



Installation Guide

QAD Business Intelligence

78-0956-3.6
BI Business Intelligence 3.6
March 2012

This document contains proprietary information that is protected by copyright and other intellectual property laws. No part of this document may be reproduced, translated, or modified without the prior written consent of QAD Inc. The information contained in this document is subject to change without notice.

QAD Inc. provides this material as is and makes no warranty of any kind, expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. QAD Inc. shall not be liable for errors contained herein or for incidental or consequential damages (including lost profits) in connection with the furnishing, performance, or use of this material whether based on warranty, contract, or other legal theory.

QAD and MFG/PRO are registered trademarks of QAD Inc. The QAD logo is a trademark of QAD Inc.

Designations used by other companies to distinguish their products are often claimed as trademarks. In this document, the product names appear in initial capital or all capital letters. Contact the appropriate companies for more information regarding trademarks and registration.

Copyright © 2012 by QAD Inc.

BIPortal_IG_v306.pdf/dbg/mat

QAD Inc.

100 Innovation Place
Santa Barbara, California 93108
Phone (805) 566-6000
<http://www.qad.com>

Contents

Change Summary	vii
Chapter 1 Installing QAD BI Analytical Modules	1
Pre-Install Information	2
Data Warehouse	2
Installation of QAD BI Analytical Modules	13
Requirements	13
Installation Steps	13
Chapter 2 Installing QAD BI Portal Web Application	23
Requirements	24
Configuration Parameters	24
Installation Notes	24
Installation Steps	24
Note on Password Encryption in the BI Portal	29
Getting Started in the BI Portal Web Application	30
Install Standard Dashboards and Reports	30
Installing QAD BI Mobile App on the iPad	32
Chapter 3 QAD BI Portal CFO Dashboard	33
QAD Standard CFO Dashboard	34
Configuring the CFO Dashboard	34
Metric Example	35
Reports Tab	35
ReportLines Tab	35
ReportLineRanges Tab	36
ReportLineCalculations Tab	36
Providing the Workbook Data to the Data Warehouse	37
Chapter 4 Reference	39
Jobs	40
Reference Tables	42

Chapter 5 Upgrading to QAD BI 3.6 from QAD BI 3.5 77

BI Metadata Upgrade Procedure	78
Preparation	78
Upgrade Steps for QAD BI Metadata	78
To Add a New Source	81
BI Portal Upgrade Procedure	82
Preparation	82
Upgrade Steps for QAD BI Portal	82
Verification	82
Update Portal Model	82
Update Portal Documentation and Standard Objects	83
Data Warehouse Tables Changed	83
Tables Modified	83
Tables Added	86
Tables Deleted	87

Appendix A Sizing Guidelines 89

Introduction	90
Important Restrictions and Caveats on BI Sizing	90
Business Intelligence Deployment Options and Terminology	91
BI Terminology	91
BI Deployment Options	92
Sizing Terminology Used Throughout this Sizing Guide	93
Sizing Methodology for QAD BI	93
CPU Sizing	94
CPU Sizing for DWD/Scheduler Component	94
CPU Sizing for BI Portal Webapp (Server Side)	94
CPU Sizing for Concurrent User Workload (Interactive Portal Users)	94
RAM Sizing	95
RAM Sizing for DWD/Scheduler Component	95
RAM Sizing for BI Portal Webapp (Server-Side)	95
RAM Sizing for Concurrent Users (Interactive Portal Users)	95
Disk and IO Subsystem	96
Note on Importance of Appropriate Disk and IO Subsystem Sizing ...	96
IO Subsystem Sizing	96
Key IO Subsystem Sizing Notes	97
Notes on Database Disk Sizing	97
Network Considerations	98
Network Recommendations	98
Example Sizing	99
Example 1	99
Reference: Estimation of Initial Database Size	100
Dependence of Initial Database Size on BI Modules	100

Estimation of Initial BI Data Warehouse Size	100
Allowance for Future Database Growth	102
Reference: Network Sizing Details	102
Terminology	102
Network Sizing Considerations - Database to BI Database Server ...	103
Network Sizing Considerations - Remote Client to BI Portal	104
WAN Considerations	104
Appendix B Customization Guidelines	105
Overview	106
Customization Challenges	106
QAD BI 3.x Customization Methodologies	109
Custom Path Methodology (Preferred)	109
Integrated Methodology (Last Resort)	111
Customization Needs for Additional Columns	111
Use Track Back Diagram to Check Tables	112
Double Checking the Data Path with a Column Track Back Report ..	114
Notes on Tables	116
Identifying Every Appearance of a Column in Existing Code	118
Naming Conventions for Custom Changes to Packaged QAD BI Modules ...	118
Custom Tables	118
Custom Columns Joined to Standard Module	119
Custom Columns Not Joined to the Baseline Module Tables	119
Guidelines for Creating Custom Path Custom Tables	119
Load Tables	119
Perm Tables and Pre-perm Stage Tables	119
All Other Stage, Extract	120
Adding Required xx_ Prefixed Columns	120
Adding Additional Columns to a Load Table	120
Adding Additional Custom Columns to Pre-perm Stage Tables	123
Adding Additional Custom Columns to perm/dim/fact Tables	124
Adding Custom Path stage/extract Tables After the perm Tables	127
Adding Additional Custom Loading and Staging Tables to Integrate in Additional Columns	130
Additional Customization Topics	131

Change Summary

The following table summarizes significant differences between this document and the previous version.

Date/Version	Description	Reference
March 2012/QAD BI 3.6	Provided instructions on upgrading from previous version	page 77
	Added sizing guide	page 89
	Added customization guide	page 105
September 2011 QAD BI 3.5	Rebranded for QAD BI 3.5	--

Installing QAD BI Analytical Modules

This chapter describes the initial setup of the Data Warehouse, and the installation and configuration steps for QAD Business Intelligence Suite (Order Management, Operations, and Financials Modules).

It discusses the following topics:

Data Warehouse 2

Describes the initial setup of the Data Warehouse.

Installation of QAD BI Analytical Modules 13

Covers the installation and configuration steps for QAD Business Intelligence Suite (Order Management, Operations, and Financials Modules).

Pre-Install Information

To save time during the installation, ask the customer in advance for information that you will need during the install. In Chapter 4 of the Install Guide, use Table 4.2 on page 72 to get system information and Table 4.3 on page 73 to get Parameter information from the customer.

There are two installation packages for Progress Open Edge for Windows: one is for 64-bit, one for 32-bit.

Currently, the 32-bit version must be installed, **not** the 64-bit version. Even if you have a 64-bit server, you have to install the 32-bit package to get the 32-bit ODBC drivers. QAD's understanding of the Progress Open Edge knowledge base is that you cannot install both on the same server (so you cannot install the 64-bit package on your 64-bit server but then install the 32-bit ODBC drivers; you have to just install the 32-bit Open Edge package).

Important The Progress Source database or databases must be running Progress version 10.1B03 or later.

Data Warehouse

A Data Warehouse must be initialized properly prior to the installation of any QAD BI Analytical Modules.

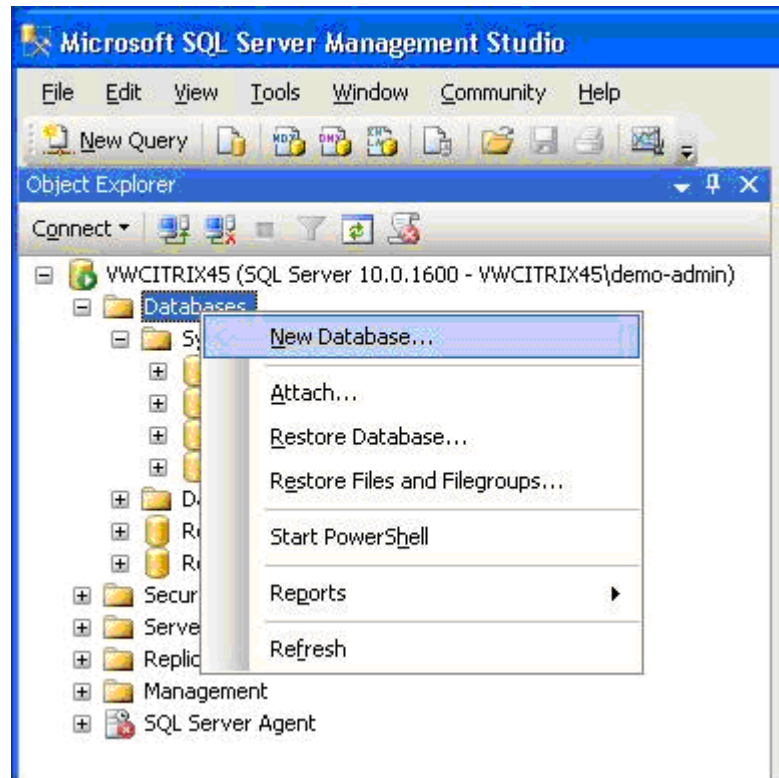
Follow the steps below to install and configure a new QAD BI Data Warehouse instance.

1 Create a database in SQL Server.

- a** From the Microsoft SQL Server Management Studio, right-click on Database, Create new database.

Fig. 1.1

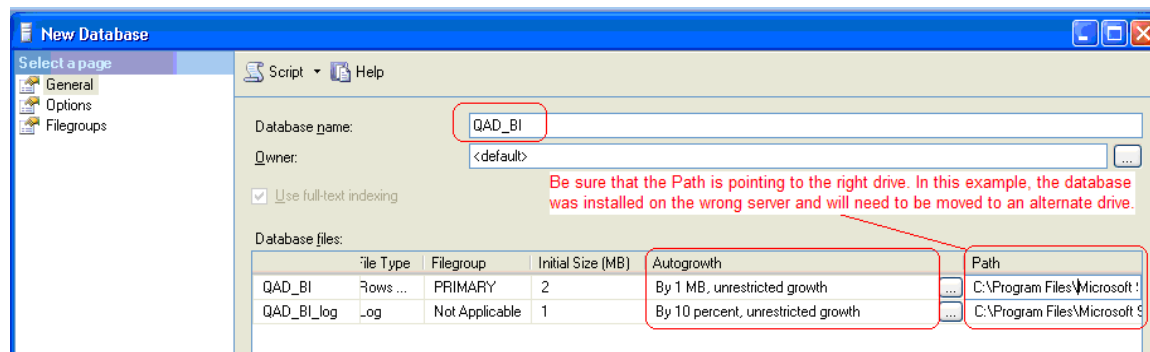
Create New Database 1



- b** Enter the new Database name. Make sure that the Path for the database is pointing to the correct server. Make a note for step c.

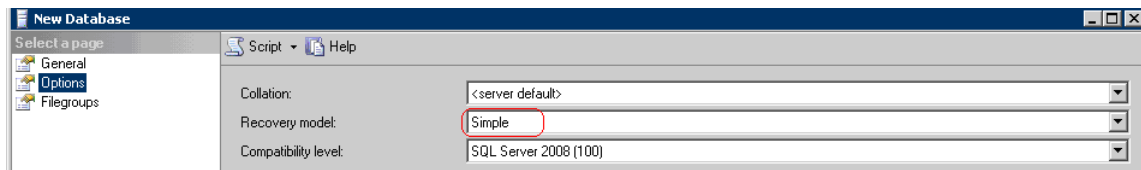
Fig. 1.2

Create New Database 2



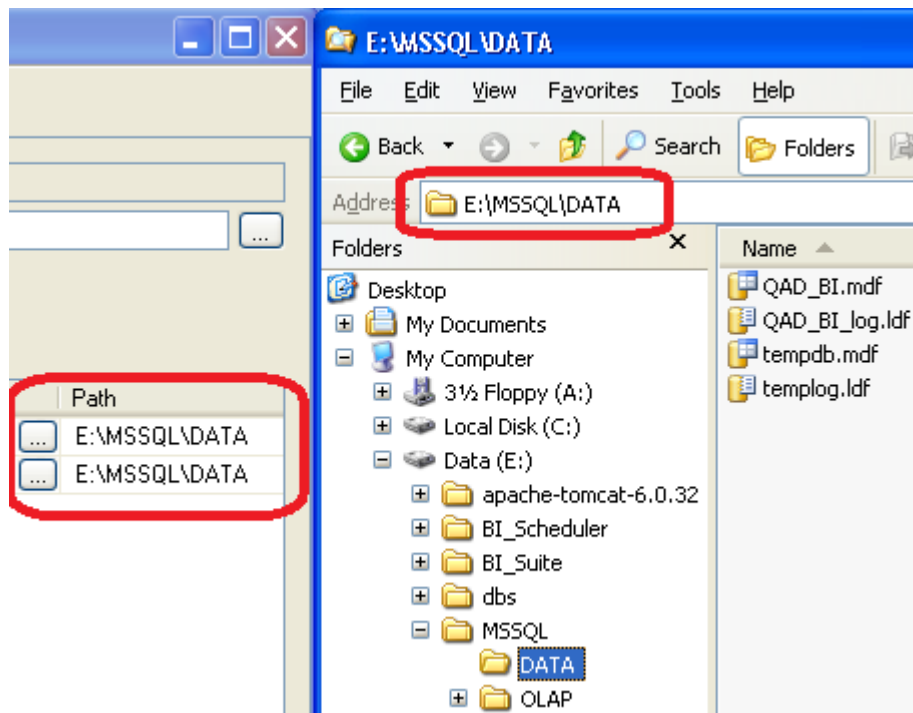
Important It is critical that you choose the Simple recovery mode from the Option tab.

Fig. 1.3
Create New Database3



- c If the Path from step b above is correct (pointing at the correct server), skip to step 2. If the path is incorrect, create a new folder on the correct server. Once created put that directory in for the Path for the database.

Fig. 1.4
Create New Database 4



- d Following up on Step c, if the Path was wrong for the database, the temp database needs to be checked to confirm that it is being accessed on the right server. Launch a SQL Server database query against your new database and run the following query:

```
USE tempdb
go
exec sp_helpfile
go
```

If the results point to the correct directory, go to step 2. Otherwise, set the tempdb folder using step e.

- e In order to set the tempdb to the correct directory, reset to appropriate directory (E drive in example below; set as appropriate):

```
USE master
GO
ALTER DATABASE TempDB MODIFY FILE
(NAME = tempdev, FILENAME = 'e:\mssql\data\tempdb.mdf')
GO
```

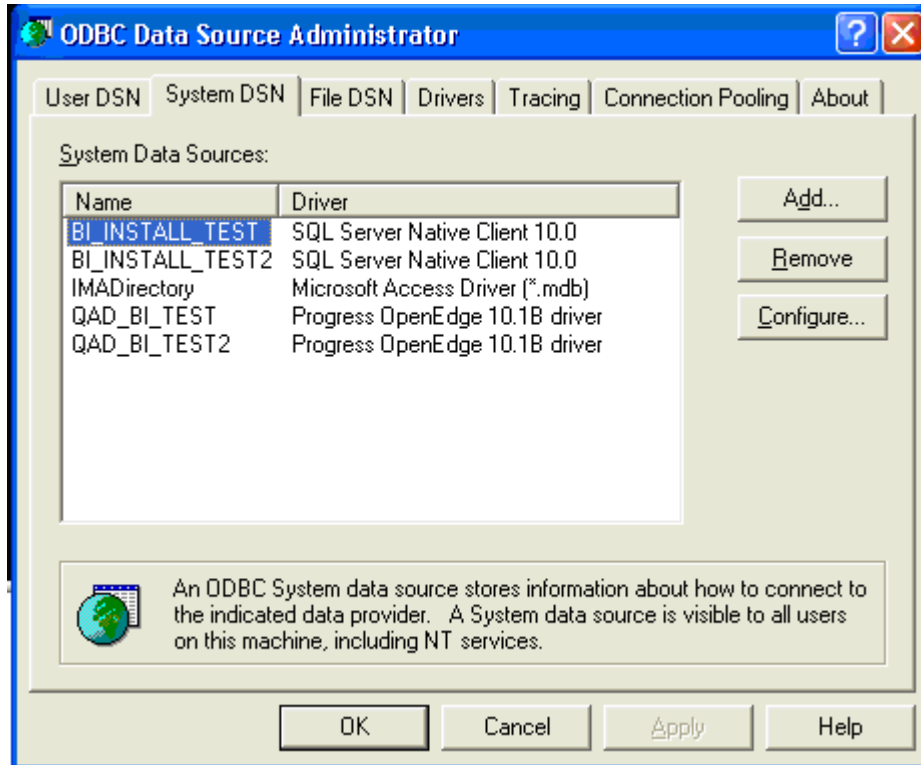
```
ALTER DATABASE TempDB MODIFY FILE
(NAME = templog, FILENAME = 'e:\mssql\data\templog.ldf')
GO
```

- f** For your the setting changes to be accepted, stop and start your SQL Server. In SQL Server, go to the server name (not database) in the top of the left hand pane, right click on it and pick Restart. When asked if you are sure you want to Restart your SQL Server session say yes. Confirm that your new temp directories now contain files indicating that the database is writing to the correct directory.
- 2** Install QAD BI DWD (6.5.4 or above) if you have not already done so. In Setup Administrator|PC Setup, pick the QAD BI Data Warehouse Designer option and enter the customer and license key.
- 3** Launch the ODBC Data Source Administrator from the QAD BI Administrator so that ODBC connections to the Data Warehouse and to the Progress databases can be created. Using the BI Administrator, in the PC Setup tab, pick the Validate/Add Odbc Source option. In the resulting ODBC Source screen on the right, right-click and select Add Odbc entry. This will launch the ODBC Data Source Administrator.

Note ODBC is used by the Data Warehouse Designer (QAD BI DWD) to communicate with the Data Warehouse database. It also allows the DWD to communicate with QAD's Progress databases and extract data from them to be used in the data warehouse. Persons who are familiar with the ODBC Data Source Administrator can launch it through other means, but there will be the possibility, in particular if on a 64-bit system, that the correct one is not being launched (the appropriate 32 bit ODBC driver, `odbcad32.exe`, must be used). By opening the ODBC Administrator via the QAD BI Administrator, any uncertainty of which is the correct ODBC Administrator is eliminated.

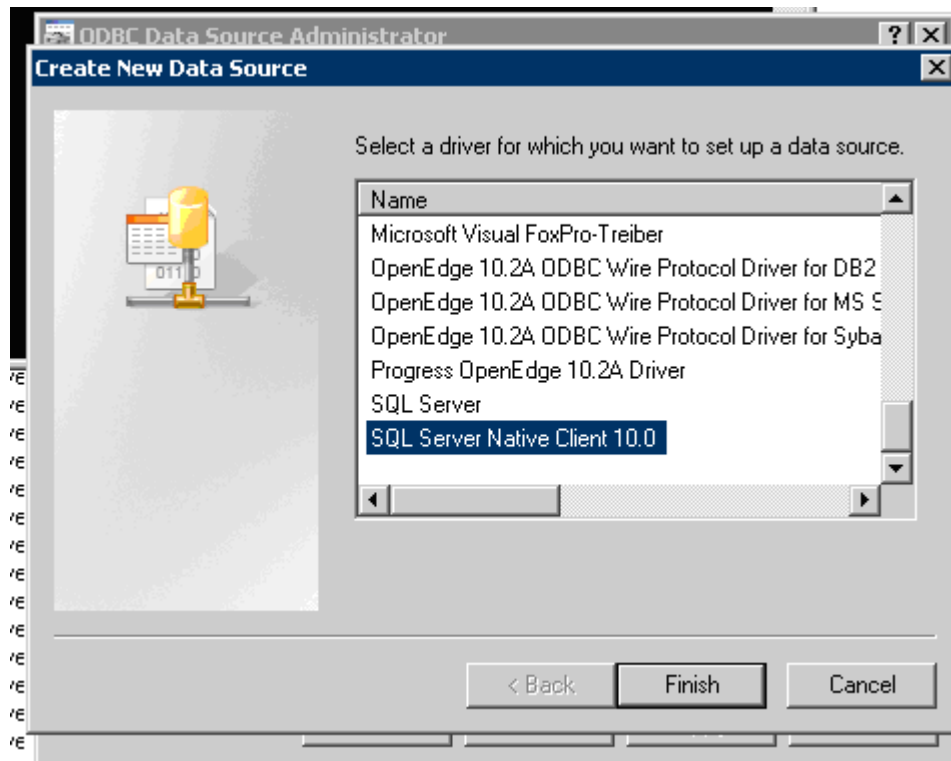
- 4 Create the ODBC connection to the SQL Server Database. This will be a System ODBC connection, using the SQL Server Native Client driver and Windows Authentication:
 - a Click on the System DSN tab and click the Add button.

Fig. 1.5
Create ODBC SQL1



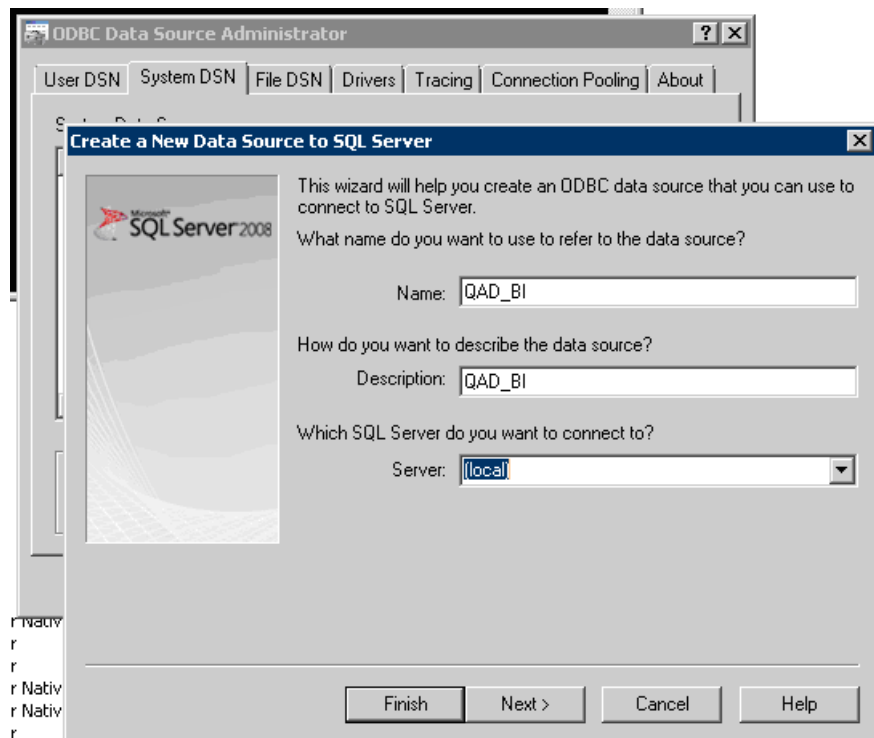
- b When Create New Data Source is displayed, select SQL Server Native Client 10.0 and click Finish.

Fig. 1.6
Create ODBC SQL 2



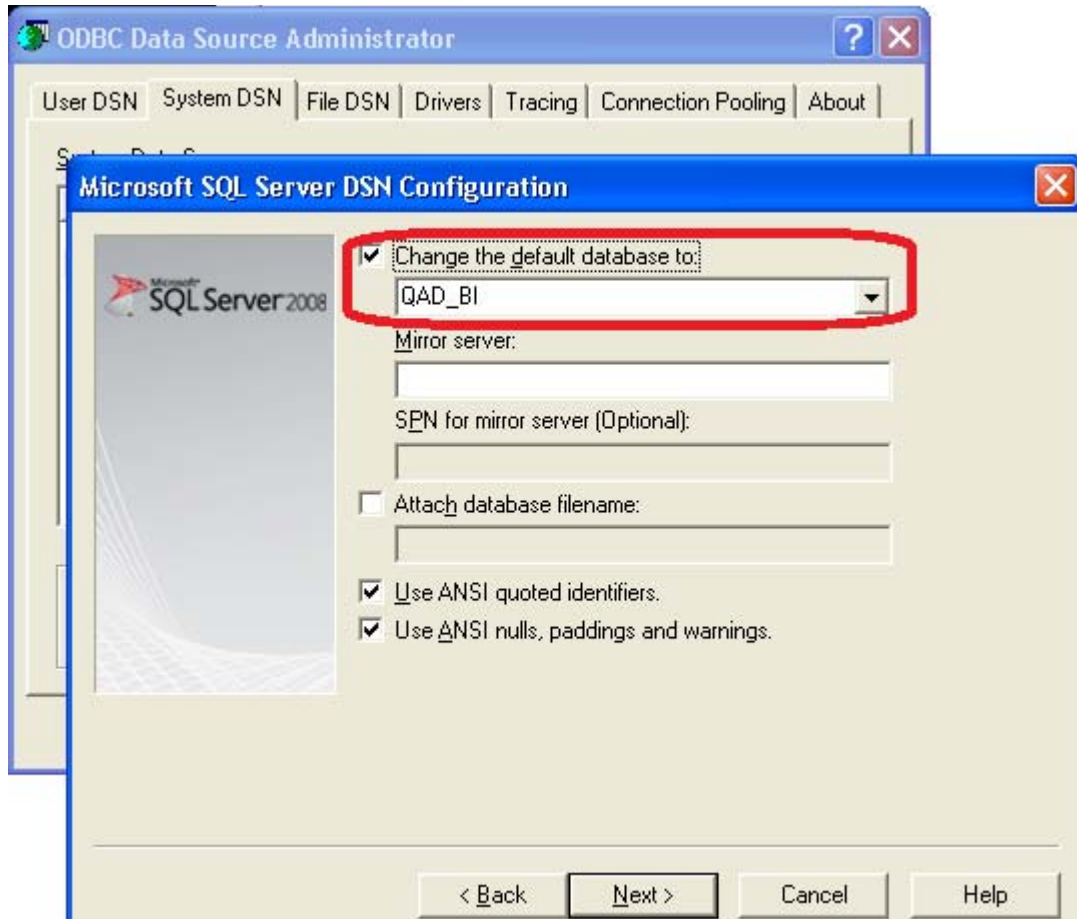
- c Enter the Name, Description and Server for the Data Warehouse and click Next.

Fig. 1.7
Create ODBC SQL 3



- d Choose the option With Integrated Windows authentication unless otherwise directed.

Fig. 1.8
Create ODBC SQL 4



- e Select the Change the default database to: box and change the database from master to the data warehouse database. Click Next, then click Finish.
- f Click the Test Data Source button in the bottom left.
- g Confirm that the database connection works properly. If successful, click OK.
- 5 If they do not already exist, create the ODBC connections to the Progress database that the data warehouse will be extracting from using the Progress ODBC driver. ODBC is used by the Data Warehouse Engine (QAD BI DWD) to access each Data Source. You will need the following information:
- If they have a preferred connection name
 - Host name
 - Port number
 - Database name
 - User ID
 - Password

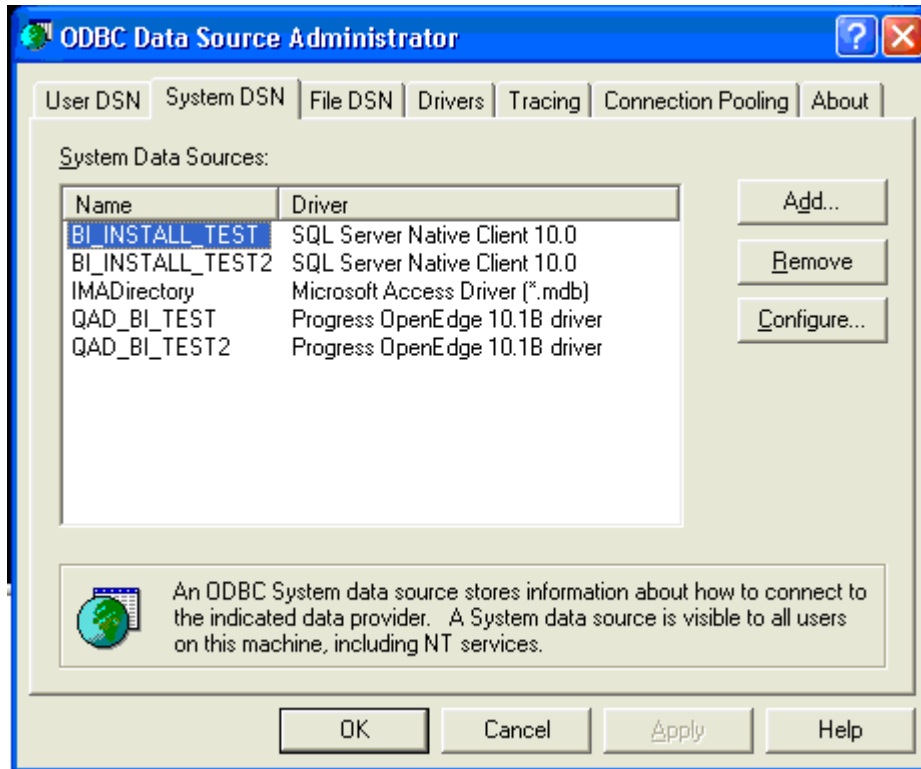
The port must be for a broker that accepts SQL connections. The user ID must have read permissions on the QAD tables.

If the Progress ODBC connects already exist, just go to steps d through f to test the connection and confirm that on the Advanced tab of the DSN configuration, the Default Isolation Level is set to READ UNCOMMITTED.

In the ODBC Administrator:

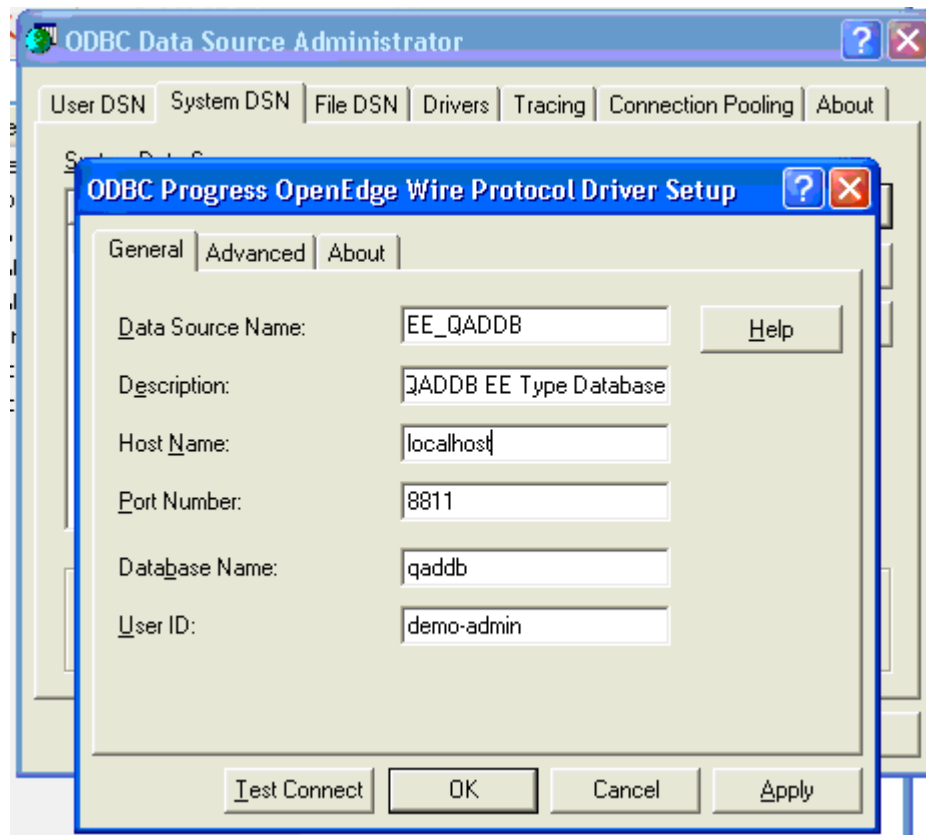
- a Click the System DSN tab and click the Add button.

Fig. 1.9
Create ODBC Progress 1



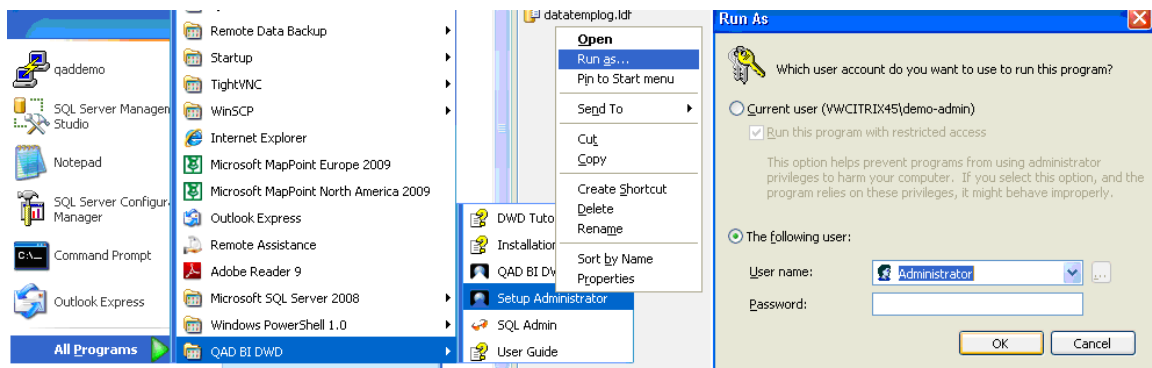
- b Pick the driver type Progress OpenEdge 10.1B driver. Click Finish.
- c Input the values provided to establish the connection to the Progress database.

Fig. 1.10
Create ODBC Progress 3



- d** Click Test Connect.
 - e** Enter Progress database password and click OK.
 - f** Click the Advanced tab and set the Default Isolation Level to READ UNCOMMITTED.
- 6** Set up any additional ODBC connections to outside data sources, if the client requires any. This can be done later if that information is not yet available.
 - 7** If you are using Windows 2008 Server, close the Setup Administrator. Re-launch the Setup Administrator as an Administrator (so that it will be able to create the Scheduler as a Windows Service).

Fig. 1.11
Setup Administrator



- 8 Choose Metadata Repository, and then Create Metadata Repository. Select the ODBC connection created in the previous step. Under Define FileGroups, leave them blank. The Repository Type should be All Licensed Objects.
- 9 When QAD BI DWD is launched after the repository is created, in the Objects tab of the options dialog, activate Cube objects by setting the position to a valid value (we are using 15). In older versions of the DWD, you had to populate the Data Store Position.

