



QAD Enterprise Applications

Training Guide QAD Reporting Framework

70-3229-2016EE
QAD 2016 Enterprise Edition
April 2016

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 ©2016 by QAD Inc.

ReportingFramework_TG_v2016EE.pdf/sti/mdf

QAD Inc.

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

Contents

QAD Reporting Framework	
Change Summary	v
About This Course	1
Additional Resources	3
Chapter 1 Introduction	5
Report Viewer Demo & Features	8
Using Filter Screen	9
Filter Screen	10
Search Conditions	11
Settings	12
Output Document Types	13
Scheduled Reports	14
Scheduled Reports	15
Viewer Screen	16
Reporting Framework Architecture	18
Reporting Framework Architecture	20
Reporting Framework Architecture	21
Reporting Framework Architecture: Report Render Engine	22
Report Render Engine	23
Report Render Engine	24
Reporting Framework Architecture: Data Source	25
Report Data Sources	26
Reporting Framework Architecture: Report Layout Definition	27
Report Layout Definition	28
Reporting Framework Architecture: Report	29
Report Output	30
Reporting Framework Architecture: Report Generation	31
Reporting Server Architecture: Report Batch Processor	32
Scheduled Reports	33
Installation and Deployment	34
Report Resource Maintenance	35
Report Resource Maintenance	36
Using Report Resource Maintenance	37

Report Resource Designer	39
Using Report Resource Designer	40
Adding Reports to Menu	41
Adding Reports to Menu	42
Template Designer	43
Template Designer	44
Using Template Designer	45
Report Import/Export and Development Process	47
Report Import/Export and Development Process	48
Report Import/Export and Development Process	49
Browse-to-Report Feature	50
Browse-to-Report Feature: Customization	51
Report Feature: Customization	52
Using Browse-to-Report Feature	53
Browse-to-Report Feature: Output	54
Exercise	55
Exercise	56
Summary	57
Exercise Steps	58

Chapter 2 Development Using Designer63

Report Designer Program	65
Using Report Resource Designer	66
Toolbar Functions	67
Using Report Wizard	69
Report Wizard: Select Report Resource	70
Report Wizard: Select Template	71
Report Wizard: Select Data Source	72
Report Wizard: Select Layout Step	73
Report Wizard: Select Fields (Column Layout)	74
Report Wizard: Select Fields (Column Layout, Groups)	75
Report Wizard: Select Fields (Label Layout)	76
Report Wizard: Summary	77
Manually Working With Layout	78
Manually Working With Layout	79
General Layout Concepts	80
Primary Tool Bar	81
Secondary Tool Bar	83
Reports Tab	85
Data Tab	86
Controls Tab	87
Properties Pane	88
Properties Pane	89

Using Properties Pane	90
Translated Labels	91
Smart Label Selection	93
Number-to-Word Translator	94
Exercise	95
Sub-Reports	96
Sub-Reports	97
Creating Sub-Reports	98
Using Sub-Reports: Tips	99
Using Sub-Reports: Tips	100
Sub-Reports: Exercise	101
VBScript Logic	102
Script Logic	104
VBScript Examples	105
VBScript: Exercise	106
VBScript: Exercise	107
Report Templates	108
Report Templates	110
Report Parameter Maintenance	111
Adding Search Criteria Report Parameter	112
Template Behavior	113
Template Behavior	114
Template Designer	115
Using Template Designer	116
Applying Templates to Report Designs	118
Applying Templates to Report Designs	119
Class Property	120
Overriding Template Attributes	121
Overriding Template Attributes	122
Template Development Guidelines	123
Special Template for Browse Reports	125
Templates: Exercise	126
Templates: Exercise	127
Summary	128
Exercise Steps	129
Chapter 3 Progress Data Source Program	133
Exercise Steps	177
Chapter 4 Reporting Framework Administration	181
System Configuration: client-session.xml	183
Report Data Source Provider Settings	184

