1: No upgrade for the artefact

In this scenario the artefact in the core folder has not been changed to a newer version by the upgrade. If the customer uses a modified version of this artefact in the custom folder or the original version in the core folder the results are the same: there is no change required.

7.3.1.2 Scenario 2: Artefact has been upgraded; customer wants to use the upgraded artefact and is currently using the core artefact

In this scenario the artefact in the core folder has been replaced with the upgraded artefact and the customer wants to use this artefact. The customer is currently using the core artefact i.e. no changes have been made to the artefact by the customer. No modifications need to be made to the artefact as the customer will now use the upgraded artefact in the core folder.

An example of this is a change to the DW_ADDRESS_ETL meta data. There has been an upgrade to this ETL to include a new destination field. Since the customer has not modified this ETL before the upgrade and wants to use the upgraded version then the customer can import this upgraded ETL into the ETL tool and use this instead of the pre-upgrade version.

7.3.1.3 Scenario 3: Artefact has been upgraded; customer does not want to use the upgraded artefact and is currently using the core artefact

In this scenario the artefact in the core folder has been replaced with the upgraded artefact and the customer does not want to use this artefact. The customer is currently using the core artefact i.e. no changes have been made to the artefact by the customer. The customer must keep the pre-upgraded version in the custom folder as this is now different from the core solution. The build script will always take the artefact in the custom folder before the core folder.

An example of this is a change to the DW_ADDRESS_ETL meta data. There has been an upgrade to this ETL to include a new destination field. Since the customer wants to use the original pre-upgraded version without the new destination field they must keep the pre-upgraded version of the DW_ADDRESS_ETL meta data in the *custom\core\etl* folder. They can do this by exporting the ETL from the ETL tool to this folder.

7.3.1.4 Scenario 4: Artefact has been upgraded; customer wants to take on the upgrade and is currently using a custom artefact

In this scenario the artefact in the core folder has been replaced with the upgraded artefact and the customer wants to take on these upgrade changes. However, the customer has already made changes to the artefact which is in the custom folder. The customer must manually make the upgraded changes to

the customized artefact. To do this the customer must check for differences in the new upgraded artefact (a comparison utility may be helpful to compare certain artifacts) and adopt these upgrade changes into the customized artefact.

An example of this is a change to the DW_ADDRESS_ETL meta data. There has been an upgrade to this ETL to include a new destination field. However the customer has already modified the meta data by deleting an existing column which they do not need. To adopt the upgrade the customer must check for the differences with the new upgraded artefact to find the new destination column. The customer can then modify their customized ETL in the custom folder to add this destination field. The build script will always take this meta data in the custom folder before the core folder.

7.3.1.5 Scenario 5: Artefact has been upgraded; customer does not want to take on the upgrade and is currently using a custom artefact

In this scenario the artefact in the core folder has been replaced with the upgraded artefact but the customer does not want to take on these upgrade changes. The customer has already made changes to the artefact which is in the custom folder. No modifications need to be made as the build script will use the custom artefact and ignore the upgraded artefact in the core folder.

An example of this is a change to the DW_ADDRESS_ETL meta data. There has been an upgrade to this ETL to include a new destination field. However the customer has already modified the meta data by deleting an existing column which they do not need. Because the customer does not want to adopt the upgrade no modifications need to be made and the build script will still use the ETL in the custom folder and ignore the upgraded ETL in the core folder.

7.4 Conclusion

Customers who take on BIA Reporting will always customize the system in some way and will also need to take on upgrades to BIA Reporting. Guidelines are provided on the manner in which customizations and upgrades should be made and these were discussed in this chapter. The chapter contained the following:

- an overview of the BIA Reporting customization process.
- a listing of all the modifiable artifacts in BIA Reporting and an explanation on how each type of artifact should be customized.
- an overview of the Upgrade process.
- the upgrade strategy chosen for each type of upgrade scenario encountered.

Chapter 8. Troubleshooting

8.1 Introduction

This chapter details possible troubleshooting tips and fixes that the user may find helpful.

8.2 Problem: Build fail errors when running the build script

8.2.1 Problem

When using the build script to create a database schema or import metadata into the ETL tool the error 'BUILD FAILED' occurs with a description.

The build script may have failed for a number of reasons such as the applications.properties\ BIbootstrap.properties files not set up correctly or the database environment variables are not set up correctly.