Fig. 1.12
Cube



- 10 Click the Ancillary tab. Populate the empty field `dss_create_time` with the value `dss_create_time`. Click OK to leave the Options page.

Fig. 1.13
Ancillary Tab

Options

Repository Objects Naming (Local) Naming (Global) Procedure Filegroup **Ancillary** Icons

`dss_batch`: Table, used to record batch numbers for data loaded into the data warehouse. Not used at this stage. See Help for further details.

`dss_batch`:

`dss_source_system`: Table, used to manage multiple source systems and non conforming data. Added by default to dimension business keys. Blank to disable. See Help for further details.

`dss_source_system`:

`dss_fact_table`: Table, used in the handling of rollup fact tables. Contains a record per fact table. Required if rollup fact tables are to use generated code. See Help for further details.

`dss_fact_table`:

`dss_create_time`: Column added to all dimension tables for information only. Leave blank to disable.

`dss_create_time`:

`dss_update_time`: Column added to all dim, stage and fact tables. Required if generated and aggregate tables are to be used.

`dss_update_time`:

`dss_start_date`: Column, used for slowly changing dimensions. Used to identify when data was replaced. Required.

`dss_start_date`:

- 11 In the Setup Administrator, create a Scheduler for the Data Warehouse. This is the Windows service that runs scheduled jobs. Use the ODBC connection that was created for the Data Warehouse. Set the Poll Interval to 15 from the default of 30 seconds. Any time less than 15 seconds raises the possibility of overloading the scheduler jobs if they do not gracefully close quickly after use. 30 seconds is acceptable, but due to the large number of scheduled jobs we have, reducing this is a big time saver.

Fig. 1.14
Poll Interval

Windows Scheduler Definition

Unique Identifier (e.g. SID_user):

Scheduler Name (e.g. W\Nnnnn):

Service Display Name:

Service logged on under User:

Work Directory:

Log File:

Poll Interval (default 30) seconds:

Log Level (0=none, 9=debug):

Direct Path Load or ODBC:

TNS Service (Oracle):

ODBC Connection:

Database User id:

Database Password:

OK Cancel

Installation of QAD BI Analytical Modules

This topic includes installation and configuration steps for QAD Business Intelligence Analytical Modules (Order Management, Operations, and Financials Modules).

Requirements

- QAD BI DWD (version 6.5.4 or later), Data Warehouse installed and Scheduler running.
- QAD Business Intelligence Metadata (Common, Order Management, Operations, and Financials Modules) installation files (in the form of QAD BI DWD application files).

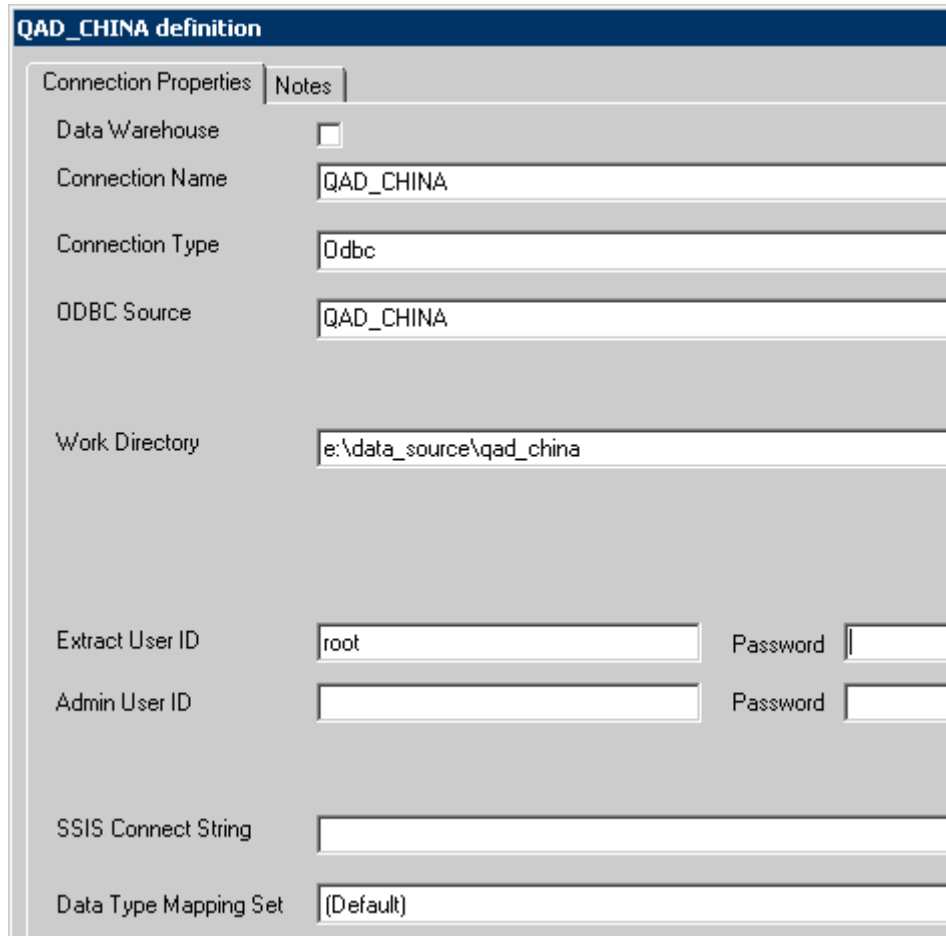
Installation Steps

Follow the steps below to install the QAD Business Intelligence metadata modules (Common, Order Management, Operations, and Financials Modules).

The previous installation task created the System ODBC connections to the QAD Progress databases. Now, for each one of these ODBC connections, in the QAD BI DWD in the Browser pane, create a Connection object for each:

- 1 Right-click Connection in the QAD BI DWD Metadata Browser and choose New Object.
- 2 Enter a short descriptive name for the data source (for example, QAD_MAIN, QAD_US, QAD_EMEA, QAD_ASIA) and click OK.
- 3 Select a Connection Type of ODBC, and select the ODBC connection.
- 4 Enter a work directory. This directory should be different for each QAD Data Source. Create this directory in the file system (you will create a matching directory in step 6).
- 5 Enter the Extract user ID and password required for the Progress database. This user must have read permission for the QAD tables.
- 6 Using Windows Explorer, create a directory to match the directory you entered as the Work Directory for the connection you just set up.
- 7 Repeat all steps for each data source.

Fig. 1.15
Object Connection Definition

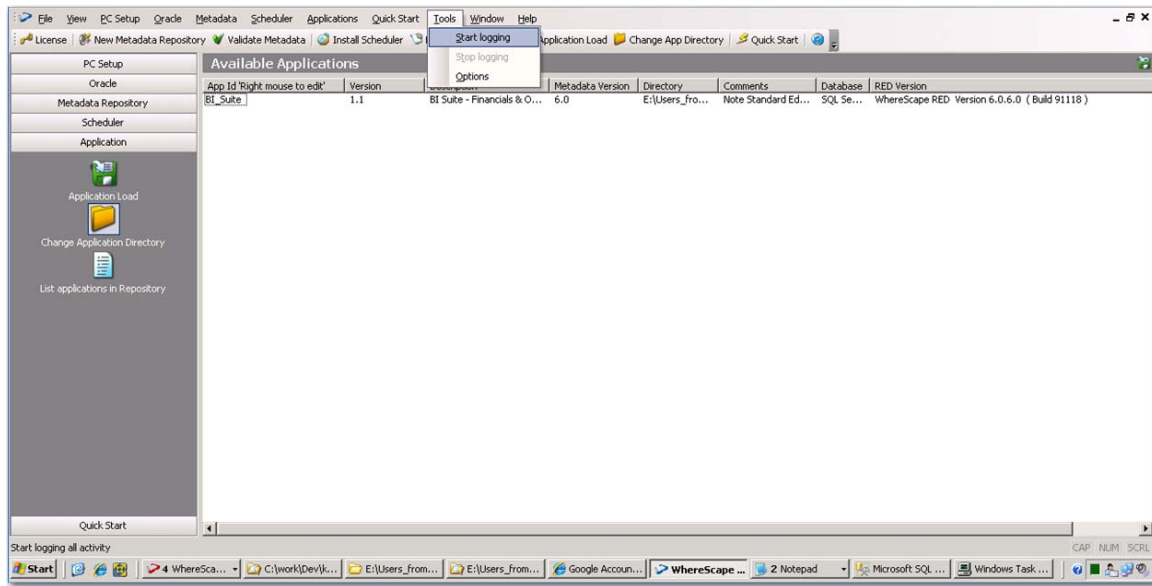


QAD_CHINA definition	
Connection Properties Notes	
Data Warehouse	☐
Connection Name	QAD_CHINA
Connection Type	Odbc
ODBC Source	QAD_CHINA
Work Directory	e:\data_source\qad_china
Extract User ID	root
Admin User ID	
Extract Password	
Admin Password	
SSIS Connect String	
Data Type Mapping Set	(Default)

Install BI Common Module

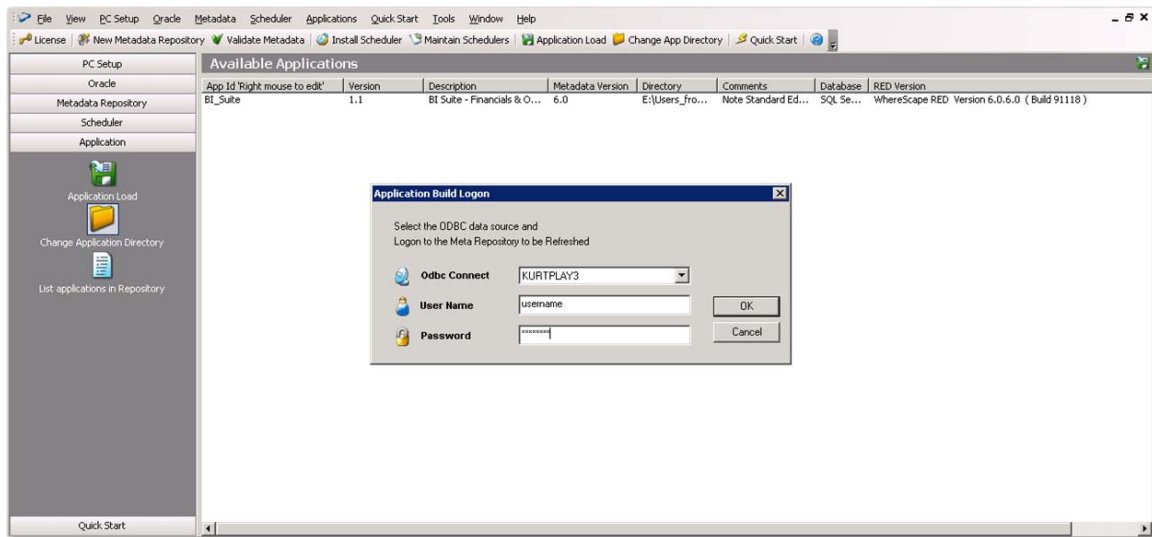
- 1 The QAD Business Intelligence Analytical Modules (Common, Order Management, Operations, and Financials Modules) are packaged as zip files. Unzip the contents (metadata files) for the Common module—plus any analytical modules you want to install—to a directory that you will reference for the installation. (The Common module must be installed before installing any of the analytical modules; it contains core components used by all modules.) It is not recommended that you just unzip the file to the default zip name subfolder. Instead, unzip the contents to whatever directory you had intended the contents for.
- 2 In Setup Administrator, confirm that the job scheduler is started (the service was created when the data warehouse was created). If it is not started, start the service and confirm that it does not stop.
- 3 Go to the Application option in Setup Administrator. Change Application Directory to the directory into which you extracted the QAD Business Intelligence Common Module metadata. The contents will consist of a series of .wst files.
- 4 Go to Tools and Start Logging.

Fig. 1.16
Start Logging



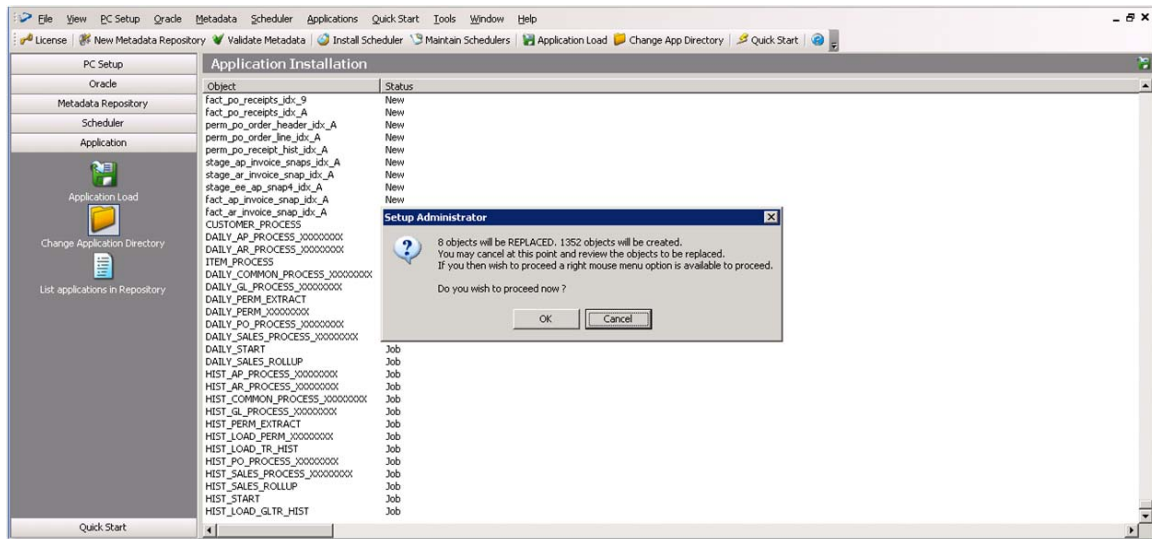
- 5 Choose a file name for your log and a directory for it to write to.
- 6 Install the application through Application Load. Right-click on the Common module and choose Install.

Fig. 1.17
Application Load



- 7 If a message comes up about files being overwritten, choose Cancel and confirm that the files are OK to be overwritten. If they are, go back to step 6 and repeat.

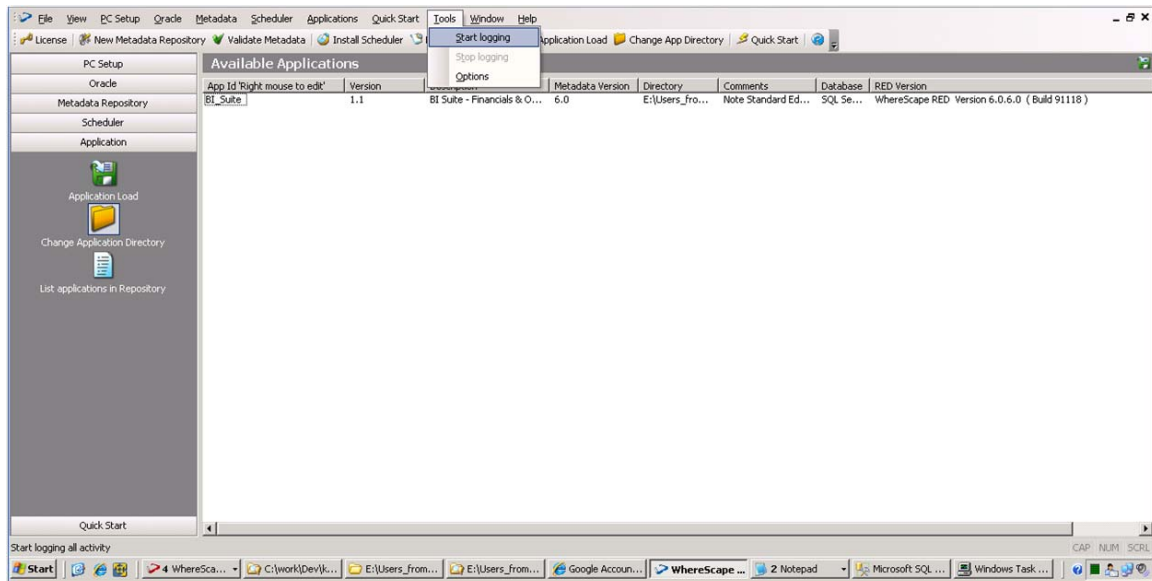
Fig. 1.18
Confirm File Overwrite



Install BI Analytical Models

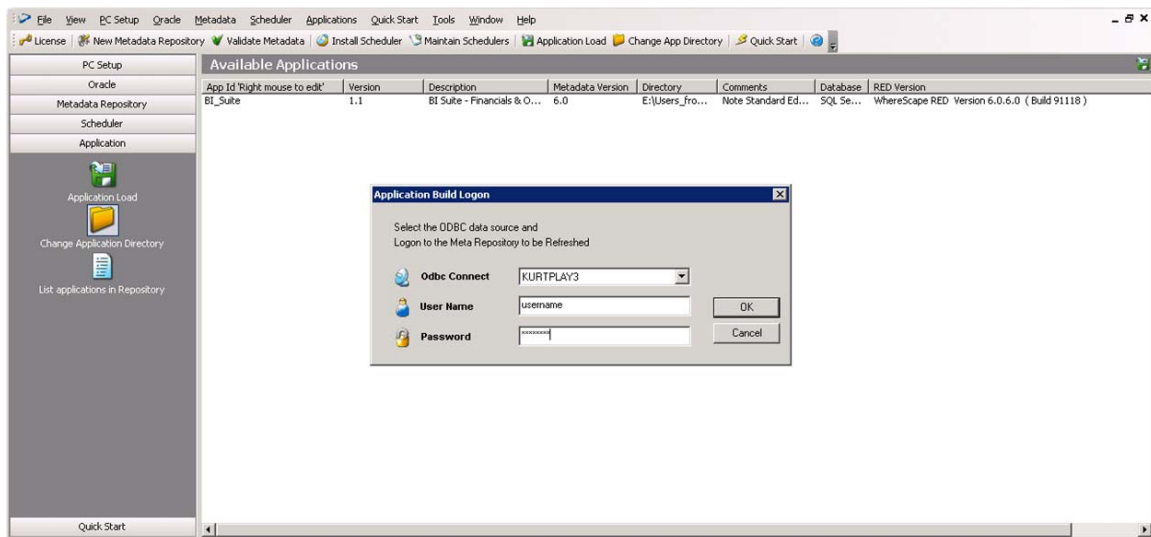
- 8 Next, go back to the Application option in Setup Administrator. Change the Application Directory to the directory into which you extracted the first Analytical Module you want to install (Financials, which contains both SE and EE financial tables; Operations; or Order Management).
- 9 Go to Tools and Start Logging.

Fig. 1.19
Start Logging



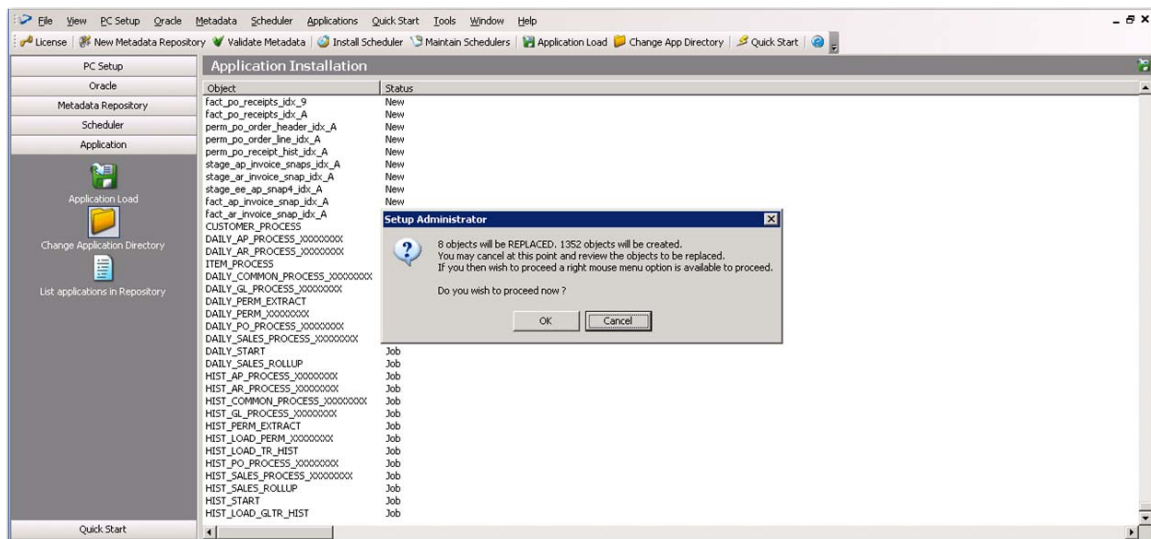
- 10 Choose a file name for your log and a directory for it to write to.
- 11 Install the application through Application Load. Right-click on the desired module and choose Install.

Fig. 1.20
Application Load



- 12** If a message comes up about files being overwritten, choose Cancel and confirm that the files are OK to be overwritten. If they are, go back to step 11 and repeat.

Fig. 1.21
Confirm File Overwrite



- 13** Repeat for any other modules that are required.
- 14** Review the message log. The files that will be replaced will be identified at the top of the log. To proceed, right-click within the log and choose Proceed to Load. Once the main application load is running, this may take as long as 5-10 minutes. If the top of the DWD screen displays Not Responding, wait up to 30 minutes for the job to finish. If it takes any longer, close the screen and start over.
- 15** After loading all the module components, you can refresh the DWD by clicking on All Objects in the Browser pane and pressing F5. If this does not work, you can log off and log back into the DWD.

16 Initial Parameter Setup

Note In QAD BI DWD, choose Parameters from the Tools menu to display the current Parameter definitions. These need to be customized for your specific installation.

Set runtime parameters to the proper settings based on the customer feedback to the parameters (Table 4.3 on page 73). Also refer to “Jobs” on page 40 and Table 4.1 on page 42 for additional assistance. In particular, change the HIST_LOAD_SOURCE_Sxx, DAILY_LOAD_SOURCE_Sxx and INITIAL_JOB_SETUP_CONNECTION_xx parameters to point at the source databases.

Note Parameters that MUST be set before running INITIAL_JOB_SETUP should have an initial value of <some value between these symbols >. See Chapter 4, “Reference,” on page 39 for more details.

17 Initial Job Setup

Note The INITIAL_JOB_SETUP job will create the proper chained job flow for the all of the source connections.

- Set the INITIAL_JOB_SETUP_CONNECTION_01 parameter to your first QAD ERP source.
- You have to also populate the INITIAL_JOB_SETUP_CONNECTION_01_TYPE parameter with the type of source system, EE or SE, in order for the chained job flow to be set up with correct jobs for the source type.
- Finally, set INITIAL_JOB_SETUP_CONNECTION_01_RUN to N (this indicates you have not run a historic load for this source yet).
- If you have more than one source system, set INITIAL_JOB_SETUP_CONNECTION_02 to your second QAD ERP connection and INITIAL_JOB_SETUP_CONNECTION_02_TYPE.
- You can continue adding as many INITIAL_JOB_SETUP_CONNECTION_XX and INITIAL_JOB_SETUP_CONNECTION_XX_TYPE parameters as necessary.
- Eliminate the values for the connections that are not being used.
- Set the INITIAL_JOB_SETUP_ENABLED parameter to Y, and in the Scheduler, execute the INITIAL_JOB_SETUP job.

Setup Steps:

- a In QAD BI DWD, chose Scheduler menu option.
 - b Change the view to All Jobs.
 - c Locate the job named INITIAL_JOB_SETUP (status On Hold).
 - d Right-click the job name and start the job.
 - e The status will change to Waiting.
 - f Refresh the screen until the job completes.
 - g If the job fails to start, check the scheduler status.
- 18** After INITIAL_JOB_SETUP has run, check Scheduled jobs to confirm that you now have SET_CONNECTION jobs with names that end with all connections you have created. If they have not been created, check the Parameters; make sure that the INITIAL_JOB_SETUP_ENABLED parameter is set to Y and run INITIAL_JOB_SETUP again.

Notes on Load History

TR_HIST_LOAD_MAXSIZE is the maximum number of rows to load in an iteration loading tr_hist.

The Transaction History load (HIST_LOAD_TR_HIST) processes data in chunks, rather than trying to extract and load all data at once (since the tables can contain large amounts of data). The History Load processing will run this job many times, processing some subset of the data with each pass. The parameters that are used by the TR_HIST jobs are listed below.

Important Do not update them manually; the HIST_LOAD jobs use these to keep track of what data has already been processed, so that if the load job is interrupted, it can pick up where it left off.

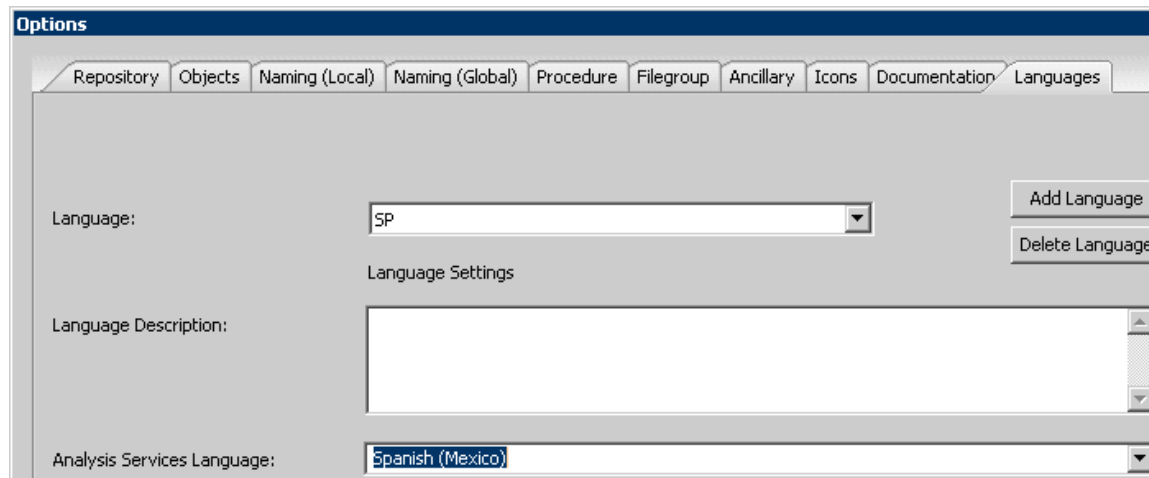
- TR_HIST_CONNECTION_NAME
- TR_HIST_DOMAIN
- TR_HIST_DOMAIN_SXX
- TR_HIST_LOOP_NO
- TR_HIST_ROW_COUNT
- TR_HIST_SOURCE_SXX
- TR_HIST_TRAN_MAX
- TR_HIST_TRAN_MAX_SXX
- TR_HIST_TRAN_MIN

19 Install the metadata language translations files (if any are required).

- a** In QAD BI DWD (Data Warehouse Designer) tool, choose Tools, Options. Click on the Languages tab and add the codes for the language files you want to install. (For example, es for Spanish, fr for French, zh for simplified Chinese, zh_TW for traditional Chinese. The complete set of files will be in the metadata package, under metadata/translations; the file names are in the format lang_, then the code, then. wst; for example, lang_zh_TW.wst for traditional Chinese.) The interface can be a little misleading here. It requires that you click the Add Language button, then type in a language or abbreviation in the Language drop down menu. What you type will help you find language options in the Analysis Services Languages list, but the initial display in the Analysis Services Language list will not necessarily be the language you want. What you type for the Language is simply used for reference purposes in later steps. For example, if you type

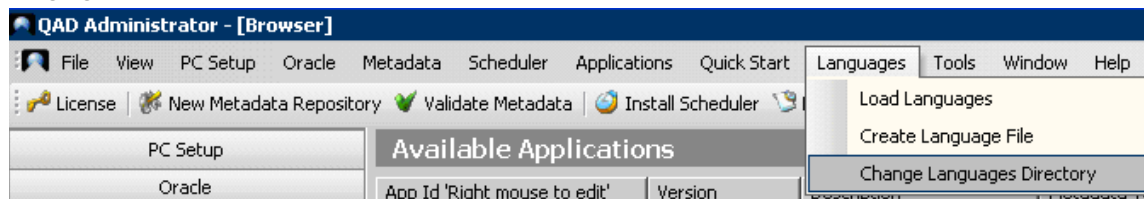
“es”, the ISO abbreviation for Spanish, the Analysis Services Language drop-down will display Estonian (since that starts with “es”). You can just click on the drop-down and scroll down to the Spanish option you want to select at that time.

Fig. 1.22
Language Install



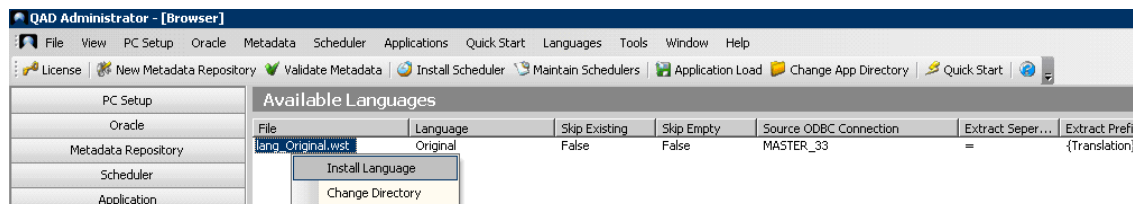
- b** After adding all the language codes you need in the DWD, launch the QAD BI Data Warehouse Setup Administrator, select Languages, Change Languages Directory, and navigate to the directory metadata/translations directory of your package.

Fig. 1.23
Language Install 2



- c** Choose Languages, Load Languages. Right-click on the language file you want to install and choose Install Language. Select the Data Warehouse’s ODBC DSN, and then choose the language to which you want to map this language file (you will be allowed to choose one of the language codes you entered in the QAD BI DWD in step 1 above).

Fig. 1.24
Language Install 3



- d** Repeat for each language you want to install.

- 20 You are now ready to populate the Data Warehouse with your ERP history. In the scheduler, find the HIST_START job, right-click on it and pick Start the Job. Depending on the size of the source tables this can take quite some time. It does not hurt to check in throughout this load process to make sure that the jobs are all running successfully.

Installing QAD BI Portal Web Application

This chapter provides information about the installation and configuration steps for the QAD BI Portal Web application.

It discusses the following topics:

Requirements 24

Lists requirements for installation.

Configuration Parameters 24

States configuration parameters.

Installation Steps 24

Lists the installation steps for the BI Portal Web Application.

Requirements

- Tomcat 5.5, 6, or 7 installed and configured
- Java 1.5 JRE or higher
- Java 6 JDK or higher
- SQL Server 2008 installed and configured (SQL Server 2005 is acceptable, but not recommended due to poor performance)
- SQL Server has TCP enabled, in order to connect via JDBC
- QAD BI Data Warehouse Designer installed and configured
- Data Warehouse created in SQL Server. In the QAD BI DWD: QAD BI Analytical Module(s)
- QAD BI DWD, customizations made, data loaded, Scheduler created, desired language translations installed

Note End users will need Adobe Flash Player installed for their browser.

Configuration Parameters

The following configuration parameters will be required during installation, so write them down before you start:

- Tomcat Port: Port in which Tomcat is running; for example, 80 or 8080
- SQL Server JDBC URL: URL required to connect to SQL Server database (host name, port number, and database name)
- QAD BI Data Warehouse Designer user / pwd: Username and Password to connect to SQL Server Database where QAD BI DWD is running. This should be the QAD BI DWD owner account.
- BI Portal database User / pwd: Username and Password to connect to SQL Server database where BI Portal repository will be installed.

Installation Notes

Note that the QAD BI Portal cannot be installed in the same Tomcat instance as Qxtend. Qxtend is forcing the use of a specific DOM parser library that is incompatible with the BI application usage. Install the two applications in separate Tomcat instances.

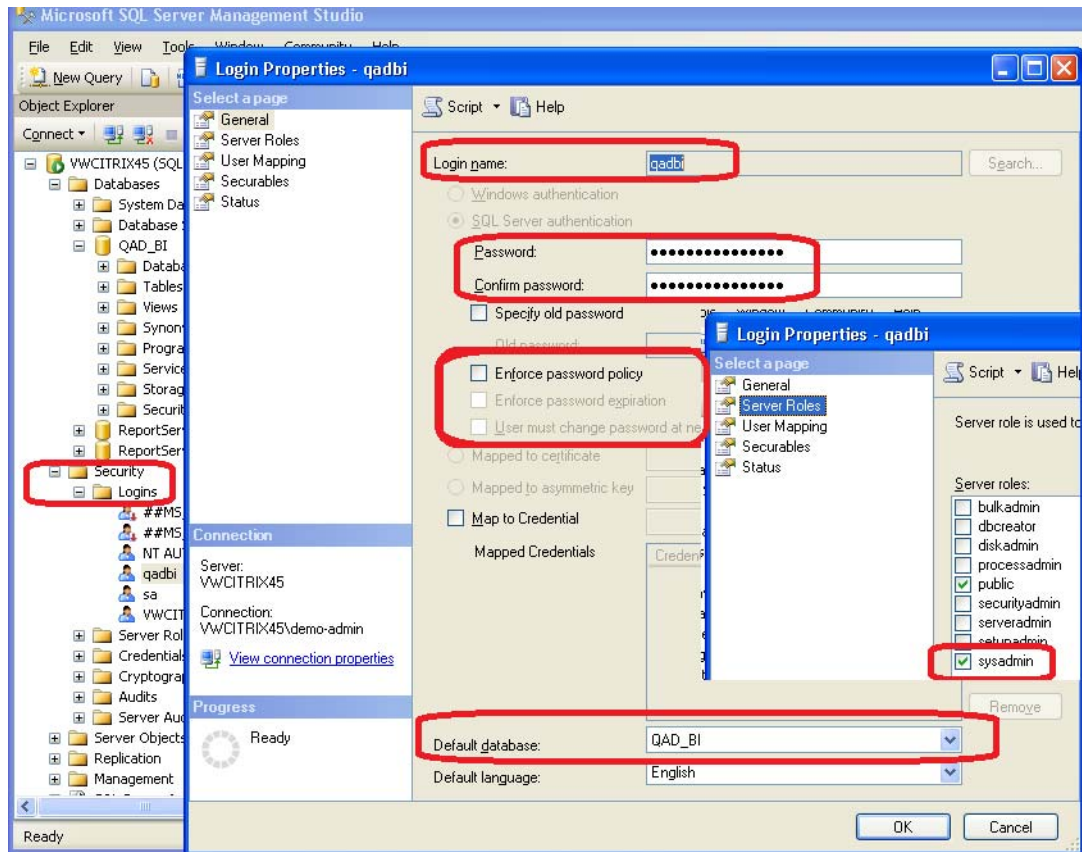
Installation Steps

Follow the steps below to install and configure a new BI Portal instance.