Data Source Provider Settings - Change For Financials 185

rptAdmin and rptDsgn Roles / Groups 186

Program Authorization Matrix 187

Adding a Report to the Menu 188

Exercise 189

Scheduled Reports 190

Scheduled Reports: Installation and Deployment 191

Scheduled Reports: Installation and Deployment 192

Reporting Framework: Rendering Architecture 193

Scheduled Report Server - Architecture 194

Batch Processor - Internal Design 195

Scheduled Report Server: Batch Processor Design 196

Scheduled Reports: Multiple Report Server Processes 197

Scheduled Reports Status 198

Scheduled Report Administration: Batch ID Maintenance 199

Scheduled Report Administration: Configuring Windows Task Scheduler ... 200

Printer Setup 201

Using Printer Setup Maintenance 202

Email SMTP Settings 203

Email Template File 204

Scheduled Report Administration 205

End User Report Scheduling 206

Scheduled Report Browse 207

Scheduled Report History Browse 208

Scheduled Report Maintenance 209

Scheduled Reports: Exercise 210

Scheduling Report from API 211

Service Mode 212

Setting up QAD Reporting Framework Service 220

Launching Reports from Progress CHUI 227

Report Bursting 229

Summary 232

Product Information Resources233

QAD Reporting Framework Change Summary

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

Date/Version	Description	Reference
April 2016/v2016 EE	Rebranded for QAD 2016 EE, added links to Preface and book	---
April 2015/v2015EE	Rebranded for QAD 2015 EE	---
March 2014/v2014 SE_EE	Rebranded for QAD 2014 SE_EE	---
September 2013 /v2013.1 SE_EE	Updated and rebranded for QAD 2013.1 SE_EE	---
March 2013/v2013 SE_EE	Rebranded for QAD 2013 SE_EE	---
September 2012/v2012.1 SE_EE	Updated and rebranded for QAD 2012.1 SE_EE	---
March 2012/v2012 SE_EE	Rebranded for QAD 2012 SE_EE	---
September 2011.1/v2011.1 SE_EE	Rebranded for QAD 2011.1 SE_EE	---

<meta name="gsa_source" content=Product Documentation

<meta name="project" content=2016 EE

<meta name="gsa_date" content=2016.04.01

<meta name="ProductSuite" content=Enterprise Edition

<meta name="ProductModule" content=Reporting Framework EE

<meta name="PRODUCT_VERSION" content=2016 EE

<meta name="BookTitle" content=QAD Reporting Framework Training Guide

<meta name="ContentType" content=Training Guide

About This Course

Course Description

This course provides training on Reporting Framework in QAD Enterprise Applications.

- Certification Preparation
- Other QAD Documentation
- Online Help
- QAD Web site
- Conventions

Course Objectives

By the end of this class, students will understand how to:

- Create new reports and add them to the QAD menu
- Work with tools and techniques for page layout design, including use of templates
- Write the Progress code for a custom data source
- Administer the Reporting Framework, including report servers for scheduled batch reports

Audience

- Implementation consultants
- System administrators
- Key users

Prerequisites

Basic knowledge of QAD Enterprise Applications.

Course Credit and Scheduling

This course is valid for 12 credit hours and is typically taught in two days.

Virtual Environment Information

This guide applies to both the Standard Edition and the Enterprise Edition of QAD Enterprise Applications.

- For Enterprise Edition, use the hands-on exercises in this book with the latest Enterprise Edition learning environment in the 10USA > 10USACO workspace.
- For Standard Edition, use the hands-on exercises with the latest Standard Edition learning environment in the Training workspace.

When prompted to log in for either environment, specify *demo* for user ID and *qad* for password.

Additional Resources

If you encounter questions on QAD software that are not addressed in this book, several resources are available. The QAD corporate Web site provides product and company overviews. From the main site, you can access the QAD Learning or Support site and the QAD Document Library. Access to some portions of these sites depends on having a registered account.