8.2.2 Solution

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 ' **appbuild configtest** ' in the command prompt from the directory that the build script resides will test to check if the reporting environment is setup correctly. The results of running this command are displayed including a message indicating if the environment has been setup correctly (BUILD SUCCESSFUL) or not (BUILD FAILED). The exact error message for failing is also displayed. Should a command not run successfully, then prior to the 'BUILD FAILED' message, an error message report would specify where the issue needs to be fixed. An example of one of these error message reports is as follows:

```
----Report Start--------------------------

you should investigate the following issues:

no informational issues Error(s),

you MUST fix the following issues:

Error: please set JAVA_HOME=

----Report End----------------------------
```

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

8.3 Problem: 'appbuild configtest' is failing

First and foremost if the appbuild configtest fails for any reason verify that the contents of the application.properties file is correct.

8.3.1 Problem

'appbuild configtest' is failing with this error:

- C:\Curam\development\Reporting\components\core>appbuild 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

8.3.2 Solution

- Verify that the contents of the application.properties file is correct and copy a tools.jar file into the C:\ProgramFiles\Java\jre1.5.0_09\lib location
- Ensure JAVA_HOME is set to the JDK of Java 1.5 or higher

8.3.3 Problem

'appbuild configtest' is failing with the below error:

- 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

8.3.4 Solution

- Ensure JAVA_HOME is set to the JDK of Java 1.5 or higher.

8.3.5 Problem

'appbuild configtest' is failing with this error:

- [echo] info:Compiling class using RDBMS compiler, using ${java.path}\javac.exe

8.3.6 Solution

- Ensure JAVA_HOME points to a JDK home and not a JRE home.

8.3.7 Problem

'appbuild configtest' is failing with this error:

- 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

8.3.8 Solution

- Ensure the OWB_HOME var is set correctly, e.g. ...\oracle\owb

8.3.9 Problem

'appbuild configtest' is failing with this error:

- jar: [echo]

  info: Compiling class using RDBMS compiler, using ${java.path}\javac.exe [javac] Compiling 87 source files to

  %REPORTING_DIR%\devenvironment\build\rdbms

8.3.10 Solution

- Ensure the application.properties file exists and is correctly configured. Specifically the database type property.

8.4 Problem: Errors with permissions

8.4.1 Problem

In the OWB Control Center an ETL fails to deploy and gives this error message:

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

8.4.2 Solution

Running this command may resolve the problem - 'appbuild grant.all'.

This command grants permissions from staging, central, and datamarts to public. It should only be used in a Development Environment.

8.5 Problem: The lastwritten field in source table is not populated

8.5.1 Problem

When attempting to extract data from a source table to the Staging database using an ETL no data is being extracted. The problem is that the last written field in the source database is not being populated.

8.5.2 Solution

All source tables being used in BIAReporting need the last written field to be populated. The developers of the source system must ensure that this is a mandatory field and always updated.

The last written field in the source table needs to be populated, the date in the control table will need to be reset and the ETL run again.

8.6 Problem: Error deploying the locators in Oracle because of Oracle version

8.6.1 Problem

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

8.6.2 Solution

The user needs to change the database version in the locators in Oracle Warehouse Builder (OWB). In the Reporting project in OWB the user needs to select the *database/oracle/locations* folder and open the properties of each location and change the version. If the version looks to be correct then change to an older version and commit and then change back and commit again.

8.7 Problem : Unwanted OWB Locations

8.7.1 Problem

The names of the location objects in OWB are related to the schema names in the application.properties file. Every client applies their own naming conventions when creating the schemas for staging, central and datamarts. It is therefore necessary to perform some clean up operations on the OWB location objects created.

8.7.2 Solution

1. Open the OWB Design Center and log in as the Runtime Owner.
2. In the Connection Explorer, in the right hand panel, expand the Databases and Oracle options. It may be necessary to delete the following locations: Any duplicate locations prefixed with "curstaging".
3. Please do not delete the following:
   - OWB locations which are prefixed with "OWB" or "REPOSITORY"
   - Reporting locations which are SOURCE_LOCATION, STAGING_LOCATION, CDW_LOCATION, DATAMARTS_LOCATIONS and Static Data.