- 1 As part of the Portal Installation, set up a new user in the database. Go to the SQL Server Management Studio. Choose Security, Logins, right-click Logins, and select New Login. When the Login - New screen comes up, enter the new login name (for example, qadbi), pick SQL Server authentication, and make sure that the Enforce password policy box is not

selected. Change the database from master to the database you are working with. Then select the Server Roles (on the upper left-hand side of the window) and make sure that the Server roles public and sysadmin are both selected.

Fig. 2.1
Login Properties



2 Install BI Portal Web Application:

- a Create a subdirectory for the web application beneath the `<tomcat_dir>\webapps\` directory; for example, qadbi.
- b With Tomcat not running, unzip `qadbi.zip` into the Tomcat webapps sub-directory created above. To shut down Tomcat, in Windows, go to Start/Administrative Tools/Services and find Apache Tomcat in the list of running services. Right click on it and choose Stop.
- c Configure Tomcat's `server.xml` file (can be found in the `/conf` directory of Tomcat) and add the `maxPostSize` directive (it is case sensitive), setting it to 0:

```
<Connector port= 8888
protocol= HTTP/1.1
connectionTimeout= 20000
redirectPort= 8443
maxPostSize= 0 />
```

3 Configure BI Portal.

- a** Copy the XML file `<tomcat_dir>\webapps\qadbi\WEB-INF\config\server-config.xml.default` to a file named `server-config.xml` in the same directory. Edit the `server-config.xml` file to match the values for the configuration parameters to the appropriate values for your environment, as described below. Contents are as follows:

- repository Element:
 - dbType: database platform. Value should be SQL for SQL Server.
 - driver: JDBC driver to be loaded at runtime for business intelligence server database. For SQL Server, the value is `com.microsoft.sqlserver.jdbc.SQLServerDriver`.
 - jdbcUrl: JDBC URL to connect to Business Intelligence Portal database
 - user: User name for the database
 - pwd: Password for the database
 - logCheckInterval: How often (in minutes) the portal should perform its maintenance tasks (log the system state, clean up logs, and so on). Default is 15 minutes.
 - logDaysToKeepEvents: Number of days to store system events in the database (`aw_evt_system` and `aw_evt_request` tables). Default is 14 days.
- log Element:
 - watch: Whether to watch the log file for changes and update log4j settings dynamically; true or false
 - configFile: The name of the log4j configuration file
- Example contents of the `server-config.xml` file:

```
<root>
<repository
dbType=SQL driver=com.microsoft.sqlserver.jdbc.SQLServerDriver jdbcUrl=
jdbc:sqlserver://myhost:1433;database=QAD_PORTAL user=myuser password=mypwd
logCheckInterval=15
logDaysToKeepEvents=14 />
<log watch=true configFile=LoggerConfig.properties />
</root>
```

4 Configure log4j in LoggerConfig file

- a** Copy the file `<tomcat_dir>\webapps\qadbi\WEB-INF\config\LoggerConfig.properties.default` to a file named `LoggerConfig.properties` in the same directory. Edit the Log4j configuration file to specify a path for the log file that is the R1 appender defined in the properties file. Note that if you are running Tomcat as a Windows Service, the log file name must be a full path, and the backslashes must be escaped; for example,

```
log4j.appender.R1.File=D:\\apache-tomcat-6.0.20\\logs\\qadbi.log
```

5 Install BI Portal Repository:

- a Connect to the SQL Server database where BI Portal Repository is to reside, using the SQL tool of choice. Locate the BI DDL script in the `<tomcat_dir>\webapps\qadbi\install\sqlserver` file. Run the `bi3_ddl.sql` SQL script to create required repository tables.

Note This script will drop the tables first if they already exist, and re-create them. Make sure you are running this in the correct database.

- b Edit the seed script `seed.sql` to set any variables to the specific values necessary for your installation.
 - Set the e-mail address for the initial root user to the e-mail address of the application administrator. If required, customize the authorization roles (the values being entered into `aw_auth_role` table, and into `aw_user_auth_role` for the initial root user). Do not modify any of the `aw_app_role` descriptions. The `aw_data_role` table, as well as the `data_role` column in `aw_user`, are no longer used.
 - Set the different `aw_sys_config` entries as follows:

- **DATASERVICE:** Configure with the information required to connect to the DWD data warehouse. This includes the `dbType` (SQL), `whSchema` (the name of the database owner—typically `dbo` for SQL Server), the `jdbcDriver` and `jdbcURL`, `jdbcUser`, and `jdbcPassword` for accessing the database, and the `metadataAdapter` (`com.awbi.server.data.adapter.WhereScapeAdapter`). For example:

```
<dataConfig dbType= SQL whSchema= dbo
jdbcDriver=
com.microsoft.sqlserver.jdbc.SQLServerDriver
jdbcURL= jdbc:sqlserver://myhost:1433;database=QAD_DW jdbcUser= myuser
jdbcPassword= mypwd
metadataAdapter= com.awbi.server.data.adapter.WhereScapeAdapter />
```

- **WELCOME_EMAIL:** Customize the text that will be contained in the Welcome email sent to new users. Make sure that the link to the BI Portal web application is correct for your environment.
 - **SMTP and MAILER:** Configure this with your specific mail server parameters.
 - **USERSERVICE:** Customize the text that will be used in the Subject of the Welcome and Password Change emails as desired.
 - **PWD_RESET_EMAIL:** Customize the text that will be contained in the Password Reset emails sent to users who request a new password.
 - **SECSERVICE:** Do not edit; security models will be built in the Administration UI.
 - **AUTHSERVICE:** If a custom authentication class will be used, set the custom `authClass` to the customized authorization class and uncomment this section. Otherwise, leave this section commented out.
- c Run the modified `seed.sql` script against your database to load the data into your portal database.

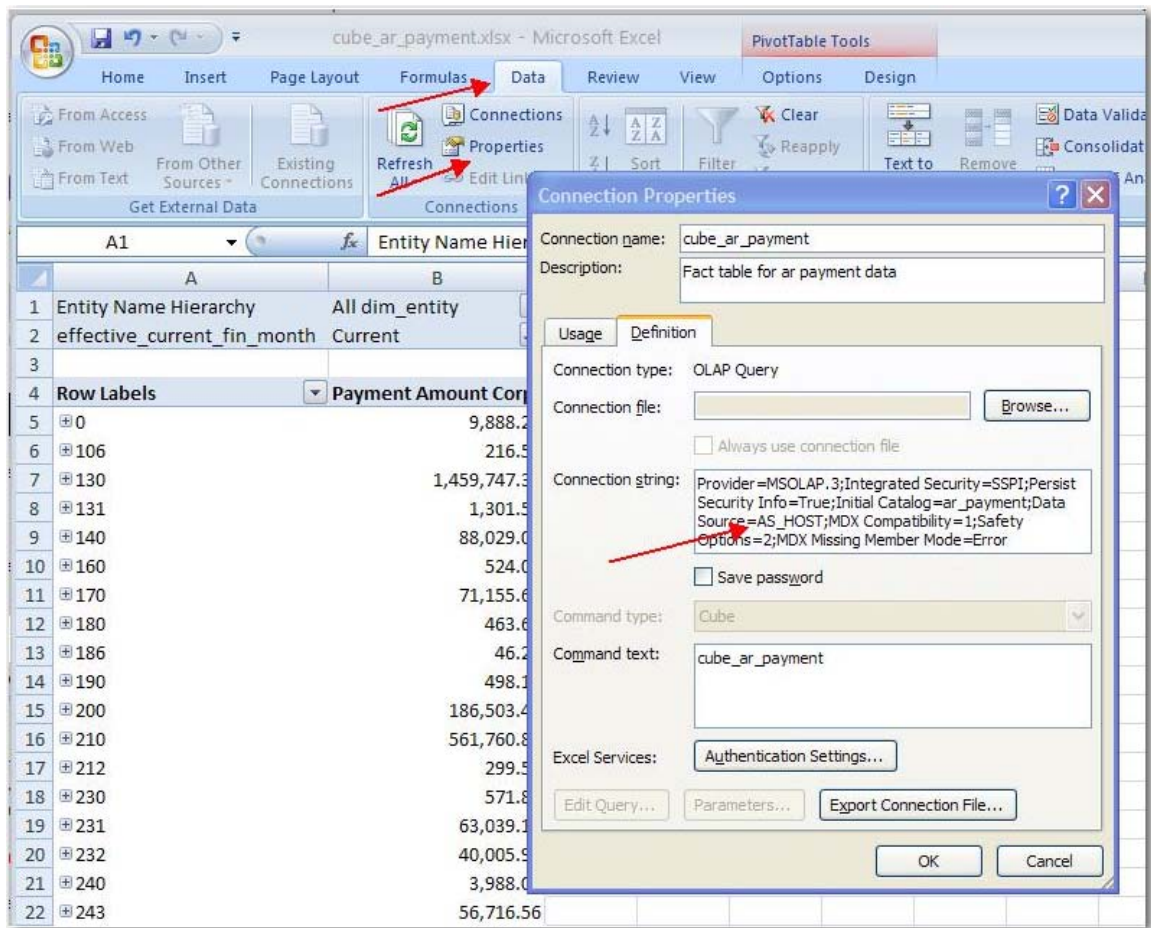
6 Install Portal constraint tables in data warehouse.

Connect to the database in which the data warehouse is to reside. This is the database specified in the DATASERVICE key, above. Locate the `bi3_wh_ddl.sql` script in the `<tomcat_dir>\webapps\qadbi\install\sqlserver` directory. Run the `bi3_wh_ddl.sql` SQL script to create the required portal tables in the data warehouse database.

- 7 The Standard Dashboards include links to the technical documentation for the Data Warehouse. This documentation is shipped with the Common analytical modules in the file `qadbi-documentation.zip`. The Standard Dashboards expect them to be located in a directory named `documentation` within the web application. To deploy the documentation, therefore, create a directory `<tomcat_home>/webapps/<application_name>/documentation` and extract the `qadbi-documentation.zip` file into that directory.
- 8 Installing Excel Workbooks:
 - a Some of the standard dashboards for some analytical models include Excel workbooks that connect to an Analysis Services system for ad-hoc analysis. Create a new web application directory, `<tomcat_home>/webapps/<application_name>/published`. Extract the file `<module_name>_adhoc_excel_files.zip` into that directory. These files are referenced by the standard dashboards loaded in the step “Install Standard Dashboards and Reports” on page 30.
 - b Each workbook uses a data source connection to the Analysis Services system. The connections used are embedded in the worksheets but must be modified to connect to the local Analysis Services host. To do this, open the workbook in Excel 2007, press Data, then Properties. On the Connection Properties tab select Definition and change the text that says `DataSource=AS_HOST` to give the name of your Analysis Services host machine.
 - c Full documentation on this process is included in the following document from Microsoft:

<http://office.microsoft.com/en-us/excel/HP101672991033.aspx>

Fig. 2.2
Excel Workbooks



- 9 Customize Report Logo. The report wizard allows you to choose to put a logo on your generated reports. The file it uses is stored in two places:
- `<tomcat_dir>\webapps\qadbi\images\logo.gif` - used for standard HTML reports.
 - `<tomcat_dir>\webapps\qadbi\WEB-INF\classes\images\logo.gif` - used for PDF and Excel exports of reports.

If you want to have your company logo print on reports, replace the default `logo.gif` in the web application's images subdirectories with a GIF file of your logo. The logo file can be different for the two classes of report types.

- 10 Restart Apache Tomcat through the Services window.

Note on Password Encryption in the BI Portal

With this release of BI Portal, you may now encrypt the SQL Server password in both the `server-config.xml` file (which contains the connection information for the BI Portal database) and in the `aw_sys_config` table (which contains the connection information for the BI Data Warehouse). The encryption uses the Blowfish algorithm.

A command-line tool to encrypt the password named `encpwd.bat` is located in `TOMCAT_HOME\webapps\your_bi_appname\cmdline`. To run this, open a DOS window, change to the `cmdline` directory, and run:

```
encpwd.bat mypassword
```

The encrypted password will be printed on the screen. Copy this value and use it in the `server-config.xml` file as the value for the `pwd`. Also make sure that `pwdEncrypted=true`. (If you change back to using an unencrypted password value, make sure you set `pwdEncrypted=false`.)

Similarly, if you want to encrypt the password used by the Portal to connect to the Data Warehouse, use the `encpwd.bat` script to generate the encrypted password, then edit the value for the `jdbcPassword` in the `aw_sys_config` table, the `DATASERVICE` row, and set it to then encrypted value; make sure you add `pwdEncrypted=true`.

If you change either password, you must stop and restart Tomcat for the change to take effect.

Getting Started in the BI Portal Web Application

The initial login is root, with a password of root. You will be required to change the password immediately.

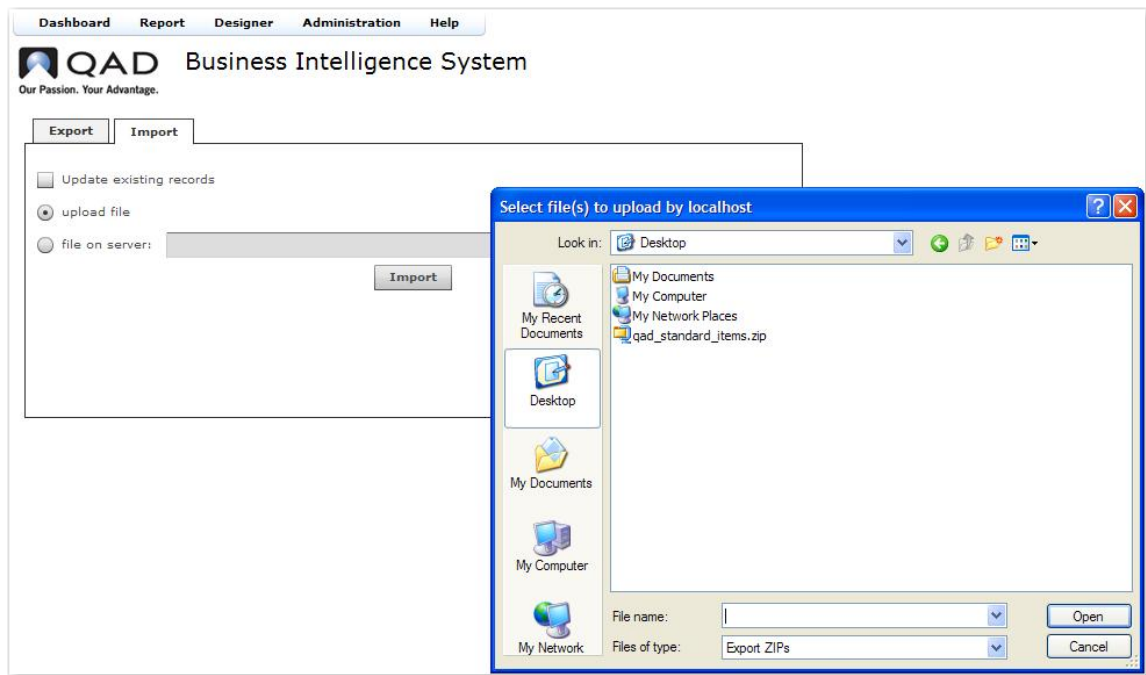
You will need to build your model before you can do any analysis. Click Administration, then Model Administration. Click on the Rebuild Master button.

Install Standard Dashboards and Reports

Standard dashboard, reports, visual items, and queries are shipped with the QAD analytical modules. The contents of the zip files `qad_<module_name>_dashboards.zip` and `qad_<module_name>_reports.zip` are portal documents used to interact with the standard modules.

Find the `qad_common_dashboards.zip` file and place it somewhere for easy access. Using the portal, go to the menu option Administration|Data Migration. Click the Import tab, click the upload file option, and press the Import button. Select the `qad_common_dashboards.zip` file and select OK.

Fig. 2.3
QAD Standard Items Zip



The items will upload into the system. On completion, a message indicating that the items were loaded will be displayed.

Repeat the import process for your module-specific `qad_<module_name>_dashboards.zip` and `qad_<module_name>reports.zip` files.

After importing the standard objects, you must log out and log in again to see them.

Installing QAD BI Mobile App on the iPad

Requirements

- Apple iPad
- QAD BI Portal installed, configured and Tomcat running
- User IDs created and passwords set up using the BI portal for all users who will install and use the app on their iPads

Installation and Configuration

- 1 From the Apple App Store, install the QAD BI app on your iPad.
- 2 Connect to your office network via VPN.
- 3 Double-click on the QAD BI icon. It will prompt you for some settings:
 - user: enter the user id that you use to access BI portal.
 - password: enter the password that you use to access BI portal.
 - Server URL: the same URL that you use to access BI portal in the form of `http://<biportalhostname>:<tomcatportnumber>/<webappname>/`. For example:
`http://company:8080/qadbi/`.

If you would like to save these credentials for future use, tap on the Save Password On/Off button and turn it on.

- 4 Click on the check mark. You should now be able to log in to BI from your iPad.

QAD BI Portal CFO Dashboard

This chapter provides information about configuring the QAD BI Portal CFO Dashboard application.

It discusses the following topics:

QAD Standard CFO Dashboard 34

Describes how to configure the CFO Dashboard.

QAD Standard CFO Dashboard

The QAD CFO Dashboard is one of the standard elements that ships with the QAD Business Intelligence Portal. The CFO Dashboard is designed to collect in one easy-to-view place the important financial metrics for your organization so that the financial health of the organization can be quickly understood.

Since GL account structures vary widely from company to company, the data elements of the CFO dashboard require initial configuration. The Dashboard's visual elements are already set up in the Portal, and are ready to display graphs and charts once the data configuration is complete.

Configuring the CFO Dashboard

Metrics on the CFO dashboard are generated from report lines defined in the QAD BI database. Report lines are like the lines of a balance sheet or an income statement, representing a range (or ranges) of GL accounts. Once defined, report lines can be rolled up into groups and computations performed on the groups to generate the various metrics and data that appear on the CFO dashboard.

The CFO Dashboard report lines are defined and maintained with the help of an Excel workbook, which creates a set of text files that are used in the nightly QAD BI data load. Using Excel allows finance managers and analysts to maintain the financial metrics themselves and gives them the most freedom in creating and defining new metrics and financial reports.

The CFO Dashboard workbook, `cfo_dashboard_template.xls`, is located in the metadata installation package. This workbook consists of four sheets, all of which are used to build the configuration data for the CFO dashboard and its charts.

The first tab, Reports, contains the very highest level of information to be displayed. This page lists the name of the report sets that the CFO dashboard describes. One set is given by default: CFO_METRICS. Multiple reports, or versions of reports, can be included in the same spreadsheet.

The second tab, Report Lines, defines the lines of each report. Report lines can be labels (which just display text), data lines, which are rollups of GL accounts, or calculations, which are rollups of data lines. The CFO_METRICS report in the default configuration is broken down on the Report Lines page into 16 different lines, such as Inventory and Net Revenue.

The third tab, Report Line Ranges, maps the Report Lines to specific accounts in the General Ledger.

Important metrics for each report may not simply be summaries of accounts but more complicated combinations of the report lines. The fourth tab, Report Calculations, allows user to combine the lines of the report arbitrarily to create derived metrics.

These tabs are more fully described below. Before configuring the CFO dashboard, however, you should start with a specific goal in mind to properly set up the configuration elements.

Metric Example

Consider the example of an organization that wants to see a metric like debtor days: the average number of days that a payment takes. This can be computed on any account by taking the total amount in accounts receivable, divided by net revenue received, and then multiplied by the number of days in the fiscal year. Furthermore, this should be done by region so we can see the measures for the North American and Pacific Rim regions.

To do this, we'll need to first set up the accounts that we want to total. We'll need to specify the AR accounts in both North America and the Pacific Rim, and then set up the revenue received in those accounts.

Once these account lines are set up we can create the computation to display on our final dashboard.

Reports Tab

For our example, the reports tab will not have to change. This report can be collected in the set for the standard CFO_METRICS.

If the system calls for very many reports that we want to divide at the highest level, we could add another element to the reports tab for that highest level grouping. That new element would then be carried over to the Report Lines tab.

ReportLines Tab

This tab holds the high level groupings on which the computations in the reports will be performed. In our specific example, we have four lines we need to define: Accounts Receivable for both North America and the Pacific Rim, and revenue received for both regions.

These accounts are created by setting the report_name column to the name given on the first tab. In our example, this is CFO_METRICS.

The line_number column must be unique for each entry on this page and it describes to the subsequent tabs how to reference this account line.

Line_type may have the value label, calc or data. A label type will be the text placed into the visual item on the CFO dashboard for this line. A row can be marked either calc for a computed column as described in the upcoming Calculations tab, or data to indicate that this line's data will be used in a computation.

Line_text is a description of the account. Indent_level should be set to 0 for a label, and 1 otherwise. Items that should be visible to the end user should be marked as Y in the visible column.

All computations on a report line are done on the value of the report. Since the report computations described below are additive, and sometimes we must subtract values of one line from another, we can specify that a line should be a negative value in the report lines tab by placing a Y in the change_sign column.

In our specific example, the line of net revenues must be subtracted from the total accounts receivable for a region. When we specify the received revenue in the report lines tab, we will change its sign to a negative by indicating a Y in the change_sign column for both the North American and Pacific Rim Net Received lines.

ReportLineRanges Tab

The report line ranges tab allows the user to group the accounts required for each report line.

Accounts in the GL are listed on the report line ranges tab sequentially to generate the higher level reporting accounts. The report line name (CFO_METRICS) must be given in the first column. The line_number column corresponds to the report line in the second tab. Range_type can take the value of include or exclude.

The remaining columns specify the characteristics range of accounts to include in the summarized report line. The description of each column is given here:

- Source_system_code: name of the source system in the QAD BI database. Most implementations have a single source system.
- From_entity / to_entity: entity codes to be processed, starting at from value and going until to value.
- From_account / to_account: account codes to process.
- From_subaccount / to_subaccount: sub accounts to process.
- From_department / to_department: departments to process.
- From_project / to_project: project indicators to process.

In our specific example, the accounts for the four elements of our computation are defined. Report Line 2, North American AR (East) is defined as 4000 to 4100, excluding accounts 4003 to 4005. Report Line 3, North American AR (West) is defined as 5010 to 5020. The source system for both is QAD_US. Line 4, Pacific Rim AR, is defined as accounts 1600 to 1700 for source system for both is QAD_PACRIM. The example tab continues to define the account ranges and source systems for all 5 data lines of the CFO_METRIC report.

ReportLineCalculations Tab

This tab defines the computations to be performed on the report lines as defined on the Report Lines tab. By combining the report lines together, the higher order metrics desired can be determined.

Each line in the ReportLineCalculations tab specifies part of a computation. The data into which the computation will be placed is specified in the line_number of type calc specified on the ReportLines tab. The computation steps are processed one at a time, in an order specified by each line's calc_seq. For each specific report_name and line_number, the calc_seq must be unique.

```
A step
Report_name - CFO_METRIC
Line_number - refers to the ReportLines tab, line_number. Must be a calc type.
Calc_seq - order in which computations should be performed
Operator - add or multiply
Source_line - refers to a data line in ReportLines
```

Providing the Workbook Data to the Data Warehouse

Once the data has been entered into the workbook as described, it must be translated into a format that the data warehouse processes can operate on to build the required elements for the CFO dashboard.

Each sheet in the workbook contains a yellow section of cells that is built automatically as the data is filled out. This section of cells must be placed into a text file and that text file moved to a location specified in the data warehouse.

The location in which the text files must be placed can be found by double-clicking on the load tables in QAD BI DWD and noting the directory location in the File Path. By default, this is set to `c:\work\QADReport\`. This can be changed to match the locations required by any site.

Fig. 3.1
File Path

The screenshot shows a dialog box titled "load_gen_led_report_line table definition" with a tabbed interface. The "File Attributes" tab is selected. The fields are as follows:

- File Path:** C:\work\QADReport\
- File Name:** ReportLines.txt
- File Column Delimiter:** | (A dropdown menu shows "CHAR(nn) inserts an ascii character (e.g. CHAR(9) = tab). No column delimiter indicates a width based record.")
- File has a header line:** ☒
- File Record Delimiter:** (A dropdown menu shows "FIX nnn indicates a fixed length record of length nnn. No record delimiter indicates a line feed delimited record.")
- BULK INSERT options:** (An empty text field)

On the right side of the dialog, there is a link that says "See online help for further details. (e.g. TABLOCK, MAXERRORS=0)" and a help icon.

The files to be used, by default, are

- Reports tab - Reports.txt
- Report Lines tab - ReportLines.txt
- Report Line Ranges tab - ReportLineRanges.txt
- Report Line Calculations tab - ReportLineCalculations.txt

Once the files are in place and the QAD Warehouse daily jobs run, the new computations will be reflected into the data warehouse and the CFO dashboard contents displayed. The daily jobs are usually run overnight.

Reference

This chapter provides reference information:

Jobs ***40***

Reference Tables ***42***

Jobs

Jobs can be roughly categorized into the following types, in the order in which the types will run in Chained job processing. The descriptions include the naming conventions associated with each type.

Truncate

The purpose of a Truncate job is to truncate—delete all data from—load tables prior to the beginning of a new load. This ensures that load jobs are starting with a “clean slate.” There is one such job per analytical module, named `PREPROCESS_[MODULE]_TRUNCATE`.

Load

The purpose of a Load job is to extract data from the ERP source(s) and load the data into the Data Warehouse for processing. The Load job for an analytical module is named `PREPROCESS_[MODULE]_LOAD`. However, if there are specific tables that are relevant to history loads only, or to daily loads only, those will be loaded in a job named `PREPROCESS_[MODULE]_HISTONLY_LOADS` or `PREPROCESS_[MODULE]_DAILYONLY_LOADS`, respectively. Also, there might be further distinction when a module requires separate job streams; for example, there are some tables processed for EE sources only, and some for SE sources only. In this case the EE or SE distinction will be included in the job name; for example, `PREPROCESS_FIN_EE_LOADS` is a load job for the Financials analytical module that will run for EE sources only. COMMON module load tables with SE- or EE-specific tables will include SEONLY or EEONLY in their job names; for example, `PREPROCESS_COMMON_EEONLY_LOADS` or `PREPROCESS_COMMON_SEONLY_LOADS`.

Perm

These jobs create the “perm” tables—permanent data stores, where the BI system stores data extracted from the ERP source(s) that needs to be maintained for long periods of time. There are two such jobs for each module—one for History loads and one for Daily loads. Typically, the History job expects a large amount of data to be processed at once as the Data Warehouse is loaded for the first time; Daily loads are usually processing a smaller amount of data, that which has changed in the source(s) since the last data load. These jobs are named `[HIST_or_DAILY]_[MODULE]_PERM_CHAINED`. For Financials, there may be an `_EE` or `_SE` after the `[MODULE]` if the job applies to one type of ERP source only.

Extract

This set of jobs is to build extract and work tables from the perm tables. Not all of the data in the perm tables is typically extracted at one time, as much of the data in the perm tables has been already processed and has not changed. So usually a subset of data, covering a certain period and/or specifically targeting changed data, is extracted from the permanent data stores for further processing into the Fact tables. There are two such jobs for each module—one for History loads and one for Daily loads. These jobs are named `[HIST_or_DAILY]_[MODULE]_PERM_EXTRACT`. For Financials, there may be an `_EE` or `_SE`

after the [MODULE] if the job applies to one type of ERP source only. Note that Extract jobs do not end in CHAINED; they are not an explicit part of the job chain, but rather are always called from a Perm job.

Process

This set of jobs is to take extracted and loaded data through various stages to build the dimensions and facts (no summaries, aggregates, snapshots, or cubes). These jobs are named [HIST_or_DAILY]_[MODULE]_PROCESS_CHAINED. For Financials, there may be an _EE or _SE after the [MODULE] if the job applies to one type of ERP source only. Also, some modules may have a main job containing the module name, and then additional jobs for processing related to a specific submodule. These jobs are named

[HIST_or_DAILY]_[MODULE]_[SUB_MODULE_ABBREVIATION]_PROCESS_CHAINED and will run after the primary job for the module. For example, the Financials module has its primary Daily job, DAILY_FIN_PROCESS_CHAINED, plus separate jobs for processing Accounts Payable (AP), Accounts Receivable (AR), and General Ledger (GL)—so the submodule jobs are DAILY_FIN_[EE_or_SE]_AP_PROCESS_CHAINED, DAILY_FIN_[EE_or_SE]_AR_PROCESS_CHAINED, and DAILY_FIN_[EE_or_SE]_GL_PROCESS_CHAINED.

Rollup (aggregate/rollup/snapshot)

This set of jobs is to create aggregate/rollup/snapshot tables, which mainly rely on the relevant facts/dimensions to be populated already. There are two such jobs for each submodule—one for History and one for Daily loads. They are typically named POSTPROCESS_[MODULE]_SUMMARY. However, if the specific post-processing is to history only, or to daily only, the job name will be POSTPROCESS_[MODULE]_HISTONLY_SUMMARY or POSTPROCESS_[MODULE]_DAILYONLY_SUMMARY, respectively. Additionally, if the post-processing is specific to a submodule of a module, the submodule's abbreviation will appear in the name; for example, POSTPROCESS_OP_PO_DAILYONLY_SUMMARY.

Cubes

Cubes jobs process analytical cubes so that they include the most recent data from the fact and dimension tables. There is one job per module. The naming convention is POSTPROCESS_[MODULE]_CUBES. For Financials, there might be an SE or EE after the MODULE name if the cubes are specific to one source type; for example, POSTPROCESS_FIN_SE_CUBES.

Reference Tables

The following tables include installation and configuration reference information in tabular form.