<http://www.qad.com/>

QAD Learning Center

To view available training courses, locations, and materials, use the QAD Learning Center. Choose Education under the Services tab to access this resource. In the Learning Center, you can reserve a learning environment if you want to perform self-study and follow a training guide on your own.

QAD Document Library

To access release notes, user guides, training guides, and installation and conversion guides by product and release, visit the QAD Document Library. Choose Document Library under the Support tab. In the QAD Document Library, you can view HTML pages online, print specific pages, or download a PDF of an entire book.

For more information, see *QAD Reporting Framework User Guide*.

To find a resource, you can use the navigation tree on the left or use a powerful cross-document search, which finds all documents with your search terms and lets you refine the search by book type, product suite or module, and date published.

QAD Support

Support also offers an array of tools depending on your company's maintenance agreement with QAD. These include the Knowledgebase and QAD Forums, where you can post questions and search for topics of interest. To access these, choose Visit Online Support Center under the Support tab.

4 Reporting Framework — Training Guide

Chapter 1

Introduction

Course Objectives

Course Objectives

You will learn how to:

- Create new reports and add them to the QAD menu
- Work with tools and techniques for page layout design, including use of templates
- Write the Progress code for a custom data source
- Administer the Reporting Framework, including report servers for scheduled batch reports



RF-IN-050

Terminology

Terminology

- Report
- Report Resource Object (RRO)
- Report Definition (Page Layout Definition)
- Report Template



RF-IN-060

A report is a collection of your data, as defined in the report resource, organized in your required format as defined in the report definition.

Report resources provide the data you want to display in the report while report definitions specify how to display the data in the report:

A report resource object (RRO) represents a unique, cross-domain report object that contains report metadata, report definitions, report data source definitions, filter definitions, report parameters, and report settings. A report definition contains all the information that defines that data binding, layout, and customized formatting of a report. It is saved as an XML file that can be edited in the Report Designer.

A report template is special kind of report definition that cannot be rendered directly by itself, but instead can be used to control certain aspects of the rendering of other reports. When designing a report, a template can be specified (optionally) in which case the report can inherit many kinds of attributes from the template, such as field colors and fonts. If at a later time these attributes are changed in the template, those changes will be seen in every report that is using that template. Any given report can inherit from at most one template, but a given template can be used to control any number of reports. Thus templates enable report developers to making changes in a single place (the template) which will have a mass effect on many reports. This is a powerful tool that can assist the report development process in many ways, such as reducing initial development time, enforcing common standards across reports, and quickly implementing future changes to these standards.