8.7.3 Necessary locations

The locations which should remain are

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

Please note that the locations will be registered correctly when the command md.deploy.runtimesetup is run in Section Websphere Application Server Post-Installation Configuration'

8.8 Problem: Error when running 'appbuild transform.aggcasemonth'

8.8.1 Problem

If the transform.aggcasemonth build task fails run the 'appbuild configtest' task to ensure the project properties are correct. If the appbuild configtest fails with the below error:

- Error, ant property <environment.jdbc.jars> can't find file C:\oracle10\product\10.2.0\db_1\jdbc\lib\classes12.zip

8.8.2 Solution

Then navigate to the application.properties file (...\Reporting\project\properties) and ensure the environment.jdbc.jars is correctly set to the Oracle home directory. (Can vary slightly from different machines)

- environment.jdbc.jars=C:\\oracle\\product\\10.2.0\\db_1\\jdbc\\lib\\classes12.zip
- Rerun 'appbuild configtest', this should now run successfully and then run 'appbuild transform.aggcasemonth' which should also successfully complete.

8.9 Problem: Cognos does not start

8.9.1 Problem

Errors thrown in Cognos Configuration preventing Cognos from starting.

8.9.2 Solution

Cognos requires a Sun JRE which must be 1.5. Ensure the JAVA_HOME points to the Cognos JRE or a compatible JRE.

8.10 Problem: Errors while importing the ETL's

8.10.1 Problem

Errors thrown when importing the ETL's from the command line.

8.10.2 Solution

If there are issues importing the ETL's, check the owbimport.tcl script located here (%REPORTING_DIR %\bin\etl) and verify the OMBCONNECT string on the first line is correct.

8.10.3 Problem

Cannot create connection to control center

8.10.4 Problem

OWBSYS is locked (This relates to any user)

8.10.5 Solution

In SQL developer: Sys>Other Users>OWBSYS -right click and choose edit user. Un-check 'Account is locked. Click Apply and close

8.10.6 Solution

Oracle>Warehouse Builder>Administration>Start Control Center

8.11 Problem: Unable to log into Control Center

8.11.1 Problem

The following error is thrown when trying to log into the control center: RTC-5260: Failed to connect to runtime platform, Please check you have provided the correct Host, user, password and service name

8.11.2 Solution

Navigate to All Programs->Warehouse Builder->Admin and click on Start control center service. Go back and log into control center again.

8.12 Problem: No Data in Tables after running owb.environment.tests.run

8.12.1 Problem

There is no data in the Tables after running owb.environment.tests.run. There may be a number of reasons for this issue, but if this error message is in the screen output then please see the below solution:

- run.etl.execute:
  [exec] ERROR:
  [exec] ORA-01017: invalid username/password; logon denied

8.12.2 Solution

appbuild 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.

Please ensure that they match and that they are correct.

8.13 Problem: Error reimporting due to Matching Conflicts

8.13.1 Problem

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

8.13.2 Solution

In order to permanently fix this bug, clients must install patch 10195667 which can be found by logging into https://support.oracle.com/CSP/ui/flash.html. The instructions in the Readme.txt file provided with this patch are complicated so a more clearer version is provided in Appendix K: Installing Patches

8.14 Problem: Error reimporting due to Matching Conflicts

8.14.1 Problem

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

8.14.2 Solution

In order to permanently fix this bug, clients must install patch 10195667 which can be found by logging into https://support.oracle.com/CSP/ui/flash.html. The instructions in the Readme.txt file provided with this patch are complicated so a more clearer version is provided in Appendix K: Installing Patches

8.15 Problem: Error reimporting due to Matching Conflicts

8.15.1 Problem

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

8.15.2 Solution

In order to permanently fix this bug, clients must install patch 10195667 which can be found by logging into https://support.oracle.com/CSP/ui/flash.html. The instructions in the Readme.txt file provided with this patch are complicated so a more clearer version is provided in Appendix K: Installing Patches

Appendix A. Configuring the BIBootstrap.properties file.