Table 4.1
Parameters (Page 1 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
ALTERNATE_EXCHANGE_DOMAIN		USA	Alternate exchange rate domain name	Will be used to calculate exchange rates if the domain in which the calculation is occurring does not have an exchange rate defined	COM	M
AP_CHECK_PROCESS_DAYS	60	60	Used to find any checks to be processed which covers up to 60 days backdated.		FIN	M
AP_EE_HISTORY_DATE		20081231	The date the AP aging table is to be refreshed from. Three years back is common. For example, 20081231	Set from AP_EE_HISTORY_DATE_MIN when initial job setup is run.	FIN	P
AP_EE_HISTORY_DATE_MIN	<customer defined - usually three years back. YYYYMMDD>	20081231	The date the AP aging table is to be refreshed from. Three years back is common. For example, 20081231	Added this in 3.4.1 to handle historic loads from multiple EE sources.	FIN	M
AP_EE_HISTORY_DATE_Sxx		20081231	Inserted via WsParameterWrite	Created by procedures. - 3.4.1, using the AP_EE_HISTORY_DATE. Sxx references which EE source number this date is for.	FIN	P

Table 4.1
Parameters (Page 2 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
AP_EE_SNAPSHOT_FREQUENCY	M	M	The frequency of snapshots in the AP aging snapshot tables. Possible values are: D for Daily, W for Weekly, M for Monthly, B for both weekly and monthly.	Note that Daily will result in 7 times the amount of data as Weekly; Weekly will be approximately 4 times the amount of data as Monthly. Both Weekly and Monthly means that you will get a record for each end-of-week date as well as a record for each end-of-month date. Need to determine frequency based upon desired view level (i.e., do you want to be able to compare things on a monthly, weekly, or daily basis), number of days of data to view (i.e., how far back should the data go in the fact table), and amount of physical disk space available	FIN	M
AP_EE_SNAPSHOT_HIST_START_DATE	2008-12-31	2008-12-31	The first date in the AP aging snapshot table in the format YYYY-MM-DD for the first run	Date used for start of AP snapshot in History load. Used for History load only; Daily loads use AP_EE_SNAPSHOT_NUM_DAYS	FIN	M
AP_EE_SNAPSHOT_NUM_DAYS	31	31	The number of days of snapshots to load back each run.	Number of days to include in snapshot. Used to determine AP_EE_SNAPSHOT_START_DATE, which will be calculated as TODAY - this value.	FIN	M
AP_EE_SNAPSHOT_REPLACE_FLAG	N	N	Indicates snapshots will be replaced or not. Y for replace, N for not replaced.	Y to replace the snapshot for recalculated periods when the snapshot job is run; N to just add new records and not update previously calculated records. Default value is N.	FIN	M
AP_EE_SNAPSHOT_START_DATE	1996-11-12	1997-05-24	The first date in the AP aging snapshot table in the format YYYY-MM-DD	Calculated as TODAY minus AP_EE_SNAPSHOT_START_DATE	FIN	P

Table 4.1
Parameters (Page 3 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
AP_PROCESS_CONNECTION_NAME		2010SE	Indicates the connection name used by the AP process job	Connection name used by the currently running AP process, or if no AP job is running, the last-used source connection	FIN	P
AP_PROCESS_DAYS	10	10	The number of days of AP to extract and process from permanent staging tables.		FIN	M
AP_PROCESS_RUNNING	N	N	Indicates if an AP process is currently running (Y,N)	Used to prevent multiple copies of the AP job from running at the same time. Under normal conditions you should not update this manually. However, if a job fails and is deleted, you may need to reset this to "N" to run the job again. But use caution and ensure that there really is no copy of the job running before changing from "N" from "Y".	FIN	P
AP_PROCESS_RUNNING_JOB_NAME		DAILY_EE_AP_CHAINED	Indicates the current or last run job name for AP processing.		FIN	P
AP_REVALUE_PERIODS	30	30	The number of periods to revalue in invoice history		FIN	M
AP_SE_SNAPSHOT_FREQUENCY	M	M	The frequency of snapshots in the AP aging snapshot tables. Possible values are: D for Daily, W for Weekly, M for Monthly, B for both weekly and monthly.		FIN	M
AP_SE_SNAPSHOT_HIST_START_DATE	2009-01-01	2009-01-01	The first date in the AP aging snapshot table in the format YYYY-MM-DD for the first run	Date used for start of AP snapshot in History load. Used for History load only; Daily loads use AP_SE_SNAPSHOT_NUM_DAYS	FIN	M

Table 4.1
Parameters (Page 4 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
AP_SE_SNAPSHOT_NUM_DAYS	31	31	The number of days of snapshots to load back each run.	Number of days to include in snapshot. Used to determine AP_SE_SNAPSHOT_START_DATE, which will be calculated as TODAY - this value.	FIN	M
AP_SE_SNAPSHOT_REPLACE_FLAG	N	N	Indicates snapshots will be replaced or not. Y for replace, N for not replaced.	Y to replace the snapshot for recalculated periods when the snapshot job is run; N to just add new records and not update previously calculated records. Default value is N.	FIN	M
AP_SE_SNAPSHOT_START_DATE	2009-01-01	2009-01-01	The first date in the AP aging snapshot table in the format YYYY-MM-DD to be loaded this run.	Calculated as TODAY minus AP_SE_SNAPSHOT_START_DATE	FIN	P
AR_EE_HISTORY_DATE		20081231	The date the AR aging table is to be refreshed from. Three years back is common. For example, 20081231		FIN	P
AR_EE_HISTORY_DATE_MIN	<customer defined - usually three years back. YYYYMMDD>	20081231	The date the AP aging table is to be refreshed from. Three years back is common. For example, 20081231	Added this in 3.4.1 to handle historic loads from multiple EE sources.	FIN	M
AR_EE_HISTORY_DATE_Sxx		20081231	Inserted via WsParameterWrite	Created by procedures. - 3.4.1, using the AP_EE_HISTORY_DATE. Sxx references which EE source number this date is for.	FIN	P
AR_EE_SNAPSHOT_FREQUENCY	M	M	The frequency of snapshots in the AR aging snapshot tables. Possible values are: D for Daily, W for Weekly, M for Monthly, B for both weekly and monthly.		FIN	M

Table 4.1
Parameters (Page 5 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
AR_EE_SNAPSHOT_HIST_START_DATE	2008-12-31	2008-12-31	The first date in the AR aging snapshot table in the format YYYY-MM-DD for the first run	Date used for start of AR snapshot in History load. Used for History load only; Daily loads use AR_EE_SNAPSHOT_NUM_DAYS	FIN	M
AR_EE_SNAPSHOT_NUM_DAYS	31	31	The number of days of snapshots to load back each run.	Number of days to include in snapshot. Used to determine AR_EE_SNAPSHOT_START_DATE , which will be calculated as TODAY - this value.	FIN	M
AR_EE_SNAPSHOT_REPLACE_FLAG	N	N	Indicates snapshots will be replaced or not. Y for replace, N for not replaced.	Y to replace the snapshot for recalculated periods when the snapshot job is run; N to just add new records and not update previously calculated records. Default value is N.	FIN	M
AR_EE_SNAPSHOT_START_DATE	1996-11-12	1997-05-24	The first date in the AR aging snapshot table in the format YYYY-MM-DD	Calculated as TODAY minus AR_EE_SNAPSHOT_START_DATE	FIN	P
AR_PAYMENT_PROCESS_DAYS	60	60	Default value should be 60 days to search for any backdated payments.		COM	M
AR_PROCESS_CONNECTION_NAME		2010SE	Indicates the connection name used by the AR process job		COM	P
AR_PROCESS_DAYS	10	10	The number of days of AR to extract and process from permanent staging tables.		FIN	M
AR_PROCESS_RUNNING	N	N	Indicates if a AR process is currently running (Y,N)	Used to prevent multiple copies of the AR job from running at the same time. Under normal conditions you should not update this manually. However, if a job fails and is deleted, you may need to reset this to "N" to run the job again. But use caution and ensure that there really is no copy of the job running before changing from "N" from "Y".	FIN	P

Table 4.1
Parameters (Page 6 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
AR_PROCESS_RUNNING_JOB_NAME		DAILY_EE_A R_CHAINED	Indicates the current or last run job name for AR processing.		FIN	P
AR_REVALUE_PERIODS	30	30	The number of periods to revalue in invoice history		FIN	M
AR_SE_SNAPSHOT_FREQUENCY	M	M	The frequency of snapshots in the AR aging snapshot tables. Possible values are: D for Daily, W for Weekly, M for Monthly, B for both weekly and monthly.		FIN	M
AR_SE_SNAPSHOT_HIST_START_DATE	2009-01-01	2009-01-01	The first date in the AR aging snapshot table in the format YYYY-MM-DD for the first run	Date used for start of AR snapshot in History load. Used for History load only; Daily loads use AR_SE_SNAPSHOT_NUM_DAYS	FIN	M
AR_SE_SNAPSHOT_NUM_DAYS	31	31	The number of days of snapshots to load back each run.	Number of days to include in snapshot. Used to determine AS_EE_SNAPSHOT_START_DATE, which will be calculated as TODAY - this value.	FIN	M
AR_SE_SNAPSHOT_REPLACE_FLAG	N	N	Indicates snapshots will be replaced or not. Y for replace, N for not replaced.	Y to replace the snapshot for recalculated periods when the snapshot job is run; N to just add new records and not update previously calculated records. Default value is N.	FIN	M
AR_SE_SNAPSHOT_START_DATE	2009-01-01	1997-05-24	The first date in the AR aging snapshot table in the format YYYY-MM-DD to be loaded this run.	Calculated as TODAY minus AR_SE_SNAPSHOT_START_DATE	FIN	P
BUDGET_ACTUAL_ROW	ACTUAL	ACTUAL	The source data for the actual row in the budget dimension		FIN	M

Table 4.1
Parameters (Page 7 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
COM_PROCESS_CONNECTION_NAME		2010SE	Indicates the connection name used by the common process job		COM	P
COM_PROCESS_DAYS	750		The number of days for Dimension tables to extract and process from load tables based on last update date.		COM	M
COM_PROCESS_RUNNING	N	N	Indicates if a Common process is currently running (Y,N)	Used to prevent multiple copies of the COMMON job from running at the same time. Under normal conditions you should not update this manually. However, if a job fails and is deleted, you may need to reset this to "N" to run the job again. But use caution and ensure that there really is no copy of the job running before changing from "N" from "Y".	COM	P
COM_PROCESS_RUNNING_JOB_NAME		DAILY_PERM_2010SE	Indicates the current or last run job name for COMMON processing.		COM	P
CORP_CALENDAR_DOMAIN	<Enter domain here for corporate calendar setup>	10USA	The Domain Code used for the corporate financial calendar	Not used currently	COM	M
CORP_CALENDAR_SOURCE	<Enter data source name here for corporate calendar setup>	QMI	The Source System Code used for the corporate financial calendar	Not used currently	COM	M
CORP_CURRENCY_CODE	USD	USD	Corporate currency code used for high level reporting		COM	M

Table 4.1
Parameters (Page 8 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
DAILY_HISTORY_DATE_SET_CURRENT	Y	Y	Default Y indicates jobs will always set the SALES and PO HISTORY DATE to the current date. If there is a need to run loads using a date prior to today, set the Value to N and change the values of the history dates to the date needed.	3.4.1 - Added to determine if the SALES_ORDER_HISTORY_DATES and PO_ORDER_HISTORY_DATES that are set by their respective setp_ jobs should increment by one (better for testing purposes) or be set to current date (presuming that current date minus process days is greater than the last _HISTORY_DATE to avoid data gaps).	COM	P
DAILY_LOAD_JOB_CURNO	0	21	Current Daily Load Job Number		COM	P
DAILY_LOAD_JOB0001		SET_CONNEC TION_QMI	Inserted via WsParameterWrite		COM	P
DAILY_LOAD_JOB0002 - DAILY_LOAD_JOBxxxx	Does not exist until INITIAL_JOB_S ETUP is run; jobs and their order determined by INITIAL_JOB_S ETUP job; should be a complete set of jobs for each source, in order		Inserted via WsParameterWrite		COM	P
DAILY_LOAD_SOURCE_CURNO	0	0	Current Source System Number being run		COM	P
DAILY_LOAD_SOURCE_S01	<Enter data source name here for initial setup>	QMI	Name of the source system loaded as the 01 job.	The connection name of the first or primary source system.	COM	M
DAILY_LOAD_SOURCE_S02 - DAILY_LOAD_SOURCE_Sxx	Does not exist by default; add if additional source(s)	2010SE		If needed, the connection name(s) of the second and subsequent source systems	COM	M

Table 4.1
Parameters (Page 9 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
DAILY_LOAD_SOURCE_S99	Dummy Parameter	Dummy Parameter	Add more by just inserting more parameters	Add parameters with consecutive numbers for each additional source system.	COM	Do Not Edit
DATE_EARLIEST_POSSIBLE	1900-01-01	1900-01-01	The latest date value possible used for unknown dates and setting history dates		COM	M
DATE_LATEST_POSSIBLE	2999-12-31	2999-12-31	The earliest date value possible used for unknown dates and setting history dates		COM	M
DEFAULT_ACCOUNT	00000	00000	The default value used for unknown account	Must not be a real account code	COM	M
DEFAULT_COST_CENTER	0000	0000	The default value used for unknown cost center	Must not be a real cost center code	COM	M
DEFAULT_DOMAIN	00000	Unknown Domain	The default value used for unknown domain	Must not be a real domain code	COM	M
DEFAULT_ENTITY	0000	0000	The default value used for unknown entity	Must not be a real entity code	COM	M
DEFAULT_PROJECT	00000	00000	The default value used for unknown project	Must not be a real project code	COM	M
DEFAULT_SOURCE		Unknown Source	The default value used for unknown source system	Must not be a real source system code	COM	M
DEFAULT_SUB_ACCOUNT	0000000	0000000	The default value used for unknown sub-account	Must not be a real sub-account code	COM	M
FIN_FIRST_DAY_NAME	SUN	SUN	The short day name of the first day of the week. Used in financial calendar derivation.		COM	M
GL_EE_POST_MAX	0	0	0		FIN	P

Table 4.1

Parameters (Page 10 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
GL_EE_POST_MAX_Sxx	Does not exist until first EE GL load	441563	DON'T UPDATE MANUALLY. Maximum Posting Id to extract this iteration.		FIN	P
GL_EE_POSTLINE_MAX	0	0	0		FIN	P
GL_EE_POSTLINE_MAX_Sxx	Does not exist until first EE GL load	441569	DON'T UPDATE MANUALLY. Maximum Posting Line Id to extract this iteration.		FIN	P
GL_EE_SOURCE_Sxx	Does not exist until first EE GL load	QMI	Inserted via WsParameterWrite		FIN	P
GL_HISTORY_LOAD_REQUIRED	Y	N	Controls initial processing of loading of historical GL data. Always 'N' unless an initial history load is required		FIN	P
GL_LOAD_DAYS	152	52	The number of days to load from the look back days forward for GL related transactional tables		FIN	M
GL_LOAD_LOOKBACK_DAYS	150	50	The number of days to look back for changes in QAD in a table for GL related transactional tables		FIN	M
GL_LOAD_PERIOD_FROM	199301	199705	The starting period for changes in QAD in a table for GL related transactional tables. Calculated using parameter GL_LOAD_LOOKBACK_DAYS.		FIN	P

Table 4.1
Parameters (Page 11 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
GL_LOAD_PERIOD_TO	201002	201012	The finishing period for changes in QAD in a table for GL related transactional tables. Calculated using parameters GL_LOAD_LOOKBACK_DAYS and GL_LOAD_DAYS.		FIN	P
GL_PERIOD1_FISCAL_ACCOUNT_ZERO_TYPES		I,E	Account types to set to zero in fact_gl_balances when a new fiscal period starts. If null, none are set. The proper values in this parameter should be a list of account codes such that the string would be appropriate in an "IN" clause.		FIN	M
GL_POST_LOOP_NO	0	0			FIN	P
GL_PROCESS_CONNECTION_NAME		2010SE	Indicates the connection name used by the GL process job		FIN	P
GL_PROCESS_DAYS	10	10	The number of days of GL to extract and process from permanent staging tables.		FIN	M
GL_PROCESS_RUNNING	N	N	Indicates if a GL process is currently running (Y,N)	Used to prevent multiple copies of the GL job from running at the same time. Under normal conditions you should not update this manually. However, if a job fails and is deleted, you may need to reset this to "N" to run the job again. But use caution and ensure that there really is no copy of the job running before changing from "N" from "Y".	FIN	P

Table 4.1

Parameters (Page 12 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
GL_PROCESS_RUNNING_JOB_NAME		DAILY_EE_GL_CHAINED	Indicates the GL process that is currently running or the GL process that last ran.		FIN	P
GL_YEAR_END_ACCOUNT	121100	121100	The default account for the year end carry over amount	Not used currently	FIN	M
GLTR_HIST_CONNECTION_NAME		2010SE	Inserted via WsParameterWrite		FIN	P
GLTR_HIST_DOMAIN		'QAD','QP','QPAU','train'	Inserted via WsParameterWrite		FIN	P
GLTR_HIST_LOAD_MAXSIZE	500000	500000	The maximum number of rows to load in an iteration loading gltr_hist		FIN	P
GLTR_HIST_LOOP_NO	0	0	DON'T UPDATE MANUALLY. The iteration number loading tr_hist.		FIN	P
GLTR_HIST_ROW_COUNT	0	4919	DON'T UPDATE MANUALLY. Number of rows loaded into gltr_hist.		FIN	P
GLTR_HIST_TRAN_MAX	0	200704110004148000.00	DON'T UPDATE MANUALLY. Maximum Transaction Id to extract this iteration.		FIN	P
GLTR_HIST_TRAN_MIN	-1	-1	DON'T UPDATE MANUALLY. Minimum Transaction Id to extract this iteration.		FIN	P
HIST_LOAD_EE_AP_RUNNING	N	C	Indicates the EXTRA_DAY_EE_AP_HISTORY load processes is underway. C for completed, Y for yes or N for no.	Added in 3.4.1 – Used by the HIST_EE_AP_AR_FINISH_CHECK job to determine if it can let the next set of jobs run.	FIN	P

Table 4.1
Parameters (Page 13 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
HIST_LOAD_EE_AR_RUNNING	N	C	Indicates the EXTRA_DAY_EE_AR_HI STORY load processes is underway. C for completed, Y for yes or N for no.	Added in 3.4.1 – Used by the HIST_EE_AP_AR_FINISH_CHECK job to determine if it can let the next set of jobs run.	FIN	P
HIST_LOAD_GLTR_HIST_RUNNING	N	C	Indicates a gltr_hist history load processes is underway. C for completed, Y for yes or N for no.		FIN	P
HIST_LOAD_JOB_CURNO	0	21	Current History Load Job Number		COM	P
HIST_LOAD_JOB_PREFIX	HIST_LOAD_	HIST_LOAD_	Base name of the History Load job		COM	P
HIST_LOAD_JOB0001		SET_CONNEC TION_QMI	Inserted via WsParameterWrite		COM	P
HIST_LOAD_JOB0002 - HIST_LOAD_JOBxxxx	Does not exist until INITIAL_JOB_SETUP is run; jobs and their order determined by INITIAL_JOB_SETUP job; should be a complete set of jobs for each source, in order		Inserted via WsParameterWrite		COM	P
HIST_LOAD_SOURCE_CURNO	1	1	Current Source System Number being run		COM	P
HIST_LOAD_SOURCE_S01	<Enter data source name here for initial setup>	QMI	Name of the source system loaded as the 01 job.		COM	M
HIST_LOAD_SOURCE_S02 - HIST_LOAD_SOURCE_Sxx	Does not exist by default; add if additional source	2010SE			COM	M

Table 4.1

Parameters (Page 14 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
HIST_LOAD_SOURCE_S99	Add more by just inserting more parameters	Add more by just inserting more parameters	Dummy Parameter		COM	Do Not Edit
HIST_LOAD_TR_HIST_RUNNING	N	C	Indicates a tr_hist history load processes is underway. C for completed, Y for yes or N for no.		COM	P
IH_INVOICETYPE	0	ih_hist.ih_invoicetype	Internal use only. Specifies the column name in ih_hist for invoice_type, or 0 if not supported on this version of the source system. There are two invoice type columns in the EE version of ih_hist. This allows us to capture both.	When extracting from EE sources, invoice type needs to reference a new column in the ih_hist table.	OPS	P
INITIAL_JOB_SETUP_CONNECTION_01	<Enter data source name here for initial setup>	QMI	Name of the connection 01 used as a source system for cloning jobs.		COM	M
INITIAL_JOB_SETUP_CONNECTION_01_RUN	N	Y	This determines if the Initial Job Setup has been run against this data source yet or not. Any new data source should have an N. Value is set to C when historic load has completed.	Will be used by INITIAL_JOB_SETUP job to ensure that existing data sources are NOT re-run as part of the historic load when new sources are added.	COM	M/P
INITIAL_JOB_SETUP_CONNECTION_01_TYPE	<Enter source type - SE or EE>	EE	Connection Type - SE for Standard Edition Financials, and EE for Enterprise Edition Financials. Defaults to SE	Used to determine whether to set up EE Financial or SE Financial jobs for this data source.	COM	M

Table 4.1
Parameters (Page 15 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
INITIAL_JOB_SETUP_CONNECTION_02		2010SE	Name of the connection 02 used as a source system for cloning jobs.		COM	M
INITIAL_JOB_SETUP_CONNECTION_02_RUN		Y	This determines if the Initial Job Setup has been run against this data source yet or not. Any new data source should have an N.	Added in 3.4.1 – Will be used by INITIAL_JOB_SETUP job to ensure that existing data sources are NOT re-run as part of the historic load when new sources are added.	COM	M/P
INITIAL_JOB_SETUP_CONNECTION_02_TYPE		SE	Connection Type - SE for Standard Edition Financials, and EE for Enterprise Edition Financials. Defaults to SE		COM	M
INITIAL_JOB_SETUP_CONNECTION_99	Add more by just inserting more parameters	Add more by just inserting more parameters	Dummy Parameter		COM	
INITIAL_JOB_SETUP_DATE	<Set to today's date before starting the initial history load>	Set to today's date (current date) so INITIAL_JOB_SETUP uses this date to populate various other parameters. YYYY-MM-DD	Added in 3.4.1. INITIAL_JOB_SETUP uses this date to populate the majority of the previously manually populated date fields for job setup.		COM	M

Table 4.1
Parameters (Page 16 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
INITIAL_JOB_SETUP_ENABLED	Y	N	Indicates the job cloning functionality is unlocked. This is a Y or N flag - initially Y then change to N when job cloning has been completed.	The INITIAL_JOB_SETUP job will not do anything if this is set to "N". If set to "Y", running the INITIAL_JOB_SETUP job will create the jobs for each source defined in the DAILY and HIST source parameters, then set the value back to "N" so that the initial job cannot be accidentally run again.	COM	M/P
INITIAL_JOB_SETUP_JOBINJOB_01	DAILY_XXXXX XXX	DAILY_XXXX XXXX	Name of the 01 job that contains other jobs.		COM	Do Not Edit
INITIAL_JOB_SETUP_JOBINJOB_02	HIST_LOAD_X XXXXXXXX	HIST_LOAD_ XXXXXXXX	Name of the 02 job that contains other jobs.		COM	Do Not Edit
INITIAL_JOB_SETUP_JOBINJOB_03			Name of the 03 job that contains other jobs.		COM	Do Not Edit
INITIAL_JOB_SETUP_JOBINJOB_04			Name of the 04 job that contains other jobs.		COM	Do Not Edit
INITIAL_JOB_SETUP_JOBINJOB_99	Add more by just inserting more parameters	Add more by just inserting more parameters	Dummy Parameter		COM	Do Not Edit
INV_BALANCE_HIST_START_DATE	2008-01-01	2008-01-01	The START date used for Inventory Transactions history loading		OPS	M
INV_HISTORICAL_PERIOD_COUNT	-24	-24	Number of periods to use to determine Average daily issued quantity historical.		OPS	M
INV_LAST_IN_MSTR_EXTRACT		40574.53221	Last Extract Timestamp of data currently in load_in_mstr		OPS	P
INV_PROCESS_CONNECTION_NAME		2010SE	Indicates the connection name used by the Inventory process job		OPS	P

Table 4.1
Parameters (Page 17 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
INV_PROCESS_DAYS	10	10	The number of days of Inventory to extract and process from permanent Data Store tables		OPS	M
INV_PROCESS_RUNNING	N	N	Indicates if an Inventory process is currently running (Y,N)	Used to prevent multiple copies of the Inventory job from running at the same time. Under normal conditions you should not update this manually. However, if a job fails and is deleted, you may need to reset this to "N" to run the job again. But use caution and ensure that there really is no copy of the job running before changing from "N" from "Y".	OPS	P
INV_PROCESS_RUNNING_JOB_NAME		DAILY_INV_PROCESS_CHANGED	Indicates the current or last run job name for INVENTORY processing.		OPS	P
INV_PROJECTED_DAYS_ON_HAND_MAX	999	99999	If there is no mrp_demand to calculate projected days on hand, the number to display to represent an infinite amount of days.		OPS	M
INV_REVALUE_PERIODS	100	10	The number of periods to revalue in inventory balances		OPS	M

Table 4.1
Parameters (Page 18 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
INV_TRANS_TYPES	'CST-ADJ','CST-TR','CYC-CNT','CYC-ERR','CYC-RCNT','ISS-CHL','ISS-COR','ISS-DO','ISS-FAS','ISS-GIT','ISS-POIT','ISS-PRV','ISS-RV','ISS-SCRIP','ISS-SO','ISS-TR','ISS-UNP','ISS-WO','PLCG-ADJ','RCT-CHL','RCT-DO','RCT-FAS','RCT-GIT','RCT-PO','RCT-POIT','RCT-RS','RCT-SOR','RCT-TR','RCT-UNP','RCT-WO','RJCT-WO','TAG-CNT','CN-CNT'		Transaction types (tr_type) that are related to inventory changes		OPS	Do Not Edit
INV_TRANS_TYPES_CONS	'CN-SHIP','CN-USE','CN-ADJ','CN-RCT','CN-ISS'	'CN-SHIP','CN-USE','CN-ADJ','CN-RCT','CN-ISS'	Transaction types (tr_type) that are related to consignment inventory movements		OPS	Do Not Edit

Table 4.1
Parameters (Page 19 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
INV_TRANS_TYPES_COUNT_ERR	'CYC-ERR'	'CYC-ERR'	Transaction types (tr_type) that are related to count errors		OPS	Do Not Edit
INV_TRANS_TYPES_COUNT_TR	'CYC-CNT'	'CYC-CNT'	Transaction types (tr_type) that are related to count transactions		OPS	Do Not Edit
INV_TRANS_TYPES_CUST	'ISS-SO','RCT-SOR','ISS-COR'	'ISS-SO','RCT-SOR','ISS-COR'	Transaction types (tr_type) that are related to customers		OPS	Do Not Edit
INV_TRANS_TYPES_GIT	'ISS-GIT','ISS-POIT', 'RCT-GIT','RCT-POIT'	'ISS-GIT','ISS-POIT', 'RCT-GIT','RCT-POIT'	Transaction types (tr_type) that are related to goods in transit (GIT).		OPS	Do Not Edit
INV_TRANS_TYPES_ISSUED	'ISS-SO','ISS-COR','ISS-WO','ISS-FAS','ISS-DO','ISS-RMA'	'ISS-SO','ISS-COR','ISS-WO','ISS-FAS','ISS-DO','ISS-RMA'	Transaction types (tr_type) that are related to inventory issued		OPS	Do Not Edit
INV_TRANS_TYPES_RECEIVED	'RCT-FAS','RCT-PO','RCT-WO','RCT-DO'	'RCT-FAS','RCT-PO','RCT-WO','RCT-DO'	Transaction types (tr_type) that are related to inventory received		OPS	Do Not Edit
INV_TRANS_TYPES_RECOUNT	'CYC-RCNT'	'CYC-RCNT'	Transaction types (tr_type) that are related to re-count transactions		OPS	Do Not Edit
INV_TRANS_TYPES_SALES	'ISS-SO','ISS-COR'	'ISS-SO','ISS-COR'	Transaction types (tr_type) that are related to sales orders		OPS	Do Not Edit
INV_TRANS_TYPES_SUPPLIER	'ISS-PRV','RCT-PO'	'ISS-PRV','RCT-PO'	Transaction types (tr_type) that are related to suppliers		OPS	Do Not Edit
JOB_AP_PROCESS_BATCH_ID	0	749	The batch id for the current DAILY_AP_PROCESS job.		FIN	P

Table 4.1

Parameters (Page 20 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
JOB_AR_PROCESS_BATCH_ID	0	750	The batch id for the current DAILY_AR_PROCESS job.		FIN	P
JOB_CHAINING_ENABLED	Y	Y	Y if a job starts the next job in the list automatically as it ends, N otherwise		COM	M
JOB_COM_PROCESS_BATCH_ID	0	748	The batch id for the current DAILY_COMMON_PROCESS job.		COM	P
JOB_GL_PROCESS_BATCH_ID	0	751	The batch id for the current DAILY_GL_PROCESS job.		FIN	P
JOB_INV_PROCESS_BATCH_ID	0	754	The batch id for the current DAILY_INV_PROCESS_JOB job.		OPS	P
JOB_OP_PROCESS_BATCH_ID	0	755	The batch id for the current DAILY_OP_PROCESS_JOB job.		OPS	P
JOB_PO_PROCESS_BATCH_ID	0	752	The batch id for the current DAILY_PO_PROCESS_JOB job.		OPS	P
JOB_SALES_PROCESS_BATCH_ID	0	753	The batch id for the current DAILY_SALES_PROCESS job.		OM	P
load_XXXXXX_0	Does not exist until after a load	1301	Rowcount from native load at 31/01/11 04:36:45	One row per load table; value is number of rows loaded the last time the specified table was loaded; comment indicates date/time of last load	Various	P

Table 4.1
Parameters (Page 21 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
load_ih_hist_ih_invoicetype	0	ih_hist.ih_invoicetype	Internal use only. Specifies the column name in ih_hist for invoice_type, or 0 if not supported on this version of the source system.. There are two invoice type columns in the EE version of ih_hist. This allows us to capture both.		OM	P
load_sod_det_sod_compl_stat	N/A'	sod_det.sod_compl_stat	Internal use only. Specifies the column name in sod_det for complete state or N/A if not supported on this version of the source system. (EE vs SE)		OM	P
load_tr_hist_tr_qty_cn_adj	0	tr_hist.tr_qty_cn_adj	Internal use only. Specifies the column name in tr_hist for consignment adjustments, or 0 if not supported on this version of the source system.		COM	P
NOT_APPLICABLE_FLAG_VALUE	N	N	The value assigned for a flag field that is not applicable.		COM	M
NOT_APPLICABLE_NUM_VALUE	0	0	The value assigned for a numeric field that is not applicable.		COM	M
NOT_APPLICABLE_TEXT_VALUE	Not Applicable	Not Applicable	The value assigned for a field that is not applicable.		COM	M
OP_HIST_LOAD_MAXSIZE	5000000	5000000	The maximum number of rows to load in an iteration loading op_hist		OPS	P
OP_HIST_LOAD_RUNNING	C	C	Indicates if a load of op_hist is running (Y/N/C)		OPS	P

Table 4.1

Parameters (Page 22 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
OP_HIST_TRAN_MAX		19975	DON'T UPDATE MANUALLY. Maximum Transaction Id to extract this iteration.		OPS	P
OP_HIST_TRAN_MAX_Sxx		433	DON'T UPDATE MANUALLY. Maximum Transaction Id to extract this iteration.		OPS	P
OP_LOAD_LOOKBACK_DAYS	20	20	The number of days to look back in QAD when extracting OP data	Used to limit the amount of data extracted from wo_mstr, wod_det, and wr_route to just those open or closed between today and (today - lookback days)	OPS	M
OP_MOVING_AVG_WEEKS	13	13	The Number of Weeks to base Moving Average on.		OPS	M
OP_OEE_SNAPSHOT_FREQUENCY	B	B	The frequency of snapshots in the OP OEE snapshot tables. Possible values are: W for Weekly, M for Monthly, B for both weekly and monthly.		OPS	M
OP_PROCESS_CONNECTION_NAME		QMI	Indicates the connection name used by the OP process job.		OPS	P
OP_PROCESS_DAYS	5	5	The number of days of OP to extract and process from permanent staging tables.		OPS	M
OP_PROCESS_RUNNING	N	N	Indicates if a OP process is currently running (Y,N)		OPS	P
OP_PROCESS_RUNNING_JOB_NAME		DAILY_OP_PROCESS_CHAINED	Indicates the current or last run job name for module processing.		OPS	P

Table 4.1
Parameters (Page 23 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
PERM_HIST_DATE	1900-01-01	1900-01-01	History Date for Permanent Staging Tables		COM	P
PERM_HIST_PREFIX	HIST	HIST	History load job name prefix		COM	Do Not Edit
PO_BOOKING_DATE	<Set to today's date before starting the initial history load>	2011-01-28	The processing date used for PO bookings processing. Format is YYYY-MM-DD		OPS	M/P
PO_BOOKING_FIRST_DATE	<Set to today's date before starting the initial history load>	2011-01-28	The first processing date for loading bookings. Format is YYYY-MM-DD		OPS	M/P
PO_EARLY_TOLERANCE_DELAY	2	4	Amount of days an order can be early and still be considered as "on-time".		OPS	M
PO_LATE_TOLERANCE_DELAY	2	1	Number of days a PO can be late and still considered as on time.		OPS	M
PO_ORDER_HISTORY_DATE	<Set to today's date before starting the initial history load>	2011-01-28	The processing date used for PO orders history calculations. Format is YYYY-MM-DD		OPS	P
PO_PERCENT_QTY_OVER	10	10	Percentage of Receipt Quantity over Order Quantity for a Purchase Order to still be considered In Full		OPS	M
PO_PERCENT_QTY_UNDER	20	20	Percentage of Receipt Quantity under Order Quantity for a Purchase Order to still be considered In Full		OPS	M

Table 4.1
Parameters (Page 24 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
PO_PROCESS_CONNECTION_NAME		2010SE	Indicates the connection name used by the PO process job.		OPS	P
PO_PROCESS_DAYS	10	10	The number of days of PO to extract and process from permanent staging tables.		OPS	M
PO_PROCESS_RUNNING	N	N	Indicates if a PO process is currently running (Y,N)	Used to prevent multiple copies of the Purchasing job from running at the same time. Under normal conditions you should not update this manually. However, if a job fails and is deleted, you may need to reset this to "N" to run the job again. But use caution and ensure that there really is no copy of the job running before changing from to "N" from "Y".	OPS	P
PO_PROCESS_RUNNING_JOB_NAME		DAILY_PO_PROCESS_2010SE	Indicates the current or last run job name for PO processing.		OPS	P
PO_RET_QTY_TOLERANCE	15	15	Percentage of Returns Quantity used for calculating Defect Rate.		OPS	M
PO_SNAPSHOT_FREQUENCY	B	M	The frequency of snapshots in the PO order snapshot table. Possible values are: W for Weekly, M for Monthly, B for both weekly and monthly.		OPS	M
PO_SNAPSHOT_HIST_START_DATE	<Set to today's date before starting the initial history load>	2011-01-28	The first date in the PO order snapshot table in the format YYYY-MM-DD for the first run		OPS	P

Table 4.1
Parameters (Page 25 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
PO_SNAPSHOT_NUM_DAYS	31	31	The number of days of snapshots to load back each run.		OPS	M
PO_SNAPSHOT_REPLACE_FLAG	N	N	Indicates snapshots will be replaced or not. Y for replace, N for not replaced.	Y to replace the snapshot for recalculated periods when the snapshot job is run; N to just add new records and not update previously calculated records. Default value is N.	OPS	M
PO_SNAPSHOT_START_DATE		1997-05-24	Internal use only. The setup procedure for the PO snapshot tables updates this parameter to determine the date at which to start.		OPS	P
QAD_MASTER_CONNECTION_NAME	QAD_MASTER	QAD_MASTE R	The name of the master generic connection used for all load tables		COM	Do Not Edit
SALES_BOOKING_DATE	<Set to today's date before starting the initial history load>	2011-01-31	The processing date used for SALES bookings processing. Format is YYYY-MM-DD		OM	P
SALES_BOOKING_FIRST_DATE	<Set to today's date before starting the initial history load>	2011-01-28	The first processing date for loading bookings. Format is YYYY-MM-DD		OM	P
SALES_LOAD_DAYS	3000	5000	The number of days to load from the look back days forward for SALES related transactional tables		OM	M
SALES_LOAD_LOOKBACK_DAYS	2000	5000	The number of days to look back for changes in QAD in a table for SALES related transactional tables		OM	M

Table 4.1
Parameters (Page 26 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
SALES_ORDER_HISTORY_DATE	<Set to today's date before starting the initial history load>	2011-01-30	The processing date used for SALES orders history calculations Format is YYYY-MM-DD		OM	M
SALES_ORDER_PERF_DATE		2011-01-28	The processing date used for SALES order performance table Format is YYYY-MM-DD		OM	P
SALES_ORDER_REVALUE_DAYS	10	10	The number of days after the end of the period to wait for revalue exchange rates to be entered before revaluing order history		OM	M
SALES_ORDER_REVALUE_PERIODS	1	1	The number of periods to revalue in order history		OM	M
SALES_PROCESS_CONNECTION_NAME		2010SE	Indicates the connection name used by the SALES process job		OM	P
SALES_PROCESS_DAYS	10	20	The number of days of sales to extract and process from permanent staging tables.		OM	M
SALES_PROCESS_RUNNING	N	N	Indicates if a SALES process is currently running (Y,N)	Used to prevent multiple copies of the Order Management/Sales job from running at the same time. Under normal conditions you should not update this manually. However, if a job fails and is deleted, you may need to reset this to "N" to run the job again. But use caution and ensure that there really is no copy of the job running before changing from "N" from "Y".	OM	P