Report Viewer Demo & Features

Report Viewer Demo & Features

- Filter Screen
- Viewer Screen

RF-IN-070

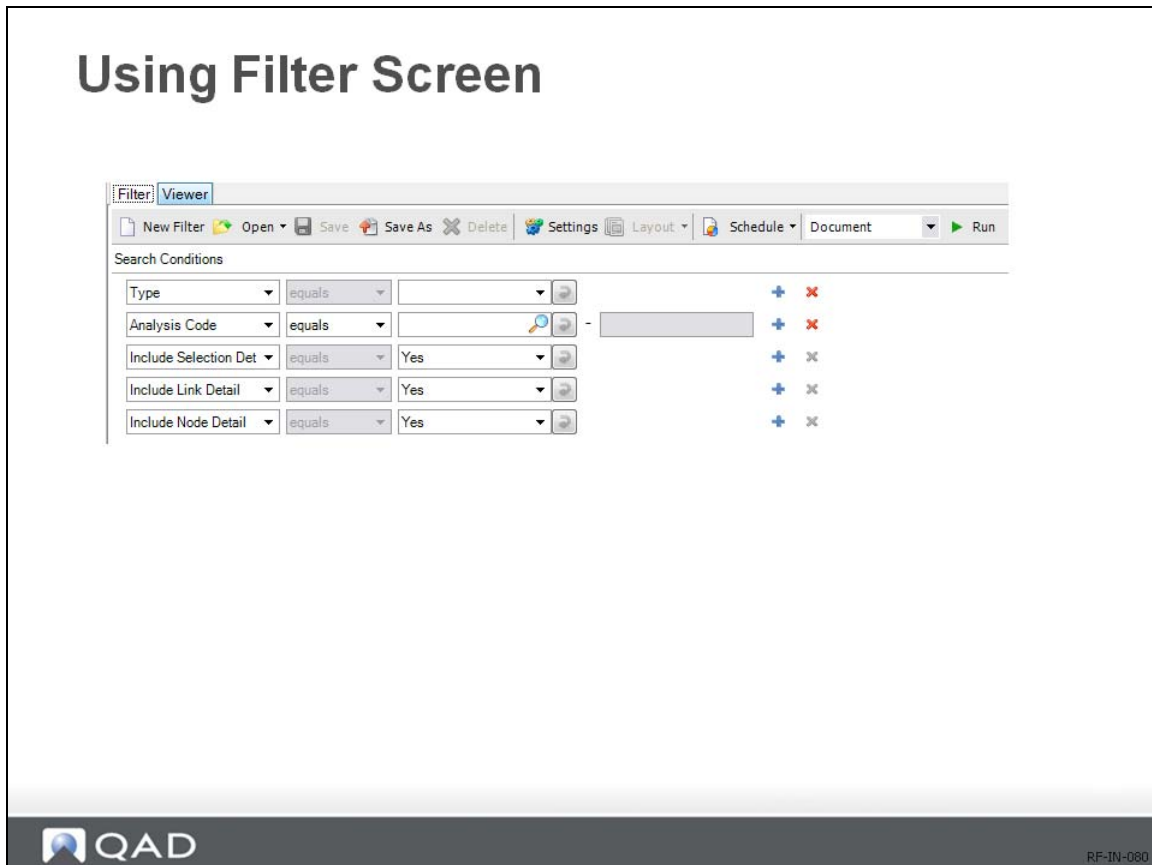
The Report Viewer includes two tabs: Filter for the filter screen and Viewer for the viewer screen.

In Report Viewer, use the toolbar buttons to navigate through the report and perform other functions such as saving and printing.

You can directly run reports from browses by selecting Report from the Action menu in the browse screen. The sorting, grouping, and search criteria in the browse are all carried over to the report, which uses the browse as its data source.

If a report always contains a certain range of data and is exported to a certain format, you do not have to define the filter criteria and output settings every time you generate the report. You can save the search conditions and output settings as a filter and open it to load the same set of configurations when you run the report later.

Using Filter Screen



You can access the Filter screen by clicking on the Report Viewer's Filter tab. On the Filter screen, you can manage search conditions to filter report results, save filters, specify report settings, and specify output document types. Finally, you can schedule reports.

Filter Screen

Filter Screen

- Search Conditions
- Open > More (saved filters)
- Save, Save As
- Delete
- Settings (General, Data, Decimal)
- Layout (available if report resource has multiple report definitions)
- Schedule
 - New (request scheduled reports)
 - View Schedule
 - View History
- Output Document Types
- Run



RF-IN-090

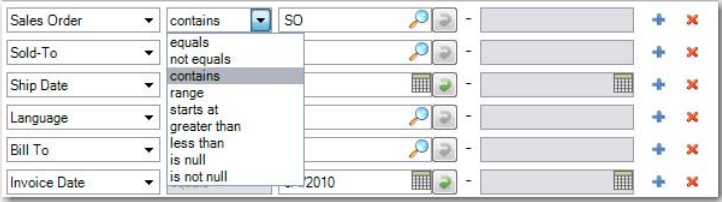
After a report is designed, you can set filter criteria to filter data in the report, run the report, and send it to different output destinations.