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

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

The following lists the main components in the properties file:

- **Database Type** - Set this for Oracle (db.type=ORA) or DB2 (db.type=DB2)
- **DB2 Source Type** - Only set for DB2. The type must be set to UDB if using DB2 Universal Database (db2.source.type=UDB)
- **Connection information for the target databases** (Source, Staging, Central, Datamarts). The server, database name, username, SID, port number (if applicable) and password must be set. We will use the target database (staging, central, or datamarts) set up in the previous section as the username. Note: For DB2 Central and Staging should be set up on the same database.
- **Connection information for the Demo Data database** (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) should be set. For Oracle we will use the design time repository user previously set up as the username. For DB2 we will use the control database automatically set up as part of the install as the username.
- **Run time repository connection information (Oracle Only).** The server, port (if applicable), database name, username and password and service name (if applicable) should be set. For Oracle we will use the runtime repository owner previously set up as the username. For DB2 we will use the control database automatically set up as part of the install as the username.
- **Password -** ***passwords for Oracle, DB2, WLS and WAS need to be encrypted in the BIBootstrap.properties as follows:***
  - Open a command prompt from Reporting\components
  - Run **appbuild 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

A.1 Sample BIBootstrap property file for DB2

#Note: These database usernames are samples. We highly recommend that you use you standardize the names to your own convention

The 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==

A.2 Sample BIBootstrap property file for Oracle

#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==

A.3 Sample BIApplication property file for Oracle

bi.bootstrap.id=local development #1

component.order.warninglevel=error component.order=core,childservices

environment.iexplorer.url=file://C:/Program_Files/Internet_Explorer/iexplore.exe

environment.variables=REPORTING_DIR,REPORTING_ENV,COGNOS_HOME,ANT_HOME, DB2DIR

environment.resetetl.date=01/01/1934:00:00:00

environment.resetetl.dateformat=dd/mm/yyyy:hh:mm:ss

environment.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/yyyy

environment.jdbc.jars=F:\\app\\dwtesting\\product\\11.2.0\\dbhome_1\\jdbc\\lib\\ojdbc5.jar

environment.owbconfig.remotedatamanagerdir.failonwarnings=false

environment.owb.oracleinstalled= true

environment.owbconfig.version=11.2

environment.owbconfig.version.sourceDB=11.2

environment.databases.curam.privilages.autogrant=false

environment.databases.bi.privilages.autogrant=true

environment.databases.curam.updatenulls.autorun=false

Appendix B. Configuring the BIApplication.properties file.

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

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

The following lists the main components in the properties file:

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

  If the staging, central and datamart schemas are being created on a remote server then the Data Manager folder and contents will need to be copied to this server after the build staticdata command has been run, refer to Section 6.8.3, Step 12. You then need to set the variable to the path of the Data Manager folder on the remote server, using the java convention for path separators (\\),e.g. Reporting\\bin\\data_manager\\. Please ensure that the trailing \\ is added or it will cause an error.
- **Environment.owbconfig.version** - Set this to the version of Oracle you are using.
- **Environment.owbconfig.version.sourceDB** - Set this to the version of Oracle you are using as the source database, e.g. 11.2.
- **Environment.owbconfig.exectemplate** - Set this to be sqlplus_exec_template.sql.

- **Environment.databases.curam.privilages.autogrant** - This grants select on all Source Tables to the Staging Schema. See Appendix G. This should be set to false.
- **Environment.databases.bi.privilages.autogrant** - This grants select on all Staging Tables to the Central Schema. See Appendix G. This should be set to false.
- **Environment.databases.curam.updatenulls.autorun** - This turns on/off the privilege of granting from Source Applications to Reporting Staging Schemas. See Appendix H. Set this to be false.

Appendix C. Build Script Commands.

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

C.1 Common DB2 and Oracle Commands

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

Build command	Description
Operational ETL Processes	
clean	Removes datawarehouse ddl, scripts, classes and jars.
configtest	Checks that the reporting environment is set up correctly.
database.all	Builds the staging, central, and datamarts schema objects into the databases specified in the applications.properties file.
owb.import.all	Imports source,staging, central and datamart
owb.deploy.all	Deploys staging, central and datamart ETL's
run.all	Runs all the ETL's

Appendix D. Scheduling of ETL Processes

This Appendix outline how the execution of our reporting ETL processes can be scheduled.

For Oracle, the Reporting component comes with the build targets - 'run.xxx' - to execute ETL processes. However, this is provided as a productivity utility to help developers and testers execute ETL processes, we do not recommend that these build targets are used in a production environment.

A production quality scheduling and execution system should be used.