Table 4.1
Parameters (Page 27 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
SALES_PROCESS_RUNNING_JOB_NAME		DAILY_SALES_PROCESS_2010SE	Inserted via WsParameterWrite		OM	P
SALES_SHIPMENT_FILTER	N	N	Filter the shipment data to remove consignments		OM	M
SALES_SNAPSHOT_DAYS	31	31	Number of days to look back when processing sales order snapshot		OM	M
SALES_SNAPSHOT_DOW	1	1	Day of Week Number of the Sales Snapshot Extract. Use % for all days.		OM	M
SALES_SNAPSHOT_FREQUENCY	M	M	The frequency of snapshots in the Sales order snapshot table. Possible values are: W for Weekly, M for Monthly, B for both weekly and monthly.		OM	M
SALES_SNAPSHOT_HIST_START_DATE	<Set to today's date before starting the initial history load>	2011-05-24	Used by the HIST_SALES_ROLLUP job to determine the date at which to start taking Sales order snapshots. Set to today's date by the INITIAL_SETUP_JOB, but can be set to an earlier date by the user if more history in the snapshot is desired.		OM	P
SALES_SNAPSHOT_REPLACE_FLAG	N	N	Indicates snapshots will be replaced or not. Y for replace, N for not replaced.	Y to replace the snapshot for recalculated periods when the snapshot job is run; N to just add new records and not update previously calculated records. Default value is N.	OM	M

Table 4.1

Parameters (Page 28 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/Programmatic
SALES_SNAPSHOT_START_DATE		2011-05-24	Internal use only. The setup procedure for the Sales snapshot table updates this parameter to determine the date at which to start by subtracting the SALES_SNAPSHOT_DAY S value from the current date.		OM	P
TR_HIST_CONNECTION_NAME		2010SE	Inserted via WsParameterWrite		COM	P
TR_HIST_DOMAIN		'AUTO','QP','QPAU','QPCAN','QPMEX','train'	DON'T UPDATE MANUALLY. Domain to extract this iteration		COM	P
TR_HIST_DOMAIN_S01	Does not exist until after a load of first source's data	'10USA','11CAN','12MEX','20FRA','21NL','22UK','30CHN','31AUS','CAN','FRA','QMSEMEA','QMSUS','Train','UK','USA'	Inserted via WsParameterWrite		COM	P
TR_HIST_DOMAIN_S02	Does not exist until after a load of second source's data, if applicable	'AUTO','QP','QPAU','QPCAN','QPMEX','train'	Inserted via WsParameterWrite		COM	P
TR_HIST_LOAD_MAXSIZE	1000000	1000000	The maximum number of rows to load in an iteration loading tr_hist		COM	P
TR_HIST_LOOP_NO	0	0	DON'T UPDATE MANUALLY. The iteration number loading tr_hist.		COM	P

Table 4.1
Parameters (Page 29 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
TR_HIST_ROW_COUNT	0	24186	DON'T UPDATE MANUALLY. Number of rows loaded into tr_hist.		COM	P
TR_HIST_SOURCE_S01	Does not exist until after a load of first source's data	QMI	Inserted via WsParameterWrite		COM	P
TR_HIST_SOURCE_S02	Does not exist until after a load of second source's data, if applicable	2010SE	Inserted via WsParameterWrite		COM	P
TR_HIST_TRAN_MAX		2134	DON'T UPDATE MANUALLY. Maximum Transaction Id to extract this iteration.		COM	P
TR_HIST_TRAN_MAX_S01	Does not exist until after a load of first source's data	27971	DON'T UPDATE MANUALLY. Maximum Transaction Id to extract this iteration.		COM	P
TR_HIST_TRAN_MAX_S02	Does not exist until after a load of second source's data, if applicable	17536	DON'T UPDATE MANUALLY. Maximum Transaction Id to extract this iteration.		COM	P
TR_HIST_TRAN_MIN	-1	-1	DON'T UPDATE MANUALLY. Minimum Transaction Id to extract this iteration.		COM	P
TR_QTY_CN_ADJ	tr_hist.tr_qty_cn_adj	0	Internal use only. Specifies the column name in tr_hist for consignment adjustments, or 0 if not supported on this version of the source system.	Will be set internally; do not set manually. Used by load_tr_hist and load_tr_hist_history	COM	P

Table 4.1
Parameters (Page 30 of 30)

Parameter Name	Parameter Default Value	Parameter Sample Value	Parameter Comments	Explanation/Notes	Module	Manual/ Programmatic
UNKNOWN_NUM_VALUE	9999	9999	Default value to insert for a column containing NULLs. Used for Primary Key Columns.		COM	M
UNKNOWN_VALUE	Unknown	Unknown	Default value to insert for a column containing NULLs. Used for Primary Key Columns.		COM	M

Table 4.2
System Information

Related to:	Information Needed
Progress Database	Have they already set up ODBC connections to Progress? If not, please answer the following Progress database questions for each data source being implemented.
Progress Database	ERP Version
Progress Database	Progress Version (must be 10.1B03 or above)
Progress Database	Preferred connection name.
Progress Database	Host Name
Progress Database	Port Number
Progress Database	Database Name
Progress Database	User ID
Progress Database	Password
Tomcat	URL
Tomcat	Port Number
SQL Server	Host Name
SQL Server	Port Number
Email for Portal	Host Name
Email for Portal	Port Number
Email for Portal	Authorization? Y or N

Table 4.3

Parameter Values that Need to Be Set Before History Load (Page 1 of 4)

Parameter Name	Parameter Default Value	Parameter Comments	Notes	Module
AP_EE_HISTORY_DATE_MIN	<customer defined - usually three years back. YYYYMMDD>	The date the AP aging table is to be refreshed from. Three years back is common. For example, 20081231		FIN
AP_EE_SNAPSHOT_HIST_START_DATE	2008-12-31	The first date in the AP aging snapshot table in the format YYYY-MM-DD for the first run	Date used for start of AP snapshot in History load. Used for History load only; Daily loads use AP_EE_SNAPSHOT_NUM_DAYS	FIN
AP_SE_SNAPSHOT_HIST_START_DATE	2009-01-01	The first date in the AP aging snapshot table in the format YYYY-MM-DD for the first run		FIN
AR_EE_HISTORY_DATE_MIN	<customer defined - usually three years back. YYYYMMDD>	The date the AP aging table is to be refreshed from. Three years back is common. For example, 20081231		FIN
AR_EE_SNAPSHOT_HIST_START_DATE	2008-12-31	The first date in the AR aging snapshot table in the format YYYY-MM-DD for the first run		FIN

Table 4.3
Parameter Values that Need to Be Set Before History Load (Page 2 of 4)

Parameter Name	Parameter Default Value	Parameter Comments	Notes	Module
AR_SE_SNAPSHOT_HIST_START_DATE	2009-01-01	The first date in the AR aging snapshot table in the format YYYY-MM-DD for the first run		FIN
CORP_CURRENCY_CODE	USD	Corporate currency code used for high level reporting	The currency code for corporate amounts calculations. Same for entire warehouse.	COM
DAILY_LOAD_SOURCE_S01	<Enter data source name here for initial setup>	Name of the source system loaded as the 01 job.	The connection name of the first or primary source system.	COM
DAILY_LOAD_SOURCE_S02 - DAILY_LOAD_SOURCE_Sxx	Does not exist by default; add if additional source(s)		If needed, the connection name(s) of the second and subsequent source systems	COM
DEFAULT_ACCOUNT	00000	The default value used for unknown account	Must not be a real account code	COM
DEFAULT_COST_CENTER	0000	The default value used for unknown cost center	Must not be a real cost center code	COM
DEFAULT_DOMAIN	00000	The default value used for unknown domain	Must not be a real domain code	COM
DEFAULT_ENTITY	0000	The default value used for unknown entity	Must not be a real entity code	COM
DEFAULT_PROJECT	00000	The default value used for unknown project	Must not be a real project code	COM
DEFAULT_SOURCE		The default value used for unknown source system	Must not be a real source system code	COM

Table 4.3

Parameter Values that Need to Be Set Before History Load (Page 3 of 4)

Parameter Name	Parameter Default Value	Parameter Comments	Notes	Module
DEFAULT_SUB_ACCOUNT	0000000	The default value used for unknown sub-account	Must not be a real sub-account code	COM
GL_PERIOD1_FISCAL_ACCOUNT_ZERO_TYPES		Account types to set to zero in fact_gl_balances when a new fiscal period starts. If null, none are set. The proper values in this parameter should be a list of account codes such that the string would be appropriate in an "IN" clause.		FIN
HIST_LOAD_SOURCE_S01	<Enter data source name here for initial setup>	QMI	Name of the source system loaded as the 01 job.	COM
HIST_LOAD_SOURCE_S02 - HIST_LOAD_SOURCE_Sxx	Does not exist by default; add if additional source	2010SE		COM
INITIAL_JOB_SETUP_CONNECTION_01	<Enter data source name here for initial setup>	Name of the connection 01 used as a source system for cloning jobs.		COM
INITIAL_JOB_SETUP_CONNECTION_01_RUN	N	This determines if the Initial Job Setup has been run against this data source yet or not. Any new data source should have an N. Value is set to C when historic load has completed.	Will be used by INITIAL_JOB_SETUP job to ensure that existing data sources are NOT re-run as part of the historic load when new sources are added.	COM

Table 4.3
Parameter Values that Need to Be Set Before History Load (Page 4 of 4)

Parameter Name	Parameter Default Value	Parameter Comments	Notes	Module
INITIAL_JOB_SETUP_CONNECTION_01_TYPE	<Enter source type - SE or EE>	Connection Type - SE for Standard Edition Financials, and EE for Enterprise Edition Financials. Defaults to SE	Used to determine whether to set up EE Financial or SE Financial jobs for this data source.	COM
INITIAL_JOB_SETUP_CONNECTION_02		Name of the connection 02 used as a source system for cloning jobs.		COM
INITIAL_JOB_SETUP_CONNECTION_02_RUN		This determines if the Initial Job Setup has been run against this data source yet or not. Any new data source should have an N.	Added in 3.4.1 – Will be used by INITIAL_JOB_SETUP job to ensure that existing data sources are NOT re-run as part of the historic load when new sources are added.	COM
INITIAL_JOB_SETUP_CONNECTION_02_TYPE		Connection Type - SE for Standard Edition Financials, and EE for Enterprise Edition Financials. Defaults to SE		COM
INITIAL_JOB_SETUP_DATE	<Set to today's date before starting the initial history load>	Added in 3.4.1. INITIAL_JOB_SETUP uses this date to populate the majority of the previously manually populated date fields for job setup.		COM

Upgrading to QAD BI 3.6 from QAD BI 3.5

This chapter includes instructions for upgrading your QAD BI 3.5 environment to the current version.

BI Metadata Upgrade Procedure 78

BI Portal Upgrade Procedure 82

Data Warehouse Tables Changed 83

BI Metadata Upgrade Procedure

Refer to the release notes for what has changed in this release.

Preparation

- 1 Plan for a couple of hours of down time.
- 2 Ensure none of the loads are currently in progress. **All jobs must be completed.**
- 3 Put all scheduled jobs On Hold until the upgrade is complete. Shut down scheduler.
- 4 Take a backup of the current warehouse database.
- 5 Before you begin, make note the the following:
 - a Note any custom DAILY_LOAD_JOBxxx jobs that appear after the final ROLLUP jobs (for example, DAILY_PO_ROLLUP, or DAILY_SALES_ROLLUP if you do not have the Purchase Orders module installed; if you have neither Sales nor Purchase Order processing, note any custom jobs that might be after the final SNAPSHOT jobs):

 - b Note any custom HIST_LOAD_JOBxxx jobs that appear after the standard ROLLUP or SNAPSHOT jobs:

Warning This upgrade will replace some load and stage tables, so if there are any customizations to these tables, ensure they are ready to be made again.

Upgrade Steps for QAD BI Metadata

- 1 Unzip the metadata for the Common module, plus any other modules you have installed, into a directory. Note that the Metadata files are now segregated by module, so you only install the modules you need.
 - All users need to run the Common metadata upgrade application.
 - If you process any of the Order Management (Sales) fact_om_* tables, also install the Order Management (OM) upgrade application.
 - If you process any of the SE and/or EE financial tables, also install the Financials upgrade application.
 - Finally, if you process Inventory and/or Purchase Order tables, install the Operations module upgrade application.

The Common upgrade should be performed first; the other upgrades can be done in any order.

- 2 Log in to Setup Administrator.
- 3 Choose Tools -> Start Logging. Select a name and location for your log file and Save.

- 4 Load the Common Module Upgrade Application as follows:
 - a Set your application directory to the new metadata/common folder.
 - b Load Application **upgradeCom v36** by right-clicking on it and choosing Install Application.
 - c Select the ODBC DSN for your DWD instance from the drop-down list.
 - d A dialog box will prompt you "Do you want to proceed now?"; click OK.

Note You can always Cancel here to review the list of new/modified objects; to continue with the load, right-click in the window and choose Proceed with Load Application.)

 - e The next dialog box is for "Application Load Properties":
 - Select Dimension on the left - on the right, click on the box next to "Existing Dimension objects will be" and select Altered.
 - Select Dimension View on the left - on the right, click on the box next to "Existing Dimension View objects will be" and select Recreated.
 - Select Stage Table on the left - on the right, click on the box next to "Existing Stage table objects will be" and select Recreated.
 - Select Permanent Stage Table on the left - on the right, click on the box next to "Existing Permanent Stage table objects will be" and select Altered.
 - Select Fact Table on the left - on the right, click on the box next to "Existing Fact table objects will be" and select Altered.
 - Click OK.
- 5 The upgrade application output may have errors that some indexes could not be dropped; this is OK as they are new indexes that do not yet exist in the database (the upgrade script tries to drop each index prior to re-creating it; the drop will fail if the index is new). Any other errors in the upgrade output other than those index errors listed above should be noted and Support should be consulted prior to continuing.
- 6 Repeat for any other modules that need to be upgraded.
 - a For Order Management, change the Application Directory to the metadata/order_management subdirectory, and install application **upgradeOM v36**.
 - b For Operations, change the Application Directory to the metadata/operations subdirectory, and install application **upgradeOps v36**. Note that you may see an error here about the inability to create index dim_po_line_status_idx_A; this will be corrected when the index is rebuilt after running the Upgrade jobs as described below.
 - c For Financials (both EE and SE are in the same metadata package), change the Application Directory to the metadata/financials subdirectory, and install application **upgradeFin v36**.
- 7 The upgrade application installations will take some time as the load process alters/recreates various objects. When you have installed all pertinent upgrade applications, if any of the tables that were modified had included customizations, re-apply those customizations to your upgraded tables. Verify that all other customizations are not affected.
- 8 Start the Scheduler, but make sure all jobs are still suspended.

- 9 You must populate some new parameters and possibly modify some existing values, as follows:
- a Modify AR_PAYMENT_PROCESS_DAYS and COM_PROCESS_DAYS from the default values if desired:
 - AR_PAYMENT_PROCESS_DAYS: Used to find backdated payments in ar_mstr and ard_det tables. To save daily load time, not all records are loaded daily from ar_mstr and ard_det - only open records, plus closed records whose ar_date or ar_paid_date is between Today and Today minus AR_PAYMENT_PROCESS_DAYS. Set this to a value that will safely allow you to extract backdated payments; for example, if payments are sometimes entered with a date that is 3 months in the past, set this to approx. 90 days. However, if backdating is never more than 45 days, enter 45. Default value is 60.
 - COM_PROCESS_DAYS: Used to reduce load time by limiting the amount of data extracted from the pt_mstr, sct_det, and exr_rate tables to only those rows that have been modified between Today and Today minus COM_PROCESS_DAYS. Default value is 750.
 - b Financial module only: Modify AP_CHECK_PROCESS_DAYS from the default value if desired:
 - AP_CHECK_PROCESS_DAYS: Used to find backdated checks or vouchers in ap_mstr table. To save daily load time, not all records are loaded daily from ap_mstr - only open records, plus closed records such as checks with a back-dated void date or clear date, or vouchers with a back-dated paid date, between Today and Today minus AP_CHECK_PROCESS_DAYS. Set this to a value that will safely allow you to extract backdated rows; for example, if transactions are sometimes entered with a date that is 3 months in the past, set this to approx. 90 days. However, if backdating is never more than 45 days, enter 45. Default value is 60.
 - c Operations module only: Modify OP_LOAD_LOOKBACK_DAYS, OP_MOVING_AVG_WEEKS, OP_OEE_SNAPSHOT_FREQUENCY, and OP_PROCESS_DAYS from the default values if desired:
 - OP_LOAD_LOOKBACK_DAYS: Used to limit the amount of data extracted from wo_mstr, wod_det, and wr_route to just those open or closed between Today and Today minus OP_LOAD_LOOKBACK_DAYS. Default value is 20.
 - OP_MOVING_AVG_WEEKS: The number of weeks on which to base the Moving Average calculations. Default value is 13.
 - OP_OEE_SNAPSHOT_FREQUENCY: The frequency of snapshots in the OP OEE snapshot tables. Possible values are: W for Weekly, M for Monthly, B for both weekly and monthly. Default value is B for both.
 - OP_PROCESS_DAYS: The number of days of OP to extract and process from permanent staging tables. Default value is 5.
- 10 *****Very important***** Check the PO_ORDER_HISTORY_DATE and SALES_ORDER_HISTORY_DATE parameters. If the current date minus the PO_PROCESS_DAYS is newer than the PO_ORDER_HISTORY_DATE, or the current date minus the SALES_PROCESS_DAYS is newer than the SALES_ORDER_HISTORY_DATE, increase the process days value so that the current date minus that number of days will produce a date that is before the related ORDER_HISTORY_DATE (PO or SALES).

- 11 Delete all of your existing SET_CONNECTION_<source> jobs. (Note: do NOT delete the SET_CONNECTION_XXXXXXX job; that is the template that will be used to create new source-specific SET_CONNECTION jobs.)
- 12 Set INITIAL_JOB_SETUP_DATE to the current date.
- 13 Change INITIAL_JOB_SETUP_ENABLED to Y.
- 14 In the Scheduler, run the INITIAL_JOB_SETUP job.
- 15 The list of History and Daily jobs will have been re-written by the INITIAL_JOB_SETUP job. Add to the end of the DAILY_LOAD_JOBxxxx and HIST_LOAD_JOBxxxx load lists any jobs that occurred after the snapshot jobs (which you noted in the Preparation section).
- 16 Run the Upgrade jobs for each module you have installed:
 - a Run the UPGRADE_35_TO_36_COMMON job.
 - b If you installed the Financials module upgrade package, run the UPGRADE_35_TO_36_FINANCIALS job.
 - c If you installed the Operations module upgrade package, first set the Parameter QBI_924_PATCH_RUN to N, then run the UPGRADE_35_TO_36_OPERATIONS job. Note: the QBI_924_PATCH_RUN parameter is initially set to Y to guard against the accidental running of the patch upgrade task where it is not warranted. Set it to N just prior to running the Operations Upgrade job; after the job completes, the parameter will be removed, ensuring the QBI-924 upgrade task is not allowed to run a second time.
 - d If you installed the Order Management (Sales) module upgrade package, run the UPGRADE_35_TO_36_ORDER_MANAGEMENT job.
- 17 If the upgrade affected any customized tables, re-apply your customizations (a list of new and changed tables can be found at the end of the Upgrade instructions.)
- 18 Resume normal processing.

To Add a New Source

If you want to add an additional data source, do the following:

- 1 Add the new source name to DAILY_LOAD_SOURCE_Sxx (where xx represents the next number in the sequence).
- 2 Add the new source name to HIST_LOAD_SOURCE_Sxx (where xx represents the next number in the sequence).
- 3 Add the new source name to INITIAL_JOB_SETUP_CONNECTION_xx (where xx represents the next number in the sequence).
- 4 Add a new INITIAL_JOB_SETUP_CONNECTION_xx_RUN (if it doesn't already exist) and set its value to N.
- 5 Add a new INITIAL_JOB_SETUP_CONNECTION_xx_TYPE (if it doesn't already exist) and set its value to either EE or SE depending on which type your financials system for that source is.

- 6 Make sure INITIAL_JOB_SETUP_DATE is set to the current date.
- 7 Set INITIAL_JOB_SETUP_ENABLED to Y.
- 8 Run INITIAL_JOB_SETUP.
- 9 Double-check the INITIAL_JOB_SETUP_CONNECTION_XX_RUN parameters to make sure that any sources that have been historically loaded have a Y next to their relative _xx_ number and that any new sources have an N.
- 10 Go to the Scheduler. Start the HIST_START job. Once it starts, double check to make sure it started running HIST_ jobs only for your new data source.

BI Portal Upgrade Procedure

Preparation

- 1 Shut down Tomcat.
- 2 Back up the BI Portal database.

Upgrade Steps for QAD BI Portal

- 1 Copy the TOMCAT_HOME\webapps\your_bi_appname directory to TOMCAT_HOME\your_bi_appname.save35 so that you have a backup of your BI 3.5 web application.
- 2 Unzip the new BI 3.6 Portal data.zip file into TOMCAT_HOME\webapps\your_bi_appname, overwriting any existing files with the newer version from the data.zip file. Your configuration files will not be overwritten.
- 3 Start Tomcat.

Verification

- 1 Connect to your BI Portal. Click on the About link on the login screen and verify that the Version is **3.6** and the Build Number is **BI3-TRUNK-JOB1-1104**.

Note End users may need to clear their browser cache prior to re-connecting to the BI Portal in order to ensure they are running the latest version of the application.

Update Portal Model

- 1 Log into the Portal as an Administrator.
- 2 Go to the Administration menu and choose Model Administration. Click on Rebuild Model.
- 3 After the model has been rebuilt, if you have any custom data models, open each model and save it to ensure it picks up changes in the Master model.

Update Portal Documentation and Standard Objects

Update the metadata documentation:

- 1 Delete the existing `TOMCAT_HOME\webapps\your_bi_appname\documentation\suite` directory.
- 2 Unzip the latest `qadbi-documentation.zip` (found on the BI 3.6 Modules CD) into your `TOMCAT_HOME\webapps\your_bi_appname\documentation` directory.

To install the latest BI Portal standard objects, copy the files from the BI 3.6 Modules CD into your `TOMCAT_HOME\webapps\your_bi_appname\export` directory:

- `portal\common\qad_common_dashboards.zip`
- For each additional module you have installed, copy the `portal\<module>\qad_<module>_reports.zip` and `qad_<module>_dashboards.zip` files

Then, in the BI Portal:

- 1 Log in as an Administrator.
- 2 Go to the Administration -> Data Migration screen.
- 3 Click on the Import tab.
- 4 Check the "Update existing records" box.
- 5 Click the radio button in front of "File on server".
- 6 Enter in the box the file name `qad_common_dashboards.zip` and click on "Import".
- 7 Repeat for any other module reports and dashboards zip files that are required.

Data Warehouse Tables Changed

The following sections list the Data Warehouse tables that have been changed, added, or deleted. Note that those in the "Modified" list may have been changed structurally (for example, columns added or deleted), or they may have had indexes added or deleted, or they may have had changes made to business display names or visibility of particular columns in the BI Portal, or any combination of these modifications.

Tables Modified

Common Module Modified Tables

- | | |
|-------------------------------------|------------------------------|
| • <code>dim_account</code> | • <code>load_hd_mstr</code> |
| • <code>dim_credit_term</code> | • <code>load_icc_ctrl</code> |
| • <code>dim_customer</code> | • <code>load_in_mstr</code> |
| • <code>dim_customer_bill_to</code> | • <code>load_loc_mstr</code> |
| • <code>dim_customer_ship_to</code> | • <code>load_pj_mstr</code> |
| • <code>dim_customer_sold_to</code> | • <code>load_pl_mstr</code> |
| • <code>dim_date_qad</code> | • <code>load_pt_mstr</code> |

- dim_effective_date
- dim_item
- dim_sales_person
- dim_sales_person1
- dim_sales_person2
- dim_sales_person3
- dim_sales_person4
- load_ac_mstr
- load_ad_mstr
- load_ar_mstr
- load_budget_actual
- load_cc_mstr
- load_cm_mstr
- load_code_mstr
- load_consignment_type
- load_ct_mstr
- load_ctd_det
- load_cu_mstr
- load_dom_mstr
- load_ee_BusinessRelation
- load_ee_Creditor
- load_ee_Currency
- load_ee_Debtor
- load_ee_Domains
- load_ee_GL
- load_ee_SharedSet
- load_en_mstr
- load_exr_rate
- load_glc_cal
- load_ptp_det
- load_rnd_mstr
- load_sb_mstr
- load_sct_det
- load_service_type
- load_shop_cal
- load_si_mstr
- load_sp_mstr
- load_SYSCOLUMNS
- load_tr_hist
- load_tr_hist_history
- load_tr_hist_list
- load_transaction_type
- load_um_mstr
- load_usr_mstr
- load_vd_mstr
- perm_item_master
- perm_shop_cal
- perm_transaction_hist
- stage_account
- stage_customer
- stage_customer_master
- stage_ee_account
- stage_ee_ic_debtor
- stage_item
- stage_item_costs
- stage_sales_person
- stage_shop_cal
- stage_transaction_hist

Financials Module Modified Tables

- extract_gl_transaction_hist
- fact_ap_invoice_snapshot
- fact_ap_voucher_history
- fact_ar_invoice_history
- fact_ar_invoice_snapshot
- fact_ee_ap_invoice_snapshot
- fact_ee_ar_invoice_snapshot
- fact_gl_transaction
- load_acd_det
- load_ap_mstr
- load_ap_transaction_type
- load_ee_Division
- load_ee_DomainProperty
- load_ee_Journal
- load_ee_Layer
- load_ee_Period
- load_ee_Posting
- load_ee_PostingHist
- load_ee_PostingLine
- load_ee_Project
- load_ee_SysProperty
- load_gen_led_report

- load_ar_transaction_type
- load_ard_det
- load_bg_mstr
- load_bdg_det
- load_ck_mstr
- load_ckd_det
- load_ee_ad_mstr
- load_ee_CInvoice
- load_ee_CInvoiceMovement
- load_ee_Company
- load_ee_CompanySharedSet
- load_ee_CostCenter
- load_ee_DInvoice
- load_ee_DInvoiceMovement
- load_gen_led_report_line
- load_gen_led_report_line_calcs
- load_gen_led_report_line_range
- load_gl_transaction_type
- load_gltr_hist
- load_gltr_hist_history
- load_gltr_hist_list
- load_vo_mstr
- stage_ee_gl_balance7
- stage_ee_gl_balance8
- stage_gl_balance1
- stage_gl_balance2
- stage_gl_transaction1

Operations Module Modified Tables

- dim_po_line_status
- dim_po_line_type
- dim_shipment_type
- extract_inv_trans_hist_cons
- extract_inv_transaction_hist
- fact_inv_transaction
- fact_po_order
- fact_po_order_history
- fact_po_order_performance
- fact_po_order_receipt
- load_inv_trans_type_cntrr
- load_inv_trans_type_cntrr
- load_inv_trans_type_cons
- load_inv_trans_type_cust
- load_inv_trans_type_git
- load_inv_trans_type_issued
- load_inv_trans_type_received
- load_inv_trans_type_recount
- load_inv_trans_type_sales
- load_inv_trans_type_supplier
- load_inv_transaction_type
- load_is_mstr
- load_ld_det
- load_mrp_det
- load_po_line_status
- load_pod_det
- load_prh_hist
- load_vep_mstr
- load_vepd_det
- stage_inv_bal_revalue1
- stage_inv_transaction
- stage_inv_transaction_con_cus
- stage_inv_transaction_con_cus1
- stage_inv_transaction_con_cus2
- stage_inv_transaction_con_cus3
- stage_inv_transaction_con_cus4
- stage_inv_transaction_con_cus5
- stage_inv_transaction_con_sup
- stage_inv_transaction_con_sup1
- stage_inv_transaction_con_sup2
- stage_inv_transaction_non_cons
- stage_inv_transaction1
- stage_inv_transaction2
- stage_inv_transaction3
- stage_inv_transaction4
- stage_po_order_hist
- stage_po_order_snap1
- stage_po_order_snap3
- stage_po_order_snapshot
- stage_po_receipt_perf

- load_po_line_type
- load_po_mstr
- work_po_perf_list

Order Management Module Modified Tables

- extract_om_invoice_line
- fact_om_order_history
- fact_om_order_performance
- fact_om_order_shipment
- load_cncu_mstr
- load_idh_hist
- load_ih_hist
- load_order_line_order_type
- load_so_mstr
- load_sod_det
- perm_om_invoice_line
- stage_om_invoice_line
- stage_om_invoice_order_line
- stage_om_invoice1
- stage_om_ord_perf
- stage_om_ord_perf_date1
- stage_om_ord_perf_full
- stage_om_ord_perf_ship
- stage_om_ord_perf1
- stage_om_order_inv_line
- stage_om_order_inv_line1
- stage_om_order_inv_line2
- stage_om_order_line
- stage_om_order_line_all
- stage_om_order1
- stage_om_order3
- stage_om_shipment
- stage_om_shipment_transaction

Tables Added

Common Module Added Tables

- dim_snapshot_period_type
- load_ee_BudgetGroup
- load_ee_CashGroup
- load_ee_DomainSharedSet
- load_ee_PaymentCondition
- load_ee_PaymentGroup
- load_ee_Unit
- load_snapshot_period_type
- stage_ee_account
- stage_se_account

Financials Module Added Tables

- work_gl_trans_change_list

Operations Module Added Tables

- dim_calendar_period
- dim_department
- dim_op_oeo_periods
- dim_production_line
- dim_release_date
- dim_start_date
- dim_work_center
- stage_op_item_efficiency2
- stage_op_item_moving_avg
- stage_op_oeo_periods
- stage_op_oeo_periods_month
- stage_op_oeo_periods_week
- stage_op_prod_line_period_hrs1
- stage_op_prod_line_period_hrs2

- dim_work_center_machine
- extract_op_oeo_dates
- extract_op_transaction_hist
- fact_op_equipment_effectiveness
- fact_op_item_efficiency
- fact_op_transaction
- fact_wo_order
- fact_wo_routing_transaction
- load_cal_det
- load_dpt_mstr
- load_ln_mstr
- load_op_hist
- load_shift_det
- load_wc_mstr
- load_wo_mstr
- load_wo_mstr_hist
- load_wod_det
- load_wod_det_hist
- load_wr_route
- load_wr_route_hist
- perm_op_hist
- perm_work_center_master
- perm_work_order_detail
- perm_work_order_master
- perm_work_order_routing
- stage_calendar_period
- stage_exception_period_hrs
- stage_op_equip_effect
- stage_op_equipment_effectiveness
- stage_op_hist
- stage_op_item_efficiency
- stage_op_item_efficiency1
- stage_op_prod_line_wc_period
- stage_op_resource_avail_hrs
- stage_op_resource_avail_keys
- stage_op_tr_hist_rjct_wo
- stage_op_tr_hist_rjct_wo1
- stage_op_trans_scrapi
- stage_op_transaction
- stage_op_transaction_merge
- stage_op_transaction0
- stage_op_transaction1
- stage_op_transaction2
- stage_op_work_center_period_hrs1
- stage_op_work_order_detail_scrap
- stage_op_work_order_detail_scrap2
- stage_op_work_order_lot
- stage_op_work_order_lot_scrap
- stage_prod_line_hrs
- stage_prod_line_period_hrs
- stage_prod_line_period_hrs_ma
- stage_wo_mstr
- stage_wo_order
- stage_wo_order0
- stage_wo_routing_transaction
- stage_wo_routing_transaction0
- stage_wod_det
- stage_work_center_period_hrs
- stage_work_center_period_hrs_ma
- stage_work_center_site
- stage_work_center_site_hrs
- stage_wr_route
- work_op_wo_routing_costs
- work_op_wo_routing_costs_cum

Order Management Module Added Tables

- dim_om_service_type

Tables Deleted

No tables were deleted for Release BI 3.6.

Sizing Guidelines

This document covers sizing for the server-side BI components. Topics include:

Business Intelligence Deployment Options and Terminology 91

CPU Sizing 94

RAM Sizing 95

Disk and IO Subsystem 96

Network Considerations 98

Example Sizing 99

Reference: Estimation of Initial Database Size 100

Reference: Network Sizing Details 102

Introduction

This document is a provisional and interim sizing guide for the appropriate sizing of hardware environments ready for the deployment of the QAD Business Intelligence product (QAD BI version 3.6 and above).

- **Provisional:** A formal sizing guide is normally produced after a detailed capacity planning analysis involving a series of formal internal sizing/scalability/performance testing. The sizing estimates in this document are provisional in the sense that they have been produced in the absence of a formal capacity planning process.
- **Interim:** This sizing guide can be considered an interim sizing guide until a formal capacity planning testing process can be carried out for QAD BI.
- **Practical Sizing:** However, at the time of publication, the sizing estimates in this guide are the most accurate available to QAD, and this document should be used for practically sizing BI deployments.
- **Caveats to BI Sizing:** Although the sizing estimates given here are the best available, the absence of a formal capacity planning process means that certain estimates in this document may need to be updated as more accurate information becomes available and hence, sizing should be carried out bearing in mind all the caveats listed under the following topic.

This document covers sizing for the server-side BI components. It does not comment on client-side BI Portal (browser-based) sizing requirements.

Important Restrictions and Caveats on BI Sizing

All restrictions and caveats related to sizing are covered in this section.

Note In the absence of a formal capacity planning testing process for QAD BI, this document presents best estimate sizing guidelines for QAD BI.

At the time of publication, these estimates are the most accurate available, and should serve as an appropriate guideline for sizing QAD BI, taking the following caveats into account.

The following restrictions and caveats need to be borne in mind when estimating sizing from this document:

- **Caveat 1:** This product has not been sized using formal internal sizing/scalability/performance testing. It should therefore be understood that all sizing guidelines in this document are based on valid but incomplete information so guidelines will be updated as more sizing tests and research are carried out.
- **Caveat 2:** All effort has been made to make these sizing estimates as accurate and reliable as possible, using standard capacity planning and sizing planning best-practice. However, as we are sizing based on incomplete information, some estimates may not be 100% accurate, this should be borne in mind.