Use the Filter screen to set the Search Conditions that specify the filter criteria, the Settings option to specify General, Data, and Decimal preferences, and the output type pull-down to specify output type as Document, Excel, or PDF.

Search Conditions

Search Conditions

- Search Operators include:
 - equals
 - not equals
 - contains
 - range
 - starts at
 - greater than
 - less than
 - is null
 - is not null
- Adding search conditions
- Saving search conditions



The screenshot shows a search configuration window with a table of search conditions. The first row is selected, showing 'Sales Order' as the field, 'contains' as the operator, and 'SO' as the value. A dropdown menu is open over the operator field, listing options: 'equals', 'not equals', 'contains', 'range', 'starts at', 'greater than', 'less than', 'is null', and 'is not null'. Other rows in the table are partially visible, showing fields like 'Sold-To', 'Ship Date', 'Language', 'Bill To', and 'Invoice Date'.

By default, a report will display all the records available in the source data. However, you can retrieve a range of records in the report; for example, sales records between last September and this March. You do this by setting search conditions to filter data in the report. You can also use filters to load existing search conditions.

To refine your search further, click the plus (+) icon to add another search row. You can add as many rows as needed, each with different search values and operators. When you specify several criteria, note that multiple criteria for the same field are treated as a logical AND condition. To remove a search criteria row, click the delete (X) icon.

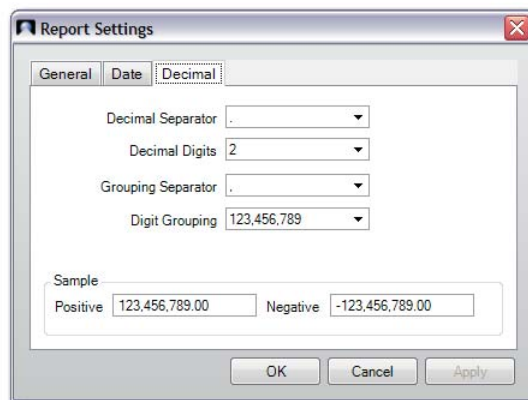
If a report always contains a certain range of data and is exported to a certain format, you do not have to define the filter criteria and output settings every time you generate the report. You can save the search conditions and output settings as a filter and open it to load the same set of configurations when you run the report later.

A filter is a personalized set of search conditions and settings, which means that the filters you created can only be accessed and managed by you and the administrator, and no one else.

Settings

Settings

- General
- Data
- Decimal



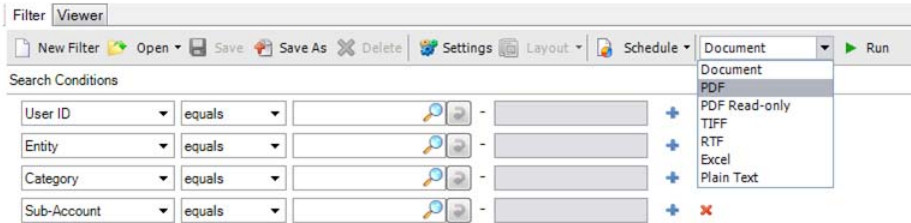
Use Settings (Report Settings window) to customize how certain elements of data will be displayed in the rendered report. In the Filter screen, click Settings on the toolbar to bring up the Report Settings dialog box.

Under the General tab, specify whether to display search criteria in the report, and if yes, whether to display this information in the report header or footer. Under the Date tab, select a format for the dates to be displayed in the report and specify a date separator. You can see a sample of the date format you specify at the bottom of the dialog box. Under the Decimal tab, specify how numbers will be displayed in the report, including decimal separator, decimal digits, grouping separator, and grouping format. A sample number is displayed at the bottom of the dialog box.

Output Document Types

Output Document Types

- Select output type from pull-down
- Output types include:
 - Document (Viewer)
 - PDF, PDF Read-only
 - TIFF, RTF
 - Excel
 - Plain Text



Scheduled Reports

Scheduled