D.1 DB2

The ETL's can be executed in the IWE Design Studio and also in the Websphere Application Server Admin Console.

The Websphere Application Server Admin Console can be used to schedule the execution of the Control Flows.

Schedules can be created in the Websphere Application Server Admin Console by navigating to InfoSphere Warehouse - SQL Warehousing - Processes- Run Processes.

D.2 Oracle

D.2.1 Design time platform

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

D.2.2 Runtime time platform

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

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

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

Appendix E. Create OWB Workspace Owner and Users using Repository Assistant

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

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

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

  Step 2: Select Manage Warehouse Builder Workspace

  Step 3: Select Create a new Warehouse Builder workspace

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

  Step 5: Enter the SYS user name and password

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

  Step 7: Enter the OWBSYS user name and password

  Step 8: Accept the defaults or else change the Tablespaces

  Step 9: Accept the default or else change the Language

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

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

Appendix F. Remove OWB Workspace Owner and Users using the Repository Assistant

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

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

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

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

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

  Step 2: Select Manage Warehouse Builder workspace users

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

  Step 4: Select the Workspace Name

  Step 5: Unregister Warehouse Builder workspace users

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

  Review the Summary and click finish to unregister the Warehouse Builder workspace users.
- If required, manually drop the CuramST, CuramDW, CuramDM, CuramDMO and CuramBIROLE schemas from the database using Oracle Enterprise Manager or SQL Developer.

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

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

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

  Step 2: Select Manage Warehouse Builder Workspace

  Step 3: Select Drop an existing Warehouse Builder workspace

  Step 4: Enter the Workspace Owner's User Name, e.g. CuramBI, and Password.

  Step 5: Select the Workspace Name

  Review the Summary and click finish to drop the OWB Workspace Owner.
- If required, manually drop the OWB Workspace Owner schema from the database using Oracle Enterprise Manager or SQL Developer.

Appendix G. Granting Database Privileges

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

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

- **environment.databases.curam.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

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

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

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

- curam_grant
- s_grant
- dw_grant
- dm_grant

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

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

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

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

Appendix H. Capturing Changed Data

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

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

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

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

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

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

Appendix I. How to Add an Oracle ETL

Please follow these steps to add an ETL:

- Log into OWB Design Center
- Import any new required Tables into OWB
- Create a New Mapping under the appropriate Reporting\Databases\Oracle\Schema Name folder
- Add the Source and Target Tables
- Add the appropriate Control Table and join it to the Source Table to extract the updated records
- Add any other required Joins or Transformations
- Map the Source Columns over to the Target Columns
- Set the Load Properties on the Target Table and its Columns
- Add the Pre and Post Mapping Transformations to update the Control Table
- Validate and Deploy the ETL
- Add a record into the ETLCONTROL Table for the ETL. This can be done manually, as a once off, or by adding a new entry into the appropriate ETLCONTROL csv file and building the database
- Run the ETL to test via the OWB Control Center
- Add an entry into the appropriate run.bat file, which can be found in..\Reporting\components\core\run\oracle. Any new ETL's should be added to a run.bat file in the custom folder, e.g...\Reporting\components\core\custom\run\oracle. This will ensure the new ETL gets run when the ETL's are executed through the build environment
- The ETL and updated Table metadata can be exported from OWB as .mdo files and stored on a file system

Appendix J. Known Issues

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

J.1 Build Environment

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

J.2 Importing in OWB

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

Appendix K. Installing Patches

K.1 Installing Patches

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

K.2 Instructions for applying a once off patch

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

1. Once the patch is downloaded, extract the contents to a directory, for example E:\stage\12345678 where 12345678 is the no. of the patch
2. Add an environment variable OPatch with value %ORACLE_HOME%\OPatch to the system advanced properties
3. Add %ORACLE_HOME%\OPatch; to the Path variable
4. Stop the runtime service by doing the following:
   - Run a command prompt from folder %OWB_HOME%\rtp\sql
   - Login to sqlplus, i.e run "sqlplus"
   - Logon as "sys as sysdba" OR "owbsys" and enter password
   - Execute "@stop_service.sql"
5. Go to Oracle>WarehouseBuilder>Administration>Stop Control Center Service, log on to the Control Center and choose Stop.
6. Ensure all Oracle programs are closed.
7. Run the OPatch Utility in a command prompt from the folder in which it was saved in Step 1:
   - Execute "opatch apply"

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

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

Appendix L. Globalization

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

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

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

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

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

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

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

Appendix M. Initial Oracle Schemas

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

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

- Copy the initoracleschemas.sql file to another location.
- Make the relevant changes/additions required to this new initoracleschemas.sql file
- Open a command prompt from Reporting\components
- Run the command 'database.create.bischemas' using the -D option to point to the updated copy of the initoracleschemas.sql file.