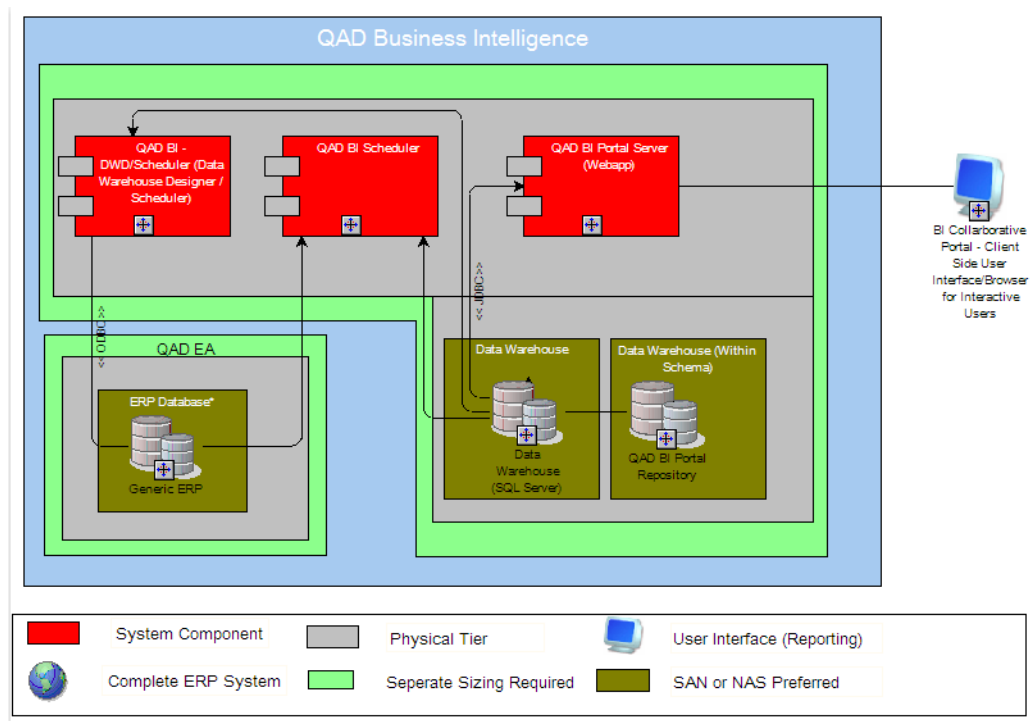
In summary, the current sizing estimates should be considered best-estimate sizing guidelines, safe to use in the field, but subject to future updates/correction and also subject to the caveats listed above.

Business Intelligence Deployment Options and Terminology

This section covers BI terminology and BI deployment options.

BI Terminology

A schematic picture of BI is provided in the following figure:



DWD - Data Warehouse Designer: The application that allows viewing and modification of metadata that describes and builds the data warehouse.

DWD Scheduler – Data Warehouse Designer Scheduler: The component of the DWD responsible for loading data (it schedules and runs jobs to extract data from a remote ERP system(s)) and building the data warehouse.

DWD/Scheduler – In this document, “DWD/Scheduler” is used to refer to all the technical components composing the DWD and the DWD Scheduler.

QAD BI Collaborative Portal (Client-Side Browser) – This is the remote client-side user interface to QAD BI. It is a flash-based interface run on a remote internet browser and provides for interactive dashboards, reports, collaboration and administration.

QAD BI Portal Server (Server-Side Webapp) - This is the server-side component of the QAD BI Portal. It is a java-based webapp (Web Application) which receives requests from the client-side BI Portal. The webapp communicates requests as queries to SQL Server and the webapp also coordinates the reply to the client-side browser.

Single Tier – “Single Tier” deployment refers to the fact that all the primary QAD BI components (DWD/Scheduler, BI Portal (Webapp) and SQL-Server) are deployed on one single server machine. In this deployment model (and all other deployment models), it is preferred and recommended that the BI data warehouse itself is placed on a separate SAN/NAS device.

Two Tier – “Two Tier” deployment refers to having the QAD BI Portal (server-side webapp) on a separate machine (server) from the DWD/Scheduler and SQL Server (the DWD/Scheduler and SQL-Server remain on a the primary server machine).

BI Deployment Options

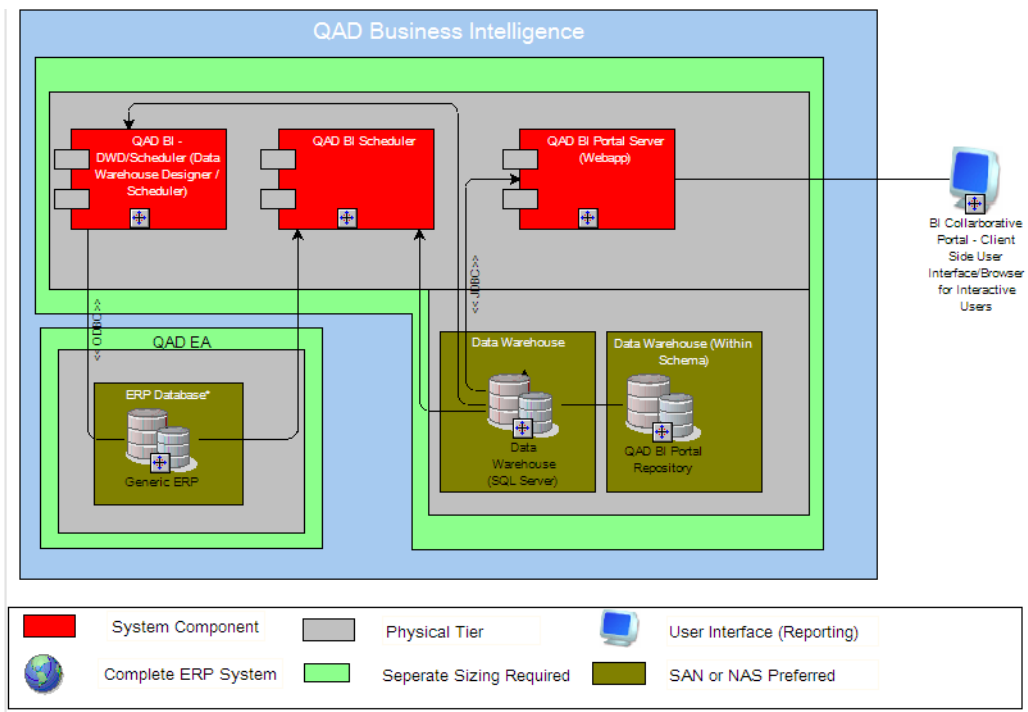
Business Intelligence involves several different components and processes which need to be sized separately and appropriately (depending on the deployment option), before determining a final sizing estimate. There are two primary deployment options available:

- Single tier (with all BI components on one server)
- Two tier (with the QAD BI Portal Server on a separate physical/virtual server to all other BI components).

Using this breakdown of BI technical components will allow us to size appropriately for either a single-tier or two-tier deployment.

Deployment Option 1 – Single Tier

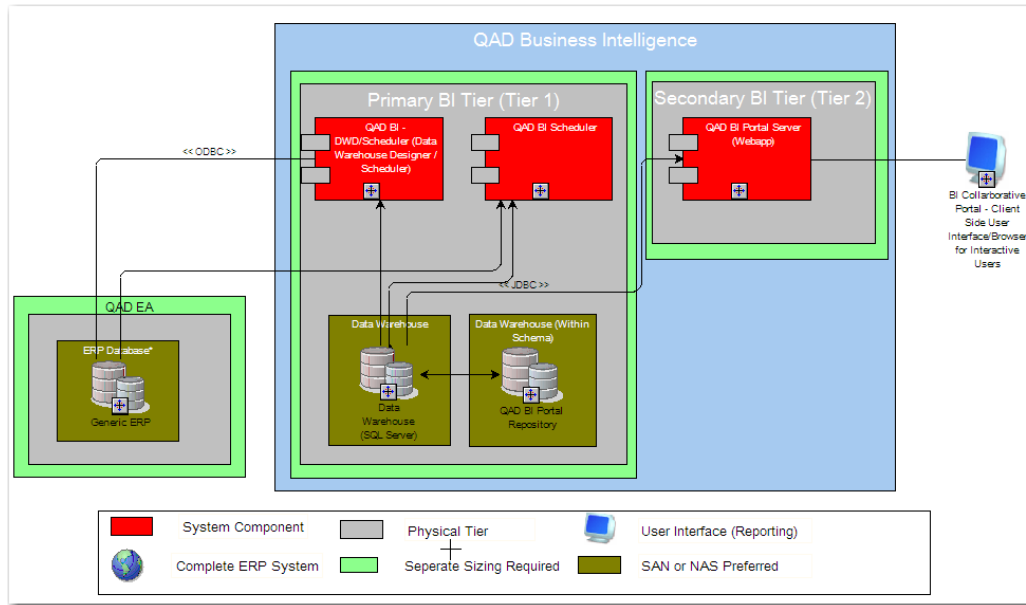
A single-tier BI deployment has all BI components on the same server (this is illustrated in “BI Terminology” on page 91 above and is repeated schematically here):



Deployment Option 2 – Two Tier

The two-tier BI deployment has the BI Portal webapp on a separate server from all other BI components. The deployment option is illustrated in the following figure.

Note The BI Portal has a much smaller resource requirement demand than the other BI components, so the size of a BI Portal server will be smaller than the BI DWD/Scheduler server. This distinction is made throughout this document.



Sizing Terminology Used Throughout this Sizing Guide

Sizing Metrics

Sizing across the following four metrics are covered separately in this guide:

- CPU
- RAM
- IO Subsystem & Disk
- Network / WAN

Sizing Methodology for QAD BI

We comment in this document on sizing for the QAD BI server-side components only.

Two different workload types place different demands on BI components. Also, there are two deployment options for QAD BI. Because of this, each type of workload must be sized for separately. The methodology used for sizing BI divides each metric into the following areas:

- Sizing for DWD/Scheduler Workload (Batch Jobs). This refers to the workload placed on the server as a result of the DWD/Scheduler automated jobs. This workload needs to be sized for separately and is of a batch-type.

- Sizing for Concurrent User Workload (Interactive Portal Users). This is the workload placed on the DW server as a result of the concurrent interactive users using the BI Portal. This workload involves concurrent active end-users effectively running queries against the BI data warehouse through their remote BI-Portal-client user interface. This workload can place significant read demand against the data warehouse and needs to be sized for separately.
- Sizing for BI Portal Webapp (Server Side). Because the BI Portal webapp can be placed on a separate tier from the rest of BI (in the two-tier deployment model), we record sizing requirements for this component separately for each relevant sizing metric. If sizing for a single-tier, combine the sizing for the BI Portal webapp with the other sizing metrics.

CPU Sizing

Note This product has not been sized using formal performance and sizing procedures. All sizing guidelines in this document are based on incomplete information and guidelines will be updated as more sizing tests and research are carried out.

CPU Sizing for DWD/Scheduler Component

Based on current information, the DWD/Scheduler processing is mostly single-threaded, meaning that it rarely requires more than one CPU core.

Hence, current guidelines suggest that for DWD/Scheduler CPU sizing, at most 2 CPU cores are required.

Sizing Metric	Recommendation	Recommendation 2
CPU – DWD Scheduler	2 CPU Cores	Use the fastest CPU cores available. Daily/Nightly DWD Scheduler jobs are CPU-bound and processing time will benefit greatly from <i>faster</i> CPU cores (not from <i>more</i> CPU cores).

CPU Sizing for BI Portal Webapp (Server Side)

Note that this is CPU sizing just for the server-side BI Portal java-based webapp. In this case, we estimate that a single CPU core is sufficient to run the BI Portal webapp.

Sizing Metric	Recommendation	Recommendation 2
CPU – BI Portal Component (java webapp)	1 CPU Core	-

CPU Sizing for Concurrent User Workload (Interactive Portal Users)

“Interactive portal users” use their remote BI-Portal-client user interface to execute queries against the primary DW server. “Concurrent User Workload” therefore refers to the load placed on the SQL-Server tier by the remote concurrent BI Portal users.

Note that multiple concurrent BI Portal users executing requests, can place a large load on the SQL-Server data server. Hence, in this section, we are sizing for the SQL-Server tier, independent of a one-tier or two-tier deployment.

Sizing Metric	Concurrent User Count	Recommendation
CPU – Concurrent BI Portal User Load	1-10	4 CPU cores
	10-20	4-6 CPU Cores
	20-40	6-8 CPU Cores
	40-60	8 CPU Cores

RAM Sizing

RAM Sizing for DWD/Scheduler Component

The amount of RAM needed will be determined by:

- The size of the data warehouse
- The amount of data being manipulated by the DWD Scheduler and SQL-Server
- The complexity and degree of customizations within the BI modules

Sizing Metric	RAM for OS (GB)	RAM for SQL-Server	Total Server RAM ¹
“Small” Load	4	6+	10+
“Medium” Load	4	8+	12+
“Large” Load	4	12+	16+

1. “Total Server RAM” assumes RAM sizing for the Windows Operating System and BI DWD Scheduler/SQL-Server only. If other components are on the server, these will have to be sized for separately and appropriate extra RAM allocated.

RAM Sizing for BI Portal Webapp (Server-Side)

Sizing Metric	Recommendation	Comment
RAM – BI Portal Technical Component	2 GB	Allocate 2GB to the BI Portal webapp. On 64-bit systems and for very large workloads (concurrent user counts), over 2 GB can be allocated.

RAM Sizing for Concurrent Users (Interactive Portal Users)

A separate server-side RAM-sizing for interactive portal users is not required in this case, because allowance has been made in the preceding subtopics for this RAM sizing. To explain:

Concurrent interactive portal user workload consists of multiple remote-client BI Portal users placing a load on the server-side BI Portal webapp and SQL-Server. Hence, “Sizing for Concurrent Users” sizes for the workload placed on the server-side components by the remote interactive portal users – allowance for this had been made in the preceding DWD/Scheduler and BI Portal Webapp sizing discussions.

Disk and IO Subsystem

Note on Importance of Appropriate Disk and IO Subsystem Sizing

Because of the heavy IO demands of each aspect of the BI product, disk and IO subsystem capacity and sizing is vitally important for product performance and scalability.

IO Subsystem Sizing

The speed and throughput of the IO subsystem is critical to the overall performance of the QAD BI application. Note that this sizing refers to the data warehouse IO subsystem.

Sizing Metric	Recommendation (IO Subsystem)
Disk Space	Predicting disk sizing requirements for QAD BI is complex (see “Key IO Subsystem Sizing Notes” below and “Reference: Estimation of Initial Database Size” on page 100 for details). The following guidelines should be followed. Recommendations 1. <u>Minimum</u> - A minimum of 500 GB disk space is recommended (but see points 5 and 6). 2. <u>Ideal</u> - Ideally 1 TB of disk space is recommended (but see points 5 and 6). General Guidelines 3. <u>Allowance for database growth</u> - To allow for BI database growth, 2-3 times the size of the original BI database is recommended.; see “Allowance for Future Database Growth” on page 102 for the exact details. 4. <u>Allowance for disk expansion</u> – If appropriate and to allow for future project growth, it is recommended to allow for disk expansion capabilities in the SAN/NAS device. 5. <u>Estimation</u> - “Reference: Estimation of Initial Database Size” on page 100 provides an approximate method of estimating the initial BI database size (from source ERP tables). This method can be used along with point 3 above (multiply “estimated” database size by a factor of 2-3 to allow for growth) to provide an estimate of the initial disk space requirements for a new implementation. Final Disk Sizing – Specific Guidelines 6. If the estimate from point 5 returns over 500 GB or over 1 TB of disk space requirements, then obviously size accordingly.
SAN/NAS?	It is preferable to use external disk arrays or SAN/NAS devices where possible. Internal disk drives should be configured to hold the operating system only.
RAID Level	RAID 5 or RAID 10 preferred.
Physical Disks	Fastest disks available: 15 k rpm SAS (or fiber channel) disks preferred. Note 1: <u>Due to the larger IO demands of BI systems, a minimum of 6 disks is recommended.</u> Double the amount of disks when using mirroring. Note 2: More disks are preferred, as this will significantly help performance and IO rates.
IO Controller	Ensure IO controller throughput does not limit transfer rates to/from disks. Where possible spread the IO across as many controllers as possible.

Key IO Subsystem Sizing Notes

- 1 In the absence of any estimates, 500 GB minimum recommendation, 1 TB ideally – but final estimate of disk space needs to be based on best-estimates (see points 2 and 3 below).
- 2 Use “Reference: Estimation of Initial Database Size” on page 100 to estimate order-of-magnitude sizing for new BI databases.
- 3 To allow for BI database growth, size total disk space for 2-5 times the size of the BI database. See “Allowance for Future Database Growth” on page 102 for details.
- 4 Fast disks – 15k SAS disks
- 5 RAID Level spread across multiple disks, recommend 6 disks minimum.
- 6 Need the associated IO controller(s) to be capable of servicing maximum throughput from each disk (or combination of disks in RAID configuration) - Ensure the IO controller(s) used do not limit transfer rates per disk below the maximum rated physical-disk and RAID transfer rates, as this can act as a bottleneck at high IO demand rates.

Notes on Database Disk Sizing

Factors Affecting Disk Sizing Estimates

Currently there is not detailed information on how to scale the expected size of the BI data warehouse from a source ERP system. The reason for this is that the size of the BI data warehouse depends on a variety of factors:

- The number of modules implemented
- The size of the source ERP tables transformed
- The number of ERP sources
- The volume of daily and historical ERP data
- The degree of customization of the BI modules

Because these factors are difficult to predict beforehand, we currently give guidelines on expected data warehouse size based on existing customer data.

Estimation of Initial Database Size

If required, refer to “Reference: Estimation of Initial Database Size” on page 100 for an approximate method of estimating the initial BI database size. Note that this estimate is very approximate in nature, and should be used only as a means of getting an order-of-magnitude estimate on database size.

Allowance for Database Growth

Depending on the daily transaction volumes, the number of BI modules and customizations, the BI database will be growing on a daily basis (depending on the daily scheduled load jobs). Allowance will need to be made up-front for this in terms of disk capacity.

The recommendation is to have a disk capacity of 2-3 times the “estimated size of the BI database.” The rule of thumb is that if the initial BI DW is based on converting 3 years of historical data, then (assuming the same rate of transactions passing through the source ERP database) expect the BI DW itself to grow by a similar volume (that is, similar volume to the DW initial size) over the coming 3 years.

Also, daily or weekly monitoring of the BI database size will provide ongoing measurement of growth rates, and adequate storage should be provided to manage this growth.

Allowance for Database Backup

Ensure a recommendation for a tape-drive backup or other appropriate backup media.

Network Considerations

Network sizing involves taking the following considerations into account (for network terminology see “Reference: Network Sizing Details” on page 102):

- Bandwidth requirements of the product
- Effect of latency and packet loss on performance and responsiveness

There are different types of network communication demand within QAD BI, and each area requires separate consideration from a sizing point of view. To this end we divide network sizing into the following areas:

- 1 Network considerations between the ERP database and BI database
- 2 Network considerations between the remote client and BI Portal
- 3 WAN Considerations for both 1 and 2 (for example, the impact of placing a WAN between the ERP database and the BI database server, and the impact of placing a WAN between a remote client and the BI Portal)

See “Reference: Network Sizing Details” on page 102 for details of estimating the impact of network considerations in each of these areas.

Network Recommendations

Please be aware of both sizing notes from this table and review “Reference: Network Sizing Details” on page 102 before finalizing network recommendations for the customer.

Network Bandwidth	LAN	WAN (Effect of Latency)	Network Card	Comment
ERP database to BI database	10+ MB/s (80 Mbps)	4 MB/s ¹ (32 Mbps)	Ensure at least 1 Gbps card	The WAN figure is a guideline figure and will depend on the actual latency involved (see “WAN Considerations” on page 104 for details).

Backup of BI database	1 Gbps	N/A	Ensure at least 1 Gbps card
Remote Client to BI Portal	80-140 kB/s (640 – 1120 kbps) per user	50+ kB/s ² (400+ kbps) per user	Ensure these rates are <i>minimum bandwidth per user</i> for each remote client to BI Portal user

1. Increasing latency (on WANs) has a very significant effect on slowing-down the transfer rate between the ERP database and the BI database. Very careful measurement should be made of the network quality (latency and packet loss, bandwidth) of any customer proposing a WAN based separation between the ERP database and the BI database servers (see “Reference: Network Sizing Details” on page 102 for details)
2. Testing has shown that client response times are expected to slow down by up to 100% across a WAN with a latency of 300-400 ms. Responsiveness will slow down across Wide Area Networks.

Example Sizing

We provide here an example of how to size a BI system.

Size a BI system with the following requirements:

- 20 expected concurrent BI Portal Users
- A 50 GB ERP source database
- Initially 3 BI Modules required

This example shows a single-tier deployment (one server to hold all BI components).

	Sizing Metric:	Recommendation	Note
A	CPU – DWD/Scheduler	2 cores	
B	CPU – Concurrent User Workload (Remote Portal Users)	20 concurrent users = 6 cores	
C	CPU – BI Portal Webapp	1 core	
D	CPU Total	6 + 1 = 7 cores	Max. (A,B) + C – Note that this is a <i>minimum requirement</i>
E	RAM - DWD/Scheduler	Assume “Medium” Load – 8+ GB	
F	RAM – BI Portal Webapp	2 GB	
G	RAM – Concurrent User Workload (Remote Portal Users)	Not relevant (as per “RAM Sizing for Concurrent Users (Interactive Portal Users)” on page 95)	
H	RAM - Total	8+ + 2 + 4 GB = 14 GB	As per “RAM Sizing for DWD/Scheduler Component ” on page 95, the 4 GB extra is for the OS
I	Disk Sizing	Initial approximate sizing needs to be carried out to create an <i>approximate</i> estimate of the initial DW size based on ERP table data. See “Reference: Estimation of Initial Database Size”.	As a note, 500 GB is a recommended minimum but all new implementations should obtain <i>disk sizing estimates</i> using the referenced guidelines.

	Sizing Metric:	Recommendation	Note
J	IO Subsystem	15 k rpm SAS disks RAID 10 on a SAN/NAS device Minimum of 6 disks (12 disks if using mirroring).	Again, depending on the estimated DW size, increase the number of disks in the RAID array.
K	Network Requirements	ERP-to-BI Server: Minimum of 3 MBps (Mega-bytes per second) bandwidth Client-to-BI Portal: Minimum of 100 kbps (kilo-bytes per second) bandwidth per user.	Ensure 1 Gbps network card minimum on DW server. For two-tier deployment also ensure at least a 1 Gbps network connections between the DW server and the BI Portal webapp server.

Reference: Estimation of Initial Database Size

If the BI data warehouse does not exist, the following can be used to provide a very approximate guideline as to the approximate initial expected BI database size.

Dependence of Initial Database Size on BI Modules

Note that BI v3.6+ is fully “modularized,” with three primary modules:

- Financials
- Order Management
- Operations

Sizing for the resultant BI Data Warehouse depends on many factors, especially the table size within the source ERP database. As different BI modules use only restricted source ERP tables, the final size of the BI Data Warehouse will depend sensitively on the modules used and also on the sub-modules used. This section describes how to create an estimate of the initial BI DW size based on the BI modules used.

Estimation of Initial BI Data Warehouse Size

The estimation of the initial data warehouse size involves a determination of the size of key ERP tables, and a subsequent calculation (which depends on the modularization) to convert these sizes into an estimated data warehouse size. Follow the calculation and description below to estimate the initial data warehouse size.

Note that:

- There is no guarantee that this estimate will give an accurate size for the initial data warehouse size, it is provided as an approximate method of generating an order-of-magnitude estimate on the initial data warehouse size.
- SQL-Server uses a large amount of extra disk space for temporary manipulation of transformational data and allowance should also be made for this. The estimate below only considers an approximation for the BI database, excluding requirements for SQL-Server temporary files.

Determine the size of all the following tables in the source QAD ERP database and use these in combination with the “Size Factor” to calculate an estimate of the initial data warehouse size.

Module	Table	Source	Size Factor**	Source ERP Table Size	(Size Factor) * (Source Size)
Financials	ap_mstr	SE	1		
	ar_mstr	SE	2/3*		
	ard_det	SE	1		
	ck_mstr	SE	1		
	ckd_det	SE	1		
	gltr_hist	SE	1		
	tr_hist	SE + EE	1/10*		
	ee_CInvoice	EE	1		
	ee_CInvoiceMovement	EE	1		
	ee_DInvoice	EE	1		
	ee_DInvoiceMovement	EE	1		
	ee_Posting	EE	1		
	ee_PostingHist	EE	1		
	ee_PostingLine	EE	1		
Order Management	so_mstr	SE + EE	1		
	sod_det	SE + EE	1		
	idh_hist	SE + EE	1		
	ih_hist	SE + EE	1		
	ar_mstr	SE + EE	1/3*		
	tr_hist	SE+EE	1/10*		
Operations (assuming all sub modules: Purchasing, Inventory, OEE)	wo_mstr	SE+EE	1		
	wod_det	SE+EE	1		
	tr_hist	SE+EE	4/5*		
	op_hist	SE+EE	1		
	po_mstr	SE + EE	1		
	pod_det	SE + EE	1		
	prh_hist	SE + EE	1		
Total Source Size for Conversion					X
Estimated Initial BI DW Size					X*(100)

Estimated Size of Initial BI Database = (X)*100

You should allow for this amount of space at a minimum. Further, if this total comes to under 500 GB it is recommended that you set up at least a 500 GB disk allowance.

*** TABLE NOTE 1:** A “size factor” of less than one indicates that not all transactions in the specified table will be used by the relevant BI module, and this needs to be taken into account in size estimations. For example, in the BI Financials Module, the ar_mstr table has a size factor of 2/3, indicating that only two-thirds of all transactions on the ar_mstr table will be converted into aggregated BI data in the BI Data Warehouse.

**** TABLE NOTE 2:** The source tables might contain data for a large number of years (n years, say $n=10$ years). If the data warehouse is only going to be populated with for example only 3 years worth of data ($y=3$), then an additional size factor of y/n (in this case $3/10$) should be applied to the relevant source ERP tables. And this extra size factor should be included in the table above and in the calculations to determine the estimated initial size of the data warehouse.

Allowance for Future Database Growth

To accommodate the growth of the data warehouse over time, use the following guidelines.

In general, if the initial data warehouse size is based on transactions over the last 3 years, expect a data warehouse size growth over the coming 3 years to be of the same size as the initial data warehouse size. If the expected transaction volumes for the customer will increase, then you should make extra allowance.

Example

If the initial data warehouse is created based on the last 3 years of activity, and the initial size is 250 GB, then expect the data warehouse size to at least double over the coming 3 years.

Hence, estimate of Disk Space Allocation to allow for medium-term data warehouse growth = $2 * (\text{initial database size estimate}) = 2 * (X * 100)$.

Reference: Network Sizing Details

Terminology

Bandwidth – Is a measure of the data transfer rate. Usually measured in “bits” per second (bps) or “kilo-bits” per second (Kbps) or “Bytes” per second (Bps) etc. Each physical network connection between two points will have a maximum bandwidth rating which can not be exceeded.

Latency – Is a measure of the “time delay” of transferring data from one point on a network to another point on the network. It is usually measures in milliseconds (ms) or seconds (s).

WAN – Anagram for “Wide Area Network”. A network where there is a large distance (or latency) between computers on the network. This is in contrast to a LAN (Local Area Network) where there all machines on the network are “local” in the sense that there is minimal latency between the network points.

The following sections discuss the impact of bandwidth, latency and WAN on BI components.

Network Sizing Considerations - Database to BI Database Server

The communication between the BI database server and the ERP database is required during the nightly BI daily load.

The BI DWD Scheduler communicates requests to the ERP database server via an ODBC connection, and in return the ERP database server returns all required data to the BI database server. Outside of BI daily load times there should be no communication between the two servers (unless the daily load fails, in which case the administrator or business may require the load to be re-run at a different time).

- Bandwidth requirements, why this is important, and how to address this by sizing properly.
- Effects of latency/packet loss (WAN), why this is important, and how to address this by sizing appropriately.

Importance of Latency and Bandwidth

Bandwidth considerations are important for this aspect of BI for the following reasons:

- As a large volume of data may be transferred between the ERP system and the BI server (100's of MBs, multiple GBs, depending on database size, transaction volumes, etc.), bandwidth must be suitably sized so as not to act as a bottleneck to ETL processing.
- The BI daily load must complete within a limited business-imposed daily time-window. The transfer rate of large data transfers can be limited by both bandwidth and latency. For this reason, sizing and allocating appropriate bandwidth is vitally important.

Bandwidth Requirements

We quote here the peak bandwidth requirements, note that this is not a sustained rate. The sustained bandwidth transfer rate will be much lower than the peak rate.

- **LAN Guideline:** No more than 10 MB/s (80 Mb/s). Note that from current internal testing (as of November 2011) we see LAN transfer rates of no more than 8 MB/s (3 MB/s normally) with a setting of 10,000 for fetcharraysize.
- **WAN Guidelines:** Under 5 MB/s. We see degradation to:
 - 0.4 MB/s at 200 ms latency (peak value)
 - 0.4 MB/s at 400 ms latency (peak value)

Impact of Latency on Bandwidth Requirements

For example, if we take a customer with 1 GB of data to transfer then we can estimate the minimum amount of time for that data to transfer from the ERP system to the BI database server:

LAN: $1000 \text{ MB} / (3 \text{ MB/s}) = 333 \text{ seconds} = 5 \text{ mins } 33 \text{ seconds}$

WAN, 200 ms: $1000 \text{ MB} / (0.4 \text{ MB/s}) = 2,500 \text{ seconds} = 41 \text{ mins } 36 \text{ seconds}$

In reality the peak rates are sustained only very briefly and average transfer rates are much lower – some average transfer rates may be 10 times lower. These figures are only provided here as an indication of how bandwidth and latency can influence data transfer times, and how these effects can impact daily load times.

Network Sizing Considerations - Remote Client to BI Portal

The remote client issues requests to the BI Portal to retrieve data and subsequently processed data is transferred from the Portal server to the client machine. This type of usage exhibits a “spiky” network pattern where periods of data retrieval may require 100's of kB/s transfer rates separated by periods of little network activity.

Because the network access pattern is “spiky,” specifying an “average bandwidth” requirement per user can be slightly misleading.

Recommendation

100 kB/s bandwidth minimum should be reserved per user, bearing in mind the caveats below.

- Caveat 1 – Dashboard loading: The initial dashboard load may result in most network traffic when the dashboard data is returned from the server. This can result in brief periods of up to 300+ kB/s bandwidth demand.
- Caveat 2 – Report Usage: Depending on the size of the reports and frequency at which reports are run, heavy report usage can result in increased bandwidth demand.

WAN Considerations

Latency is an unavoidable consequence of Wide Area Networks (WANs).

On WANs, latency will have the effect of:

- Extending the data transfer times from the ERP system to the BI database system.
- Increasing the average client response times above the expected response times for LAN (Local Area Network) based client-server communication.

Latency will result in a decreased network transfer rate, thereby extending communication time, and decreasing bandwidth requirements, at the cost of overall responsiveness.

Customization Guidelines

This chapter describes QAD-recommended best practices for QAD Business Intelligence customizations. Topics include:

Overview 106

Customization Challenges 106

QAD BI 3.x Customization Methodologies 109

Customization Needs for Additional Columns 111

Identifying Every Appearance of a Column in Existing Code 118

Naming Conventions for Custom Changes to Packaged QAD BI Modules 118

Guidelines for Creating Custom Path Custom Tables 119

Adding Additional Columns to a Load Table 120

Adding Additional Custom Columns to Pre-perm Stage Tables 123

Adding Additional Custom Columns to perm/dim/fact Tables 124

Adding Custom Path stage/extract Tables After the perm Tables 127

Additional Customization Topics 131

Overview

Most customers that purchase the QAD BI modules will invariably want additional information that falls outside of the baseline code of the warehouses. This comprehensive guide to QAD-recommended best practices for BI 3.x customizations should help you get the additional information you need. The QAD BI Data Warehouse Designer tool (DWD) makes it easy to add in additional columns and tables to existing BI modules. Doing so, however, can create difficulties at later points when a bug patch or upgrade is required for tables or modules that have been customized. It is important that any customizations are created in a way that avoids having to redo them if patches or upgrades are applied.

Customization Challenges

The DWD tool makes the addition of additional columns or tables to a data warehouse a fairly straightforward exercise, and it is one of the primary selling points of its creator, Wherescape. One of the great challenges in data warehousing is that when a data warehouse is built, it is difficult to make changes to it. Wherescape's approach to this eliminates much of the difficulty with getting these changes right, but the original intent of the tool is for a single customer to be able to make changes to their data warehouse. As QAD is building out its BI solution to package warehouses into different modules that will be implemented for many different customers, it is a much more challenging environment to release bug patches and upgrade code: Each customer may have a different set of customization,s and installing patches or upgrades to tables and procedures that have customizations will potentially wipe out the customization work and require it to be done again.

As an example, look at a customized dim_item table that then has a patch (without the customizations) applied to it.