  E.g. >database.create.bischemas -Dschema.createscript=[full path of updated file]\initoracleschemas.sql to run the new script.
- Modifications to naming conventions need only be applied in the BIBootstrap.properties file

Appendix N. Security

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

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

Recommendation:

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

- Reporting/project/properties/BIBootstrap.properties

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

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

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

We also highly recommend the use of encrypted passwords in the BIBootstrap.properties file. Please refer the password section in **Appendix A**

Appendix O. Upgrading to Oracle 11gR2

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

How to Upgrade:

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

Items of Note:

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

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

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

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

Intellectual Property Licensing

Legal and Intellectual Property Law.

IBM Japan Ltd.

19-21, Nihonbashi-Hakozakicho, Chuo-ku

Tokyo 103-8510, Japan

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

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

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

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

Dept F6, Bldg 1

294 Route 100

Somers NY 10589-3216

U.S.A.

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

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

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources.

IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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

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

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

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing

application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

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

Apache is a trademark of Apache Software Foundation.

BIRT is a registered trademark of Eclipse Foundation.

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

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

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

Glossary

Aggregation
Information stored in a data mart in a summarized form. Aggregations are used to save storage space and to improve the performance of the data mart.

Change Data Capture
In database replication, changed data capture occurs when only the data that has changed since the previous replication is copied.

Data Cleansing
The process of removing errors and inconsistencies from data being imported into a data warehouse.

Data Mart
A database, or collection of databases, designed for reporting and to help users make strategic decisions about their business. A data warehouse combines databases across an entire enterprise; data marts are usually smaller and focus on a particular subject or department.

Data Warehouse
A database created from operational extracts that adhere to a single, consistent, enterprise data model to ensure consistency of decision-support data across the corporation.

Data Modeling
A method used to define and analyze data requirements needed to support the business functions of an enterprise. These data requirements are recorded as a conceptual data model with associated data definitions. Data modeling defines the relationships between data elements and structures.

Dimension
A dimension is a structural attribute of a data mart that is a list of members, all of which are of a similar type in the user's perception of the data. For example, all months, quarters, years, etc., make up a time dimension.

ETL Short for Extract, Transform, Load. ETL refers to the process of getting data out of one data store (Extract), modifying it (Transform), and inserting it into a different data store (Load).

Fact Table
In a star schema, the central table which contains the individual facts being stored in the database. There are two types of fields stored in a fact table:

1. The fields storing the foreign keys which connect each particular fact to the appropriate value in each dimension.
2. The fields storing the individual facts (or measures) - such as number.

Foreign Key
A foreign key is the primary key of one data structure that is placed into a related data structure to represent a relationship among those structures. Foreign keys resolve relationships, and support navigation among data structures.

Granularity
The level of detail of the data stored in a data warehouse.

Measure
A numeric value stored in a fact table.

Metadata
Data that describes the data in the warehouse. This includes the database schemas, ETL's, and any other information that is needed to support and manage the operation of the data warehouse.

Normalization
The process of organizing data in accordance with the rules of a relational database. The central data warehouse is normalized while the data marts, with its emphasis on efficient retrieval of data, is de-normalized (see *star schema*).

OLAP On-Line Analytical Processing. Processing that supports the analysis of business trends and projections.

OLTP On-Line Transactional Processing. OLTP describes the requirements for a system that is used in an operational environment.

Primary Key
A column or combination of columns whose values uniquely identify a row or record in the table. The primary key(s) will have a unique value for each record or row in the table.

Star Schema
A star schema is a set of tables comprised of a single, central fact table surrounded by de-normalized dimensions. Each dimension is represented in a single table. Star schema implement dimensional data structures with de- normalized dimensions.

XML eXtensible Markup Language. A method of sharing data between disparate data systems, without needing a direct connection between them.

Printed in USA