Here is the layout for the dim_item table from the 3.4.1 release. The last several columns are circled for reference purposes.



dim_due_date	promotion_group	Promotion Group	varchar(80)	stage_item	promotion_group	Y	N	N	Y	Y
dim_effective_date	current_cost	Current Cost T...	numeric(38,10)	stage_item	current_cost	###.00;...	Y	Y	Y	N
dim_end_of_month_date	standard_cost	Standard Cost ...	numeric(38,10)	stage_item	standard_cost	###.00;...	Y	Y	Y	N
dim_entered_date	latest_standard_cost	Latest Standar...	numeric(38,10)	stage_item	latest_standard_cost	###.00;...	Y	Y	N	N
dim_entity	net_weight	Net Weight	numeric(38,10)	stage_item	net_weight	###0	Y	Y	Y	N
dim_exchange_rate	net_weight_um	Net Weight Unit...	varchar(30)	stage_item	net_weight_um		Y	N	N	Y
dim_expected_pay_date	latest_current_cost	Latest Current ...	numeric(38,10)	stage_item	latest_current_cost	###.00;...	Y	Y	N	N
dim_gl_transaction_type	dss_extract_timestamp	Extract Timesta...	datetime	stage_item	dss_extract_timest...	MM/dd/yyyy...	Y	N	N	Y
dim_invoice_date	dss_process_batch_id	Process Batch Id	numeric(18)	stage_item	dss_process_batch_id	###; -#...	Y	Y	Y	N
dim_item	dss_update_time	Update Time	datetime	stage_item	dss_update_time	MM/dd/yyyy...	Y	N	N	Y
dim_last_due_date										

Here is the layout for the dim_item table for a customer that has a 3.3 customized version of the modules that not only has the easily identifiable custom columns that begin with an x_ prefix (QAD's preferred prefix for custom columns now is xx_), but also note that they have some other columns that do not match the 3.4.1 dim_item table. These had been added at time of installation with the belief that these same columns would be added to the code base. That ended up not yet happening, so an upgrade to 3.4.1 is likely to require even more work and investigation. Unless columns are already in the code base, treat any additions as custom.

dim_discount_sub_account	configuration	Configuration T...	varchar(30)	stage_item	configuration	
dim_draft_discount_date	promotion_group	Promotion Group	varchar(80)	stage_item	promotion_group	
dim_draft_submission_date	current_cost	Current Cost T...	numeric(38,10)	stage_item	current_cost	###0
dim_due_date	standard_cost	Standard Cost ...	numeric(38,10)	stage_item	standard_cost	###0
dim_effective_date	latest_standard_cost	Latest Standar...	numeric(38,10)	stage_item	latest_standard_cost	###0
dim_end_of_month_date	latest_current_cost	Latest Current ...	numeric(38,10)	stage_item	latest_current_cost	###0
dim_entered_date	promotion_group_desc...	Promotion Grou...	varchar(80)	stage_item	promotion_group_d...	
dim_entity	supplier_description	Supplier Descrip...	varchar(30)	stage_item	supplier_description	
dim_exchange_rate	buyer_description	Buyer Description	varchar(80)	stage_item	buyer_description	
dim_expected_pay_date	abc_class	ABC Class	varchar(1)	stage_item	x_abc_class	
dim_gl_transaction_type	safety_time	safety time	numeric(20)	stage_item	safety_time	###0
dim_invoice_date	dss_extract_timestamp	extract timestamp	datetime	stage_item	dss_extract_timest...	
dim_item	dss_process_batch_id	process batch id	numeric(18)	stage_item	dss_process_batch_id	###0
dim_last_due_date	dss_update_time	dss update time	datetime	stage_item	dss_update_time	
dim_last_receipt_date	x_reporting_unit_of_mea...	reporting unit o...	varchar(64)	stage_item	x_reporting_unit_of...	
dim_layer	x_ship_day_tolerance	ship day tolerance	integer	stage_item	x_ship_day_tolerance	###0
dim_location	x_ship_quantity_tolerance	ship quantity to...	decimal(13,2)	stage_item	x_ship_quantity_tol...	###0
dim_order_date						

In SQL Server you can see that the table matches the meta-data observed via the DWD.

configuration (varchar(30), null)
promotion_group (varchar(80), null)
current_cost (numeric(38,10), null)
standard_cost (numeric(38,10), null)
latest_standard_cost (numeric(38,10), null)
latest_current_cost (numeric(38,10), null)
promotion_group_description (varchar(80), null)
supplier_description (varchar(30), null)
buyer_description (varchar(80), null)
abc_class (varchar(1), null)
safety_time (numeric(20,0), null)
dss_extract_timestamp (datetime, null)
dss_process_batch_id (numeric(18,0), null)
dss_update_time (datetime, null)
x_reporting_unit_of_measure (varchar(64), null)
x_ship_day_tolerance (int, null)
x_ship_quantity_tolerance (decimal(13,2), null)

And if we open the affiliated procedure, in the DWD, we can see what columns are being inserted into and that they match.

current_cost	current_cost
0	standard_cost
latest_standard_cost	latest_standard_cost
latest_current_cost	latest_current_cost
promotion_group_description	promotion_group_description
supplier_description	supplier_description
buyer_description	buyer_description
x_abc_class	abc_class
safety_time	safety_time
dss_extract_timestamp	dss_extract_timestamp
dss_process_batch_id	dss_process_batch_id
x_reporting_unit_of_measure	x_reporting_unit_of_measure
x_ship_day_tolerance	x_ship_day_tolerance
x_ship_quantity_tolerance	x_ship_quantity_tolerance

```

FROM stage_item
WHERE NOT EXISTS (
  SELECT 1 FROM dim_item
  WHERE ( stage_item.source_system_code = dim_item.source_system_code
    OR ( stage_item.source_system_code IS NULL AND dim_item.source_system_code IS NULL ))
  AND ( stage_item.domain_code = dim_item.domain_code
    OR ( stage_item.domain_code IS NULL AND dim_item.domain_code IS NULL ))
)

```

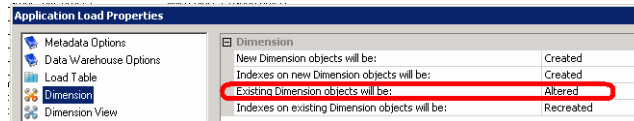
A patch is then applied via the Administrator tool to fix some issue with the table or to upgrade the table to the 3.4.1 version of the code. You can see that it initially says that existing objects will be replaced.

Object	Status
Application load begins	D:\kurt\work\custom_testslapp_obj_Patch_Test_1.wst
set_dim_item_key	Will replace existing object
update_dim_item	Will replace existing object
update_stage_item	Will replace existing object
update_stage_item_site	Will replace existing object
update_stage_order3	Will replace existing object
update_stage_order2	Will replace existing object
update_stage_order1	Will replace existing object
custom_stage_order1	Will replace existing object
update_fact_order_history	Will replace existing object
custom_fact_order_history	Will replace existing object
dim_item	Will replace existing object
stage_item	Will replace existing object
stage_item_site	Will replace existing object
stage_order3	Will replace existing object
stage_order2	Will replace existing object
stage_order1	Will replace existing object
fact_order_history	Will replace existing object
dim_item_idx_0	Will replace existing object
dim_item_idx_A	Will replace existing object
stage_order1_idx_A	Will replace existing object
fact_order_history_idx_1	Will replace existing object
fact_order_history_idx_1n	Will replace existing object

Setup Administrator

47 Object/s will be replaced. You may cancel at this point and review the objects to be replaced and/or added.

However, the default setting for each object during application installation ensures that the dim, perm and fact tables are only altered by changes and the updates proceed successfully.



A refresh of the DWD Browser pane to review the changes to dim_item then reveals that the data in the tables looks exactly like the 3.4.1 version of dim_item. Any other columns that had existed for this table in the DWD's meta-data are no longer shown.

dim_due_date	promotion_group	Promotion Group	varchar(80)	stage_item	promotion_group	Y	N	N	Y	Y	
dim_effective_date	current_cost	Current Cost T...	numeric(38,10)	stage_item	current_cost	###.00;...	Y	Y	N	Y	
dim_end_of_month_date	standard_cost	Standard Cost ...	numeric(38,10)	stage_item		###.00;...	Y	Y	N	Y	
dim_entered_date	latest_standard_cost	Latest Standar...	numeric(38,10)	stage_item	latest_standard_cost	###.00;...	Y	Y	N	Y	
dim_entity	net_weight	Net Weight	numeric(38,10)	stage_item	net_weight	##0	Y	Y	N	Y	
dim_exchange_rate	net_weight_um	Net Weight Unit...	varchar(30)	stage_item	net_weight_um		Y	N	N	Y	
dim_expected_pay_date	latest_current_cost	Latest Current ...	numeric(38,10)	stage_item	latest_current_cost	###.00;...	Y	Y	N	Y	
dim_gl_transaction_type	dss_extract_timestamp	Extract Timesta...	datetime	stage_item	dss_extract_timest...	MM/dd/yyyy...	Y	N	N	Y	
dim_invoice_date	dss_process_batch_id	Process Batch Id	numeric(18)	stage_item	dss_process_batch_id	###;-#...	Y	Y	Y	N	Y
dim_item	dss_update_time	Update Time	datetime		dss_update_time	MM/dd/yyyy...	Y	N	N	Y	N
dim_last due date											

When the patch was applied, we had selected to not change existing tables, only alter them. Why are the other columns gone? Going to SQL Server reveals that they still exist in the database and that any new columns that were found when the patch was applied were placed at the end of the table column list. In this case, net_weight and net_weight_um were new columns in the 3.4.1 table.

```

configuration (nvarchar(30), null)
promotion_group (nvarchar(80), null)
current_cost (numeric(38,10), null)
standard_cost (numeric(38,10), null)
latest_standard_cost (numeric(38,10), null)
latest_current_cost (numeric(38,10), null)
promotion_group_description (varchar(80), null)
supplier_description (varchar(30), null)
buyer_description (varchar(80), null)
abc_class (varchar(1), null)
safety_time (numeric(20,0), null)
dss_extract_timestamp (datetime, null)
dss_process_batch_id (numeric(18,0), null)
dss_update_time (datetime, null)
x_reporting_unit_of_measure (varchar(64), null)
x_ship_day_tolerance (int, null)
x_ship_quantity_tolerance (decimal(13,2), null)
net_weight (numeric(38,10), null)
net_weight_um (nvarchar(30), null)

```

Reviewing the affiliated procedure in the DWD shows that all the logic referencing the custom columns has been removed and the new logic only reflects the columns that exist in the 3.4.1 version that was applied.

```

configuration      promotion_group
promotion_group    current_cost
current_cost       latest_standard_cost
latest_standard_cost
net_weight          net_weight_um
net_weight_um      latest_current_cost
latest_current_cost
dss_extract_timestamp
dss_process_batch_id
FROM
stage_item
WHERE NOT EXISTS (
SELECT 1 FROM dim_item
WHERE ( stage_item.source_system_code = dim_item.source_system_code
OR ( stage_item.source_system_code IS NULL AND dim_item.source_system_code IS NULL ))
AND ( stage_item.domain_code = dim_item.domain_code
...OR ( stage_item.domain_code IS NULL AND dim_item.domain_code IS NULL ))

```

Note Since the new columns were added to the table and the table was not dropped, the custom columns still would contain their data, but would no longer be updated by the new procedure.

As you can see, implementing patches and upgrades to tables that have any custom work done to them is not a straightforward process and likely requires a fair amount of followup work to get the customizations back in place in the DWD meta-data (the underlying SQL table can remain

unchanged). It is for this reason that **ideally** custom work will not be done directly on standard release tables in the data warehouse or at the very least, the footprint of the customized work will mostly fall outside of the baseline code.

QAD BI 3.x Customization Methodologies

With the potential problems affiliated with adding custom columns to our existing standard infrastructure, it is clear that whenever possible, an alternate way of doing custom work needs to be utilized to ensure easy patch and upgrade installations.

Custom Path Methodology (Preferred)

To avoid the pitfalls of making changes to our existing baseline tables at customer sites, it would be much simpler if the custom requirements could be met without ever touching the existing infrastructure. This is nearly a possibility! The approach to resolve this issue is to create a custom path for the custom work that has its own separate objects and is integrated into the baseline fact/dim by a custom procedure that updates the custom columns in the dim/fact table. This task is run after the regular tasks for populating/updating that table have completed.

Integration Points

The customized objects are integrated with the baseline objects using update procedures. If this option is chosen then the custom column will need to be added back into the DWD (but not the actual database), any time a patch impacts the baseline objects. If cubes are used and the new customization desires them, the additional customization should be added as a new attributes to the existing OLAP dimension.

Note The grain of the update procedure should be the same as the core dimension table.

Processing Flow

Fig. B.1
DWD Processing Architecture with Customized Objects and Update Procedure for a Dimension

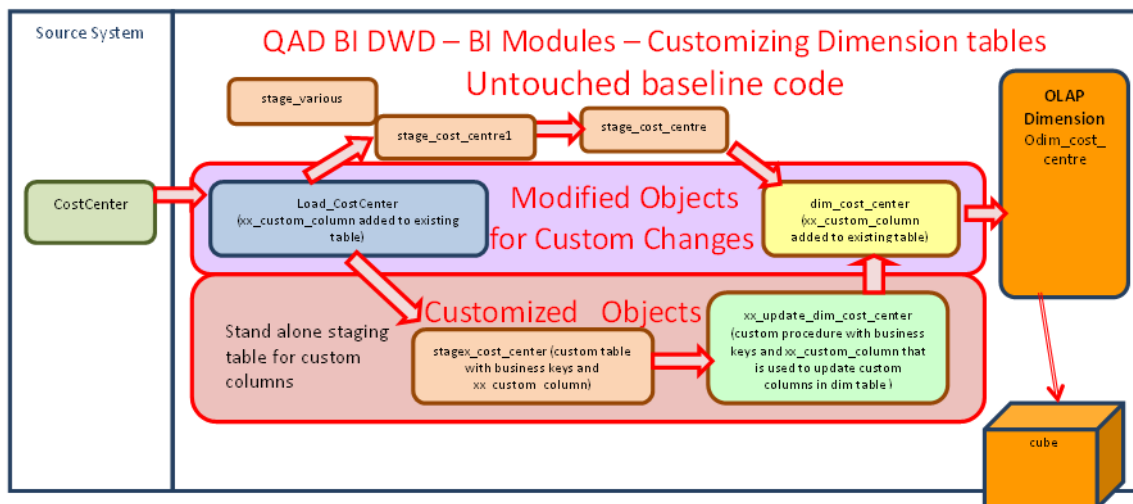


Fig. B.2
DWD Processing Architecture with Customized Objects and Update Procedure for a Fact (not History or Rollup)

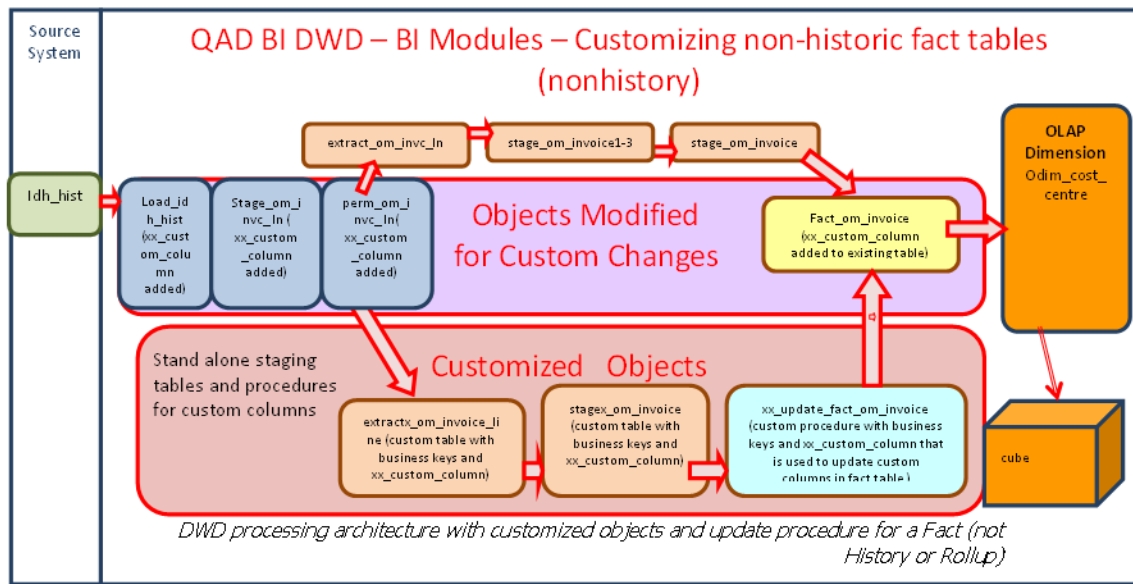
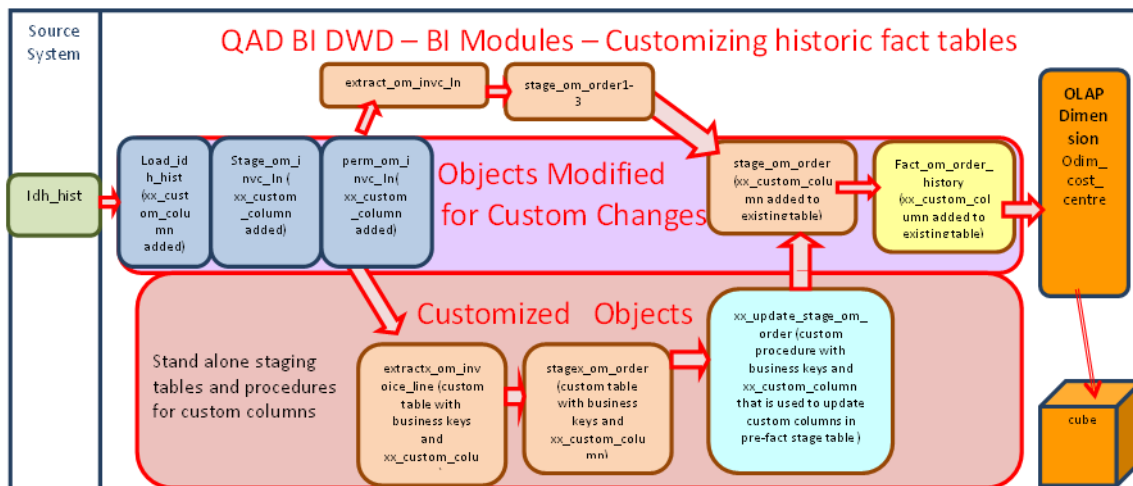


Fig. B.3
DWD Processing Architecture with Customized Objects and Update Procedure for a Fact (History)



Pros and Cons

Pros

- Any existing reports would continue to work as the dimension only has additional fields added.
- Separation of baseline and customizations tables and stored procedures is maintained throughout the processing, making upgrades much easier.
- There is a fairly low risk of data loss with this method of integration because the core dimension and fact tables only have extra columns added to them by the customizations that are populated by stored procedure. In the event of a patch or upgrade, the columns need to be restored in the Designer, and the custom column update procedure re-added to any jobs it may have been lost from.

Cons

- This method of customization requires custom columns be added directly to the dim/fact table—custom columns would need to be reapplied in the DWD only after an application upgrade or patch so they would still be visible to end users.
- Performance time-wise could be impacted by having this parallel path and by running a separate procedure to update dim or fact tables that were just populated by their normal insert/update processes.
- There is more up-front work in creating the parallel table path (instead of just adding columns to existing infrastructure).

Integrated Methodology (Last Resort)

There are times when the sheer complexity of the staging and transformation processes makes the use of a parallel table path too difficult to set up. This is particularly true for some fact tables, in particular those that do a lot of ranking, like the `fact_om_order_history` table. If an attempt has been made to create a parallel path and it is just too difficult to achieve a satisfactory result, adding custom columns to the existing infrastructure can be allowed. Future patches and upgrade installations will have to take these into account and they will have to be reimplemented when they get disrupted. The customer will need to be informed of this possibility as it may require an additional services engagement to reimplement the customizations.

Pros and Cons

Pros

- Easier to do an initial implementation.
- Ensures the customization follows the same path that like data is moving through, eliminating risk of somehow getting the data connections wrong.
- No need for keeping separate sets of views or staging tables.

Cons

- Far more difficult to do upgrades or install patches as customization work needs to be reimplemented after the patch is applied. This is the number-one reason why this methodology is a last resort.
- Creates difficulties for support in trying to determine how to resolve issues with the custom code.
- There is some risk of disrupting existing logic if custom work is not done right. The implementer needs to be very careful and aware of Cartesian joins.

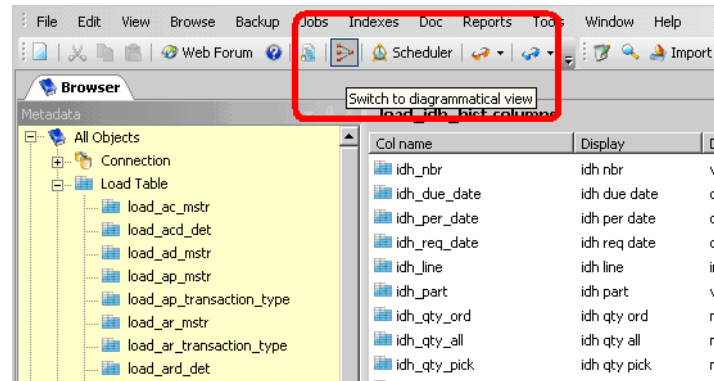
Customization Needs for Additional Columns

When data from a QAD Progress table is identified as important to a customer and does not exist in one of the BI fact tables, it is important to check first to see if the data is already being pulled over into one of the pre-staging tables as well as if the source table is already being referenced.

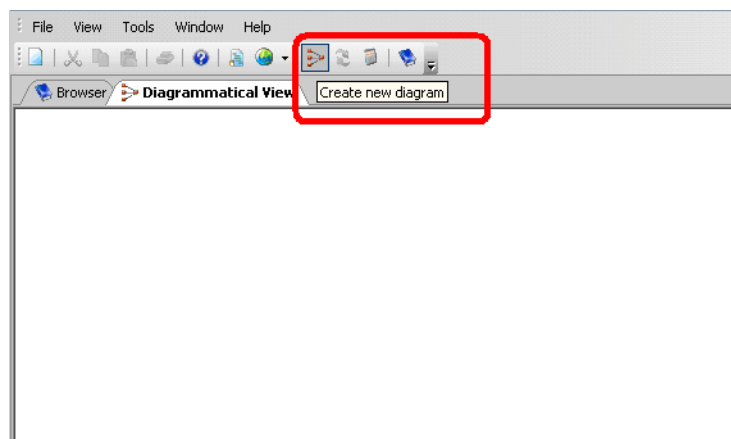
Use Track Back Diagram to Check Tables

First do a track back diagram of the fact_ or dim_ table that the custom column is to be added to.

Click the Diagram icon to the left of the Scheduler icon in the QAD BI Data Warehouse Developer (DWD).

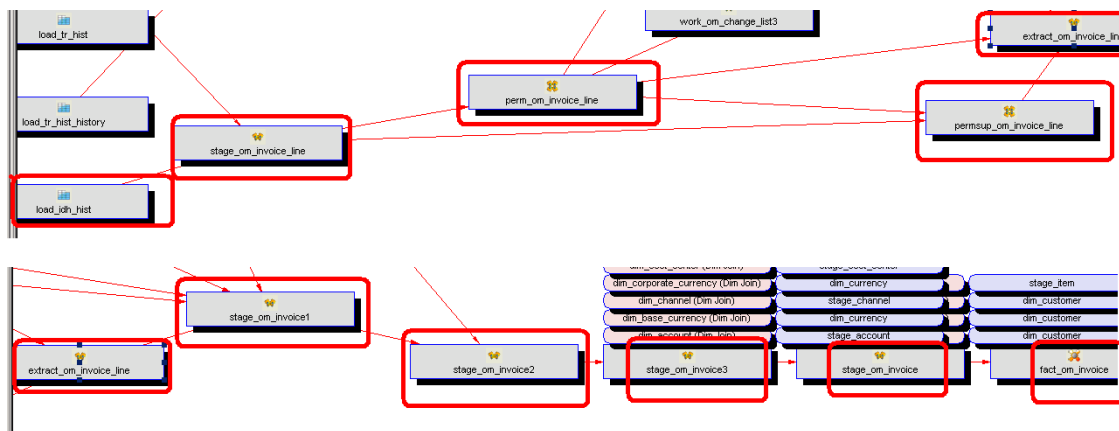


If the Diagrammatical View tab was not already open, you have to click the icon a second time.



Pick the fact table you want to track back on and click the OK button. You can start to type in the name of the table to find it in the drop-down selection list faster, but if you type the whole name in, you still have to select the item from the drop-down list otherwise nothing will appear when you hit OK.

For the purpose of this example, we will use the fact_om_invoice table and look to add a custom column pulling data idh_propline from the idh_hist table.



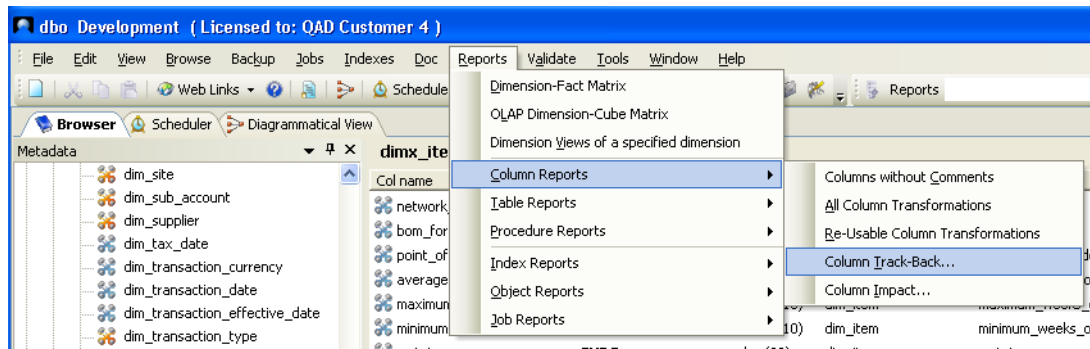
The list should contain:

- load_idh_hist
- stage_om_invoice_line
- perm_om_invoice_line
- permsup_om_invoice_line. The permsup_om_invoice_line table is used to store parallel date data about the perm_om_invoice_line table based on the primary business key. No custom work should be necessary here unless the business key is changing for some reason.
- work_om_change_list1-3 and hist. The work tables that you can see in the first of the two dataflow diagrams above are for determining new data based on dates, so they were not included in the list to be modified.
- extract_om_invoice_line
- stage_om_invoice1
- stage_om_invoice2
- stage_om_invoice3
- stage_om_invoice
- fact_om_invoice

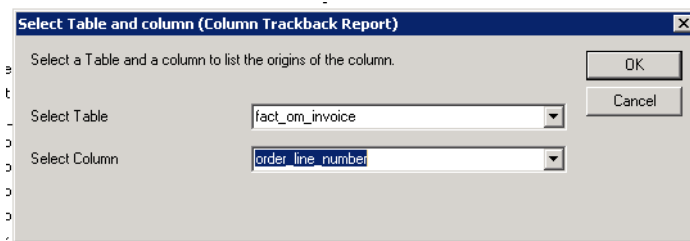
As a matter of practice, it is helpful to type this list into notepad or some other text editor for reference purposes as you work through the customization.

Double Checking the Data Path with a Column Track Back Report

To be extra sure your table list is complete, you can do a track back on a column name from the fact table if you know that column to be from the same source table as the one that the custom column would be added from. Go back to the Browser tab, select Reports|Columns|Column Track-back. (Prior to the 6.5.1 version of the DWD this option was second from the bottom in the reports list as Track back on specific column usage.)

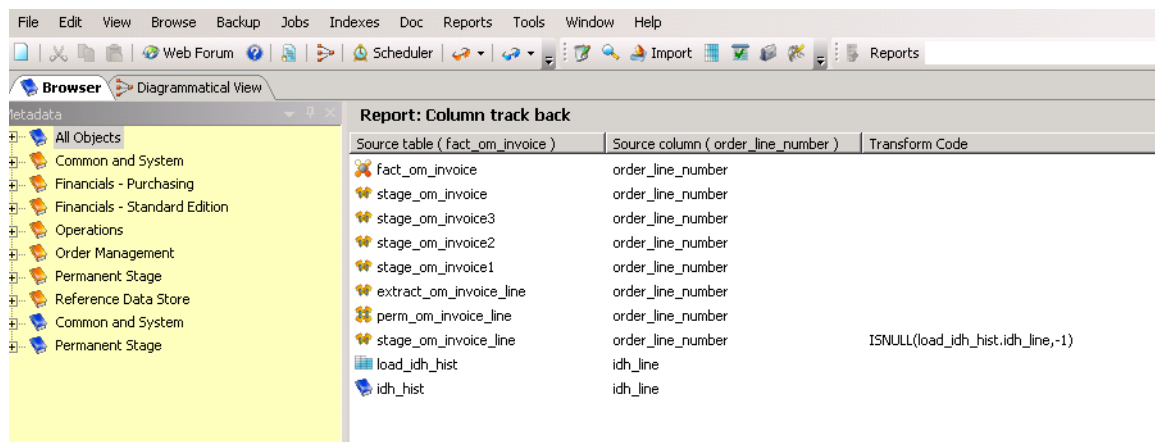


You will be prompted with a box. Select the name of the fact_ table and a column you know comes from the source table in question. In this instance, we know that order_line_number comes from the idh_mstr table.



Once picked, click OK. A report will be generated tracking back the column from the Select Table back to its source.

Note For some dimension values in dim_ or fact_ tables, this report is more effective starting prior to the staging table that the dim_ value keys was defined in.



Aside from showing you all the tables that lead to the fact table, you also see the column names in those tables and any transform code that occurs for a column. This table list closely matches the list we wrote down from the Diagramatic View.

We have identified that the QAD table the desired extra data is sourced from as already existing in the dataflow as a load_ table. We now need to see if the column in question also exists in that load_ table. In the DWD go to Browser|All Objects|Load Table, select the load_ table, and see if the needed column is in the list. You can find columns more quickly and effectively by clicking on the Col name. In this case, idh_procline is not included in the load_idh_hist table.

Col name	Display	Data type
idh_list_pr	idh list pr	numeric(38,
idh_loc	idh loc	nvarchar(80
idh_lot	idh lot	nvarchar(80
idh_nbr	idh nbr	nvarchar(80
idh_part	idh part	nvarchar(30
idh_per_date	idh per date	datetime
idh_pickdate	idh pickdate	datetime
idh_price	idh price	numeric(38,
idh_pricing_dt	idh pricing dt	datetime
idh_project	idh project	nvarchar(80
idh_promise_date	idh promise date	datetime
idh_qty_all	idh qty all	numeric(38,
idh_qty_chg	idh qty chg	numeric(38,
idh_qty_cons	idh qty cons	numeric(38,
idh_qty_exch	idh qty exch	numeric(38,
idh_qty_inv	idh qty inv	numeric(38,
idh_qty_ord	idh qty ord	numeric(38,
idh_qty_pend	idh qty pend	numeric(38,

If it did exist in the column list, the next step would be to check to see if it exists in the stage_ table. If it does not exist, review the steps in this process before proceeding.

Go to the Stage Table section of the Browser and find the staging table that was second in the list of tables above, in this instance, stage_om_invoice_line. When you click on the table name, look for the source column that is to be added in the Src column list. You can click on the header to sort alphabetically if the list is long and it is not immediately obvious if the column is there or not. In most instances, the column name in this first staging table is changed to something with a longer name than the source column name. Keep going through this process of tracking the soon-to-be-added custom column through the stage_, perm_, extract_ and more stage_ until you find a table in the path that does not contain it.

Notes on Tables

A few things to note about the tables:

- load_ tables

Load_ tables contain a straight extract from their source system. The majority of the BI extracts come directly from the QAD ERP system's Progress databases. The only change or transformation that should (there are some exceptions to this rule) occur for these columns is to clean up any blank spaces that may have been inadvertently left in the Progress table or to remove the '|' symbol from the data as it will disrupt the data load. The transformation looks like this:

```
REPLACE(REPLACE(REPLACE(idh_hist.idh_type,'|',' '),CHR(10),' '),CHR(13),' ')
```

This logic should be used for any source table character (char,varchar) type column that gets added to the load table.

- stage_ tables

Stage tables are used to do a variety of things that transform the data in some way. The stage tables that come right after load tables are typically used to just rename the column names into something more readable as they pass the data up to the perm or dim tables or other staging tables. Any later in the flow, stage tables are doing something with the data to make it more usable.

- perm_ tables

In many instances, there will also be perm tables that hold data that is permanently maintained in the source system and would put a heavy load on the system to be extracted each run. The perm has one stage table prior to it where the column names change from the load table and no transformations should occur prior to a perm table. The data in the perm table should essentially represent a like image of what is contained in its source system. Perm tables are one of the three types of tables with which great care must be taken to not inadvertently drop or delete data.

- work_ tables

These tables are a type of stage_ table used to check the perm tables to see if any data has changed as per the date requirements set for that work table. This might mean that the work table is looking for any records that have been updated in the perm tables in the last 10 days. For a historic load, all the records are pulled from the perm table. The determination of what date values to use is usually derived from parameters (such as PO_HISTORY_DATE, the day to start looking back from and PO_PROCESS_DAYS, the number of days that should be looked back from PO_HISTORY_DATE).

Custom paths may reference the existing work_ table infrastructure.

- extract_ tables

Perm_ tables should have at least one and in some cases multiple extract tables. These tables use the records as determined by the work_ tables to extract those records from the correlating perm_ table and prepare them to be picked up by the staging tables and fed on to fact_ tables of that particular module's Job.

A custom path referencing a perm_ table will need its own extractx_ table which will contain the business key and the required additional columns.

- dim_ tables

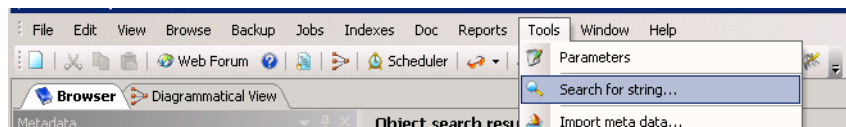
Where non-calculated potentially redundant information is stored in the data warehouse and the points of a star schema. It is very critical that dim_ (or dimension) tables are not accidentally dropped or data deleted from them, as they hold the keys that the fact uses to link to them and there is no guarantee that a dim_ table will create keys in the same sequence if it needs to be repopulated. Dropping and re-creating an existing dimension from a populated warehouse will likely force you to have to repopulate all the fact tables that reference that dim_ table.

- fact_ tables

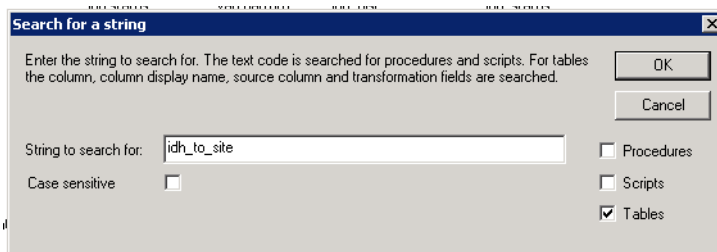
Where the calculated values of the data warehouse are stored and the center of a star schema. It is very important that fact_ table data is not inadvertently deleted or the table re-created, but recovering from accidentally doing so is not as painful as it is for perm_ or dim_ tables.

Identifying Every Appearance of a Column in Existing Code

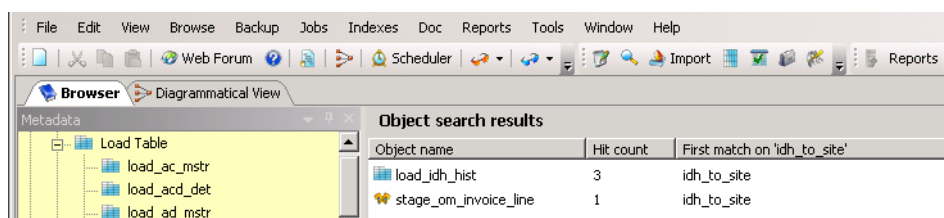
One last shortcut in the column search process: If you believe that the column is somewhere in the warehouse and it didn't show up through the prior methods, you can also do a search for the character string in the code.



Depending on what you are looking for, pick the relevant boxes; in this case, only Table would be the best choice.



Results are displayed as follows.



Naming Conventions for Custom Changes to Packaged QAD BI Modules

It is important that any custom work is always named per our defined standards so that anybody looking at code (support, services, customers) can feel confident that custom work has been applied in a uniform way and identifying custom work should be straightforward. We have three different categories of naming conventions, which are outlined below.

Custom Tables

Any new tables that need to be created should have an x affixed to the first part of their name before the first underscore:

loadx_, stagex_, permx_, extractx_, factx_, dimx_, olapx_

By naming them this way, the custom tables can easily be distinguished from the regular tables in the modules and readily found. They will show up at the bottom of the list for each type in the Browser pane in the DWD tool.

Custom Columns Joined to Standard Module

Any new columns added to any of the existing tables or new parallel path custom tables that will eventually join with existing tables should be prefixed with an xx_ to identify them as such. For example, if you were to add a column called recipient_date to an existing table in the standard release of the module, it should be named xx_recipient_date. The reason for using xx_ is that the

underline acts like wild card in the DWD's Search string feature, so a search for any columns that contain x_ will return any value that contains an x (such as tax_code). By using xx_, searches can be narrowed dramatically to only columns that contain xx—which should be nearly all customizations.

Custom Columns Not Joined to the Baseline Module Tables

Any columns in new custom tables that are not going to end up being joined to existing dim_ or fact_ tables as part of a view or similar methodology can forgo any prefixing of the column names. The table that they are in, however, should be properly prefixed with an xx as explained above. (Putting in an xx_ prefix for these does ensure that we can easily find all custom columns in the system though.)

Guidelines for Creating Custom Path Custom Tables

This section describes QAD's preferred and recommended custom methodology.

To avoid difficult and challenging upgrades and patch installs, custom work should ideally reside in its own set of parallel tables that closely mimic the tables and logic that we are already using to populate our dim_ and fact_ tables. If a data path has been identified as already existing from the same source as the requested custom column, and that path does not appear overly complex, it is a simple matter to create a series of tables that follow that path in parallel to get our data to its ideal destination, a view created from the existing dim or fact and the dimx or factx. Even under this methodology, there are exceptions to the rules.

Load Tables

Despite the ideal design of having custom path tables to avoid problems with upgrades, for load tables, if the source of the data for the new custom column is coming from a source that is already being accessed by a load table in the data warehouse, it would be redundant to repeat the action of loading a second custom table from the same source for just one new column. So in an example as provided earlier where idh_procline needs to be added to the fact_om_invoice table, it should be added as a custom column xx_idh_procline to the existing load_idh_hist table. In cases where the required new columns from source tables that are not being utilized by the existing warehouse, a whole new loadx_ table should be created (and should contain all relevant columns for linking that data to the existing data warehouse if that is the intent).

Perm Tables and Pre-perm Stage Tables

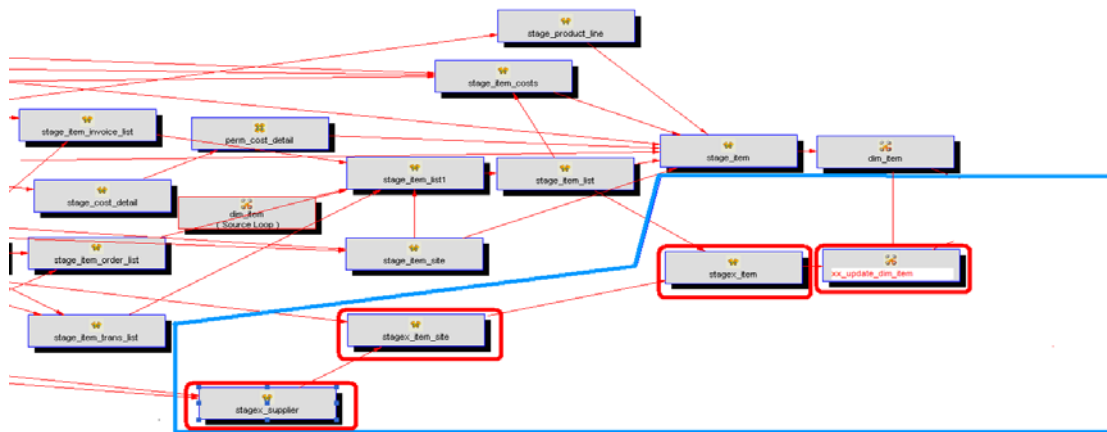
The data from the source table through the load, post-load/pre-perm stage and then perm table in the baseline code is usually left intact with few or no transformations. Changes to these tables in future releases will be limited and will require a minimal number of required changes to then re-add custom columns to this handful of tables if a patch or upgrade does disrupt their infrastructure. As a result, it is advisable to add new custom columns to the existing tables as is recommended for the load tables. If the customizations occur after a historic load for a perm table like perm_transaction_history, which only adds new records on an incremental basis, any custom columns added to that table will need to get populated via either a full historic reload or a special one-time script to update the perm table with the missing column data.

All Other Stage, Extract

To add parallel or custom path tables, it would simply be a drag-and-drop procedure from the source pane, copying the entire table and then deleting the many unnecessary columns. The logic of the WHERE clause that the new table is mimicking should be copied over and made part of the new custom path tables logic so that everything uses essentially the same logic.

Adding Required xx_ Prefixed Columns

After the custom path has been created all the way to the dim or fact (non-history) table, add the extra required xx_ prefixed columns to that table. In the column definition, do NOT give the new xx_ prefixed custom column a source table. It needs to be blank. Transform those column values to some dummy value. Then build a procedure with an xx_ prefix that will update the new custom xx_ columns **after** the table is populated during its normal load process. Below is an illustration of what the resulting diagram would be for a parallel path effort for the dim_item table.



Adding Additional Columns to a Load Table

To add an additional column to a load table that is to coming from the same source, click on the load table that the column is to be added to so that you can see all the columns for that table in the target pane.

Then do the following.

1 Get Column from Data Source

Open a connection to the Progress database in the source pane and find your source table. Double-click on that source table so that all columns appear. Find the new column that currently is not being utilized, click on it and hold the button down, and drag it into the Target Pane to the bottom of the column list so that it will be the last column in the table.

2 Naming/Column Placement/Data type

Now change the prefix of the column name to identify it as a custom addition. Also check to see the Column order to make sure that the new custom column is appearing after the last column number in the table (source_system_code above is Column order 490). Finally, if the column is of a varchar datatype, change it to be an nvarchar datatype.

Column definition for load_idh_hist.xx_idh_prodlne

Column Properties | Transformation

Table: load_idh_hist

Column name: xx_idh_prodlne

Business display name: idh prodlne

Data type: nvarchar(30)

Format:

Column order: 500

Nulls: ☒

Numeric: ☐

Additive: ☐

Attribute: ☒

Source table: idh_hist

Source column: idh_prodlne

Business Definition:
Source strategy: (max 2048 chars.)

3 Transform

If the column is originating from any type of char, varchar, or nvarchar datatype column in the source, perform a transformation on the column to strip out blank spaces and | symbols, which will disrupt data loads. This is done using the following commands, which should be found in any other varchar data type columns in the table.

```
REPLACE(REPLACE(REPLACE(idh_hist.idh_prodlne,'|',' '),CHR(10),' '),CHR(13),' ')
```

This is placed via the Transformation tab.

Column definition for load_idh_hist.xx_idh_prodlne

Column Properties | Transformation

Transformation: Warning: Transformations are only used for file and script loads or when transformations are enabled in the source screen of load table properties for database loads. See Help

Column: xx_idh_prodlne Source table/column: idh_hist.idh_prodlne

REPLACE(REPLACE(REPLACE(idh_hist.idh_prodlne,'|',' '),CHR(10),' '),CHR(13),' ')

The result should be as seen below. Scrolling very far to the right will also show the transformation.

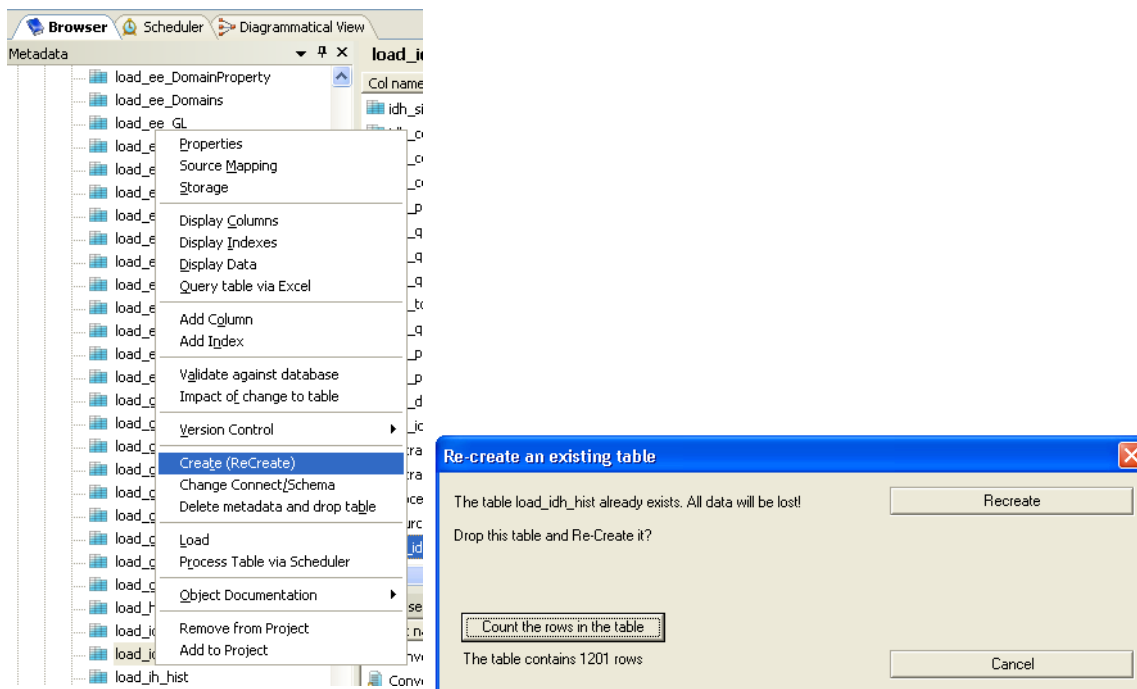
load_gen_led_report_line_range	extract_row_id	extract_row_id	numeric(18)	extract_row_id	N	N	N	N	470	The row n
load_gl_transaction_type	process_batch_id	process_batch_id	numeric(18)	process_batch_id	N	N	N	N	480	The batch
load_glc_cal	source_system_code	source_system ...	nvarchar(64)	source_system_code	Y	N	N	Y	490	The sourc
load_glr_hist	xx_idh_proline	idh_proline	nvarchar(30)	idh_hist	Y	N	N	Y	500	
load_glr_hist_history										
load_glr_hist_list										
load_jhd_mstr										
load_jcc_ctrl										
load_jdh_hist										
load_jh_hist										
load_jn_mstr										

4 Create (Recreate)

With all the changes in place, it is time to re-create the table.

Warning Do not use Create (Recreate) for Dim, Fact or Datastore/Perm tables unless you are sure they are empty).

In the Browser Pane, right-click on the table name, and click the Create (Recreate) option. When the window comes up, do a quick check of the number of records in the table you are about to drop by clicking the Count rows in table button. If ready to move ahead, click the Recreate button.



A confirmation that the table re-created correctly should appear in the results pane.

Results	
Object	Message
load_idh_hist	table comment added.
load_idh_hist	44 column comments added.

Adding Additional Custom Columns to Pre-perm Stage Tables

To add an additional column to a staging table, go to the Browser pane and click on that staging table so that the table's columns are in the Target pane.

Then do the following.

1 Connect

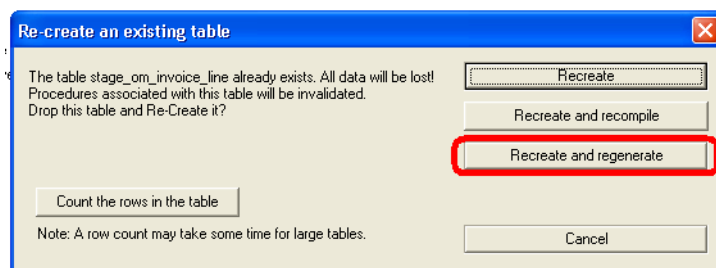
Connect to the Data Warehouse in the Source pane and find the appropriate source table. Double-click on the table and find the column to be added. Click on the column, hold the mouse button down, and drag it to the Target pane at the end of the column list. Note that the DWD did not place the column at the very end of the column list. It may be easier to edit the last column (dss_update_time) and change the column order date for it to 535 so that xx_idh_proline will be last.

2 Transformation/Business Name

Apply any transformation that might be applicable to this column. If this is the pre-perm staging table, this is the best spot to also rename the business name of the column.

3 With the new column in place, Create(Recreate) the staging table.

In the Browser pane, right-click on the table name and pick the Create(Recreate) option. When the window pops up, click on the Recreate and Regenerate button. This only works as long as the existing procedure is not *Modified*. Otherwise manual changes to the procedure will need to be made.



Adding Additional Custom Columns to perm/dim/fact Tables

To add an additional column to a perm, dim or fact table, go to the Browser pane and click on that staging table so that the table's columns are in the Target pane.

Then do the following.

1 Data source

Open the Source Pane using the Data Warehouse ODBC connection. Find the source staging table to be referenced. Double-click the table to get a column list. Drag and drop the column into the perm (fact/dim) table's Target pane to the very bottom of the column list.

2 Placement

Ensure that the column is indeed at the very bottom of the list. Drag and drop any columns below it if it will not go any lower, or renumber the Column Order if necessary to get it to be the last.

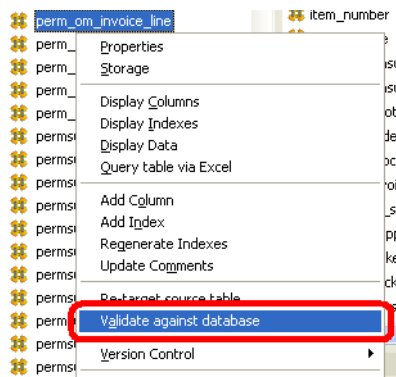
perm_po_order_header	due_date	Due Date	datetime	stage_om_invoice_line	due_date	Y	N	N	Y	Y	430
perm_po_order_line	performance_date	Performance Date	datetime	stage_om_invoice_line	performance_date	Y	N	N	Y	Y	440
perm_po_receipt_hist	required_date	Required Date	datetime	stage_om_invoice_line	required_date	Y	N	N	Y	Y	450
perm_transaction_hist	revised_date	Revised Date	datetime	stage_om_invoice_line	revised_date	Y	N	N	Y	Y	460
permsup_ee_credit_invoice	expiry_date	Expiry Date	datetime	stage_om_invoice_line	expiry_date	Y	N	N	Y	Y	470
permsup_ee_credit_invoice_move	pick_date	Pick Date	datetime	stage_om_invoice_line	pick_date	Y	N	N	Y	Y	480
permsup_ee_debit_invoice	pricing_date	Pricing Date	datetime	stage_om_invoice_line	pricing_date	Y	N	N	Y	Y	490
permsup_ee_debit_invoice_move	promise_date	Promise Date	datetime	stage_om_invoice_line	promise_date	Y	N	N	Y	Y	500
permsup_ee_posting	shipment_type_code	shipment_type...	nvarchar(30)	stage_om_invoice_line	shipment_type_c...	Y	N	N	Y	Y	510
permsup_ee_posting_line	dss_create_time	dss_create_time	datetime		dss_create_time	Y	N	N	Y	Y	1000000008
permsup_mrp_detail	dss_update_time	dss update time	datetime		dss_update_time	Y	N	N	Y	Y	1000000018
permsup_om_invoice_header	xx_idh_prodlne	Product Line	nvarchar(30)	stage_om_invoice_line	xx_idh_prodlne	Y	N	N	Y	Y	1000000028
permsup_om_invoice_line											

3 Column table source and transformation

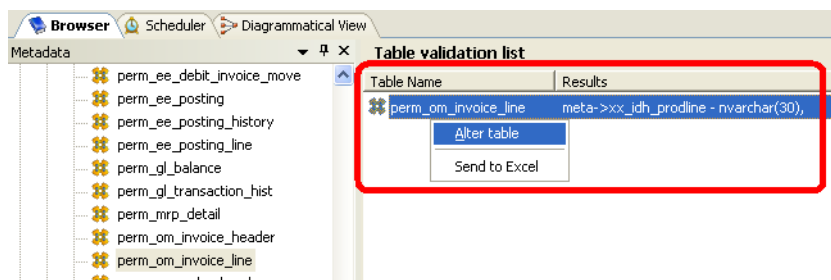
If this column has been added to a dim_ or a fact_ table and is being updated by a procedure, remove the Source Table from the column definition and put in some dummy value in the column Transformation so that a recompilation of the procedure affiliated with this table will not break.

4 Modify table

Do **not** Create(Recreate) the table. If there is any data in one of these tables, the table needs to be altered to accept the new columns. This is done by right-clicking on the table in question, and picking the Validate against database option.



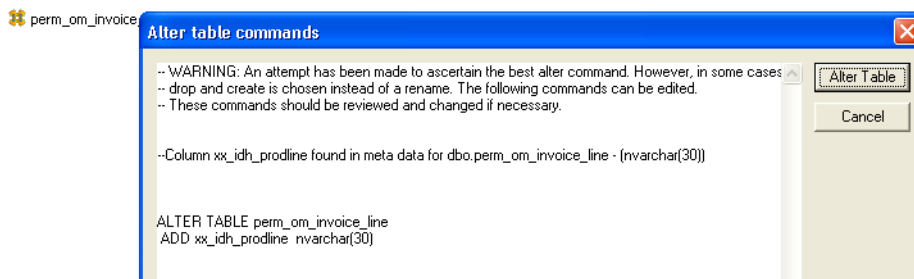
A list of new columns will appear in the Results part of the Table validation list in the Target pane. Right-click on the list and choose the Alter table option.



An editable list of table alteration commands will appear in the window called Alter table commands.

Important It is critical to review the commands listed in this window as sometime it will create commands that will drop all the columns in the table. This is particularly true of dim_ tables.

Any excess incorrect commands can be removed. The only commands that should be used here are to ADD columns or possibly modify data types.



When satisfied with the commands, click on the Alter Table button to make the required additions to the table.

5 Rebuild the procedure

Double-click on the table that has just been modified, and if there is **not** an indicator that the procedure is *MODIFIED* to the right of the Regenerate button, click on the Rebuild button to go through the steps to rebuild the procedure. Pick the default settings on all three screens to quickly pass through the procedure rebuild process.

If any custom code is affiliated with this table, it will need to be modified manually by editing that custom procedure.

Adding Custom Path stage/extract Tables After the perm Tables

If at all possible, any additional columns beyond the perm table should be added via custom tables to ensure smooth upgrades. In the Browser Pane, double-click on the Stage Table header to pull up the complete list of stage tables in the Target pane. Then do the following.

1 Drag and drop new table

In the source pane, be sure the Data Warehouse connection is open and find the source table for the new parallel table. Drag and drop that source table into the Target Pane. A prompt to rename the table will appear. Rename it to match the table it is supposed to be parallel to (if there is a parallel table). Also be sure that the Object type is correct. It should be for Object Type Stage Table whether it is a stage or an extract.

2 Identify and set the columns

Identify the components of the primary key that this table will use to later connect with a dim or fact. Delete all columns except those columns, the new custom column, and any critical date columns at the end of the list. To identify the primary key components, go to the destination table to be mimicked and Display Indexes. Pick the Business key index from the list.

The end result should be a table with only a handful of columns.

extractx_om_invoice_line columns										
Col name	Display	Data type	Src table	Src column	Format	Nulls	Nu...	Ad...	Att...	Ke...
source_system_code	Source System ...	nvarchar(64)	perm_om_invoice_line	source_system_code		N	N	N	Y	10
domain_code	Domain Code	nvarchar(8)	perm_om_invoice_line	domain_code		N	N	N	Y	20
invoice_number	Invoice Number	nvarchar(80)	perm_om_invoice_line	invoice_number		N	N	N	Y	30
order_number	Order Number	nvarchar(80)	perm_om_invoice_line	order_number		N	N	N	Y	40
order_line_number	Order Line Num...	integer	perm_om_invoice_line	order_line_number	###0	N	Y	Y	N	60
dss_update_time	dss update time	datetime		dss_update_time		Y	N	N	Y	530
xx_idh_proline	Product Line	nvarchar(30)	perm_om_invoice_line	xx_idh_proline		Y	N	N	Y	540

3 Create(Recreate) the table

In the Browser pane, right-click on the table name and choose Create(Recreate).

4 Identify the code to be used in the new procedure

Go to the parallel table (if there is one) and view its column sources and its procedure code. Note in the column sources below the inclusion of work_om_change_list.

extract_om_invoice_line columns										
Col name	Display	Data type	Src table	Src column	Format	Nulls	Nu...	Ad...	Att...	Ke...
source_system_code	Source System ...	nvarchar(64)	perm_om_invoice_line	source_system_code		N				
domain_code	Domain Code	nvarchar(8)	work_om_change_list	domain_code		N				
invoice_number	Invoice Number	nvarchar(80)	perm_om_invoice_line	invoice_number		N				
order_number	Order Number	nvarchar(80)	perm_om_invoice_line	order_number		N				
sold_to_code	Sold To Code	nvarchar(80)	perm_om_invoice_line	sold_to_code		Y				
order_line_number	Order Line Num...	integer	perm_om_invoice_line	order_line_number	###0	N				
item_number	Item Number	nvarchar(30)	perm_om_invoice_line	item_number		Y				
invoice_date	Invoice Date	datetime	perm_om_invoice_line	invoice_date		Y				

To view the procedure code Right click on the table name in the Browser pane and choose Code -> View <code name>. In the code screen, scroll down to the FROM/WHERE clause of the query, highlight it, and copy it into a notepad. In this instance we do not need the permsup_ table data as that data will be passed through in the standard modules.

```

dbo Development ( Licensed to: QAD Customer 4 ) -
File Edit View Format Compile Execute Tools Window Help

Browser Scheduler Diagrammatical View update_extract_om_inv_line

perm_om_invoice_line.pricing_date,
perm_om_invoice_line.promise_date,
perm_om_invoice_line.shipment_type_code,
permsup_om_invoice_line.dss_extract_timestamp,
permsup_om_invoice_line.dss_process_batch_id
FROM perm_om_invoice_line
JOIN work_om_change_list
ON perm_om_invoice_line.source_system_code = work_om_change_list.source_system_code
AND perm_om_invoice_line.domain_code = work_om_change_list.domain_code
AND perm_om_invoice_line.order_number = work_om_change_list.order_number
JOIN permsup_om_invoice_line
ON perm_om_invoice_line.source_system_code = permsup_om_invoice_line.source_system_code
AND perm_om_invoice_line.domain_code = permsup_om_invoice_line.domain_code
AND perm_om_invoice_line.order_number = permsup_om_invoice_line.order_number
AND perm_om_invoice_line.invoice_number = permsup_om_invoice_line.invoice_number
AND perm_om_invoice_line.order_line_number = permsup_om_invoice_line.order_line_number

```

5 Change any source tables for the columns with which to build the query.

As was seen in the baseline table, there is an important reference to a work table. We need to have at least one of the columns in this new table defined with a source from work_om_change_list. Each custom table being set up will need to be considered differently based on what the original baseline table does. It is also possible that there may be several baseline tables and the custom path merges all their logic into one table for expediency.

The screenshot displays the 'extractx_om_invoice_line' table definition in the QAD Business Intelligence interface. The 'Column definition' window is open, showing the following details:

- Table:** extractx_om_invoice_line
- Column name:** domain_code
- Business display name:** Domain Code
- Data type:** nvarchar(8)
- Format:** (empty)
- Column order:** 20
- Nulls:** ☐
- Numeric:** ☐
- Additive:** ☐
- Attribute:** ☒
- Source table:** work_om_change_list
- Source column:** domain_code

The 'Column definition' window also shows a list of columns for the 'extractx_om_invoice_line' table:

Col name	Display	Data type	Src table	Src column
source_system_code	Source System ...	nvarchar(64)	perm_om_invoice_line	source_system_code
domain_code	Domain Code	nvarchar(8)	perm_om_invoice_line	domain_code
invoice_number	Invoice Number	nvarchar(80)	perm_om_invoice_line	invoice_number
order_number	Order Number	nvarchar(80)	perm_om_invoice_line	order_number
order_line_number	Order Line Num...	integer	perm_om_invoice_line	order_line_number
dss_update_time	dss update time	datetime	perm_om_invoice_line	dss_update_time
xx_idh_proline	Product Line	nvarchar(30)	perm_om_invoice_line	xx_idh_proline

6 Create (Recreate) the table

Right click on the table name and pick Create(Recreate). The table should be created.

7 Build the procedure for the custom table

Double-click the new custom table name in the Browser pane to open up the Properties window. Click on Rebuild. Follow the standard settings through the first two screens (unless a parameter is required which you would see in the originating table). At the third screen, paste in the code copied in step 4 on page 128. Then hit the OK button.

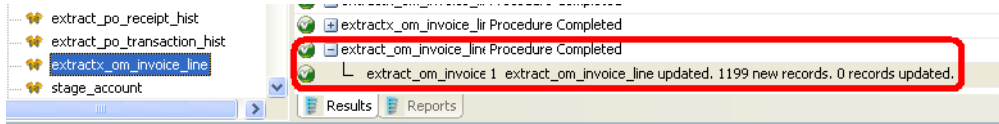
The screenshot shows the 'Provide Cursor mapping' dialog box with the following details:

- Cursor Name (if applicable):** Load
- Source Tables:** perm_om_invoice_line, work_om_change_list
- 'From and Where' clause:**

```
FROM perm_om_invoice_line
JOIN work_om_change_list
ON perm_om_invoice_line.source_system_code = work_om_change_list.source_system_code
AND perm_om_invoice_line.domain_code = work_om_change_list.domain_code
AND perm_om_invoice_line.order_number = work_om_change_list.order_number
```
- ANSI join code generated:** ☒

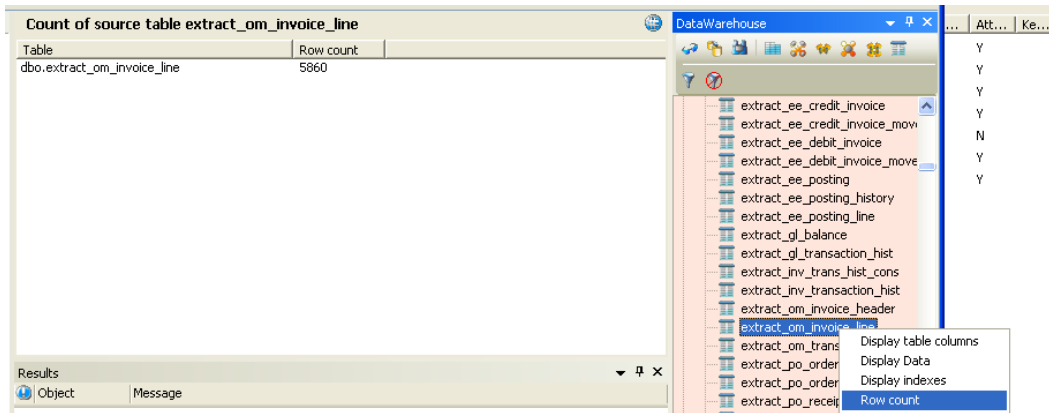
8 Load the table

With the new procedure compiled, load the table by right-clicking on it in the browser pane and clicking on the Execute Update Procedure option. Note the results quantity in the Results pane.

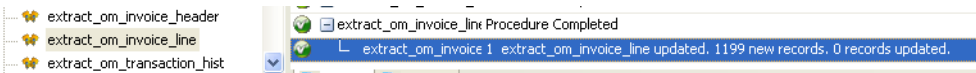


9 Confirm Results

Confirm that the number of records matches the correlating original table. Refresh the Source pane and right-click on original and select Display table count. That number should match the number just loaded in step 7 on page 129.



If the results here do not match, and this is a multi-source system, right-click on the original table in the browser and Execute Update Procedure for that as well. Compare the results again. If they match, move on to building the next table. If they do not match, it will be necessary to extract the original procedure query and the parallel procedure query and start working through them in SQL to see what part of the FROM/WHERE clause may be causing the results to differ. This can be due to lack of an OUTER JOIN or the use of an OUTER JOIN or some other table that was not built into the parallel query.



Adding Additional Custom Loading and Staging Tables to Integrate in Additional Columns

If certain source tables are required by a customer for data that is currently not in the warehouse and we do not currently have matching load tables for these source tables, they can be added and integrated into the warehouse as custom loadx and stagex tables.

If new loadx tables are created, it is critical that:

- 1 All char/varchar columns from the ERP or wherever are nvarchar columns in the new loadx table.
AND
- 2 In the source mapping for that load table, the Unicode mapping option is checked.

Properties | Source Mapping | File Actions | Storage | Notes

Source schema: Update columns and tables from the table column mappings

Source table name(s):

Native Load Routine: Delimiter: UNICODE extract file: ☒

Distinct data select: ☐ Insert hint: (e.g. TABLOCK)

Use source column/transformation: ☒ Uses the information stored against each column rather than 'Source columns' below

Source columns:

Where, Group etc. clauses:
Parameters can be replaced inline by entering them in the following construct \$Pparameter_name\$ (i.e. leading \$P and trailing \$)

Using those custom tables, one of the staging tables will likely have to reference existing staging tables to find a match to the business key of the destination dim or fact table.

The new column will be added to the dim or fact table, but with no source table and have a dummy transformation value set for it.

A post fact or dim table update procedure will then populate the contents of the new columns (unless for a history table) as per the diagrams.

Additional Customization Topics

- Using the wizard to re-create a procedure.** In the few instances that existing baseline tables have custom columns added to them, presuming the procedure is not modified, the wizard should be used to regenerate the procedure that loads that table.
 In considering dim and non-history fact tables that will have the new custom columns populated by a post-load update procedure, it's not actually necessary to update the existing procedure for that dim/non-history fact.
- Dealing with custom/modified code.** In the few instances that existing baseline tables have custom columns added to them, and the correlating load procedures for that table are either Custom or Modified AND that new column needs to be populated by a prior table (for instance history fact tables), the column will need to be carefully added to all relevant places in the corresponding Custom/Modified procedures.
- Populating new custom columns in empty perm/fact tables.** If custom columns have been added to empty perm or fact tables, the tables have been re-created and their procedures regenerated, these tables should populate normally as part of the historic load. In some instances, there will need to be a custom tables created to mimic the original historic load for the population of fact tables.
- Populating new custom columns in already populated perm.** In the event that a new custom column needs to be added to an existing already populated perm table, some type of creative update script will need to be devised that can update that perm table for all of the history records it contains (the exception here is the perm_om_invoice_header and

perm_om_invoice_line tables, which are currently getting repopulated every day). This will be a particularly interesting exercise for the very large tr_hist and gltr_hist and their correlating perm tables, especially if the customer has already archived some of the data in those tables.

Once the data is loaded into the perm tables for those columns, it can be fed on up to the fact tables. Depending on the structure of the customization, this may require a full historic reload of that fact (see historic reload documents), or just a historic reload of that individual column via the custom path. If at all feasible, try to use the latter option.

- **Populating new custom columns in populated non-history fact table** where custom columns were added to tables that come after the perm table (column was in perm table, but not tables after). In the case where the new tables added to a fact already existed in the perm table, the ideal first option is to use the new custom path to do a full historic update of the column in question. If that is difficult, option two would be to simply do a historic reload from perm to accomplish this. See the historic reload document for additional details about how to accomplish this at (contact services for this document).
- **Populating new custom columns in populated history fact table** where custom columns were added to tables that come after the perm table (column was in perm table, but not tables after). In the case where the new tables added to a history fact already existed in the perm table. It is likely that a historic reload from perm that includes the new custom path staging tables will need to be done in order to accomplish this. See the historic reload document for additional details. (Contact Services for this document.)
- **Cloning load tables to bring in outside data sources (flat files or other databases).** If the data from an outside data source needs to be added to the data warehouse, the customer needs to provide a column map of which tables/columns in their outside data source map to which ERP tables and columns. A clone of the existing load tables can be made and mapped to their outside data source. A post-load procedure would need to be created to insert the data from the clone into the existing baseline load tables. The job sequence would then be to:
 - Truncate the existing load table.
 - Run the cloned table load process, which should include the post-load procedure that will populate the baseline load table once the cloned load is complete.

All the remaining steps should be essentially the same. There may be a requirement to reload all data in the fact for that given source if the source system for the outside source is not incremental.

- **Post load procedures on cloned tables to populate existing load tables.** The post-load procedure on a cloned load table simply truncates the load table, then it inserts the contents of the cloned table into the baseline load table (mapped as specified per the customer).
- *****Critical*** - Groups/Projects.** If any customization work is to occur in a customer environment, a Group called Custom Work should be set up with two projects (Custom Work – Modified Objects, Custom Work – New Objects). The projects need to be given the extended name, not just Modified Objects or New Objects, because the DWD tool has some issues around confusing same named Projects. Any custom work done in that environment should then be allocated to the appropriate Project that you just created. This will ensure that a person can find all custom changes in one place when they get into the environment.
- **Customizations for a pre-domain environment (see specific document).** There is a document (contact Services for this document) outlining the steps for reconfiguring the baseline load tables to work with an eB2 environment (does not contain load columns, plus a few other changes). Going this route is OK, but if the customer has an eB2 environment and a

post-eB2 environment, making both work is a fairly big effort. It appears to be much more efficient to have a Progress DBA create a schema in the Progress environment that has mapped views of the eB2 tables and all the custom work from the document is instead mapped into the schema views. That way no customizations need be done to the existing load tables. An effort is underway to have a Progress package that can be provided to eB2 customers that their DBA just installs, but for now, this has only been done at customer sites by the customer's DBA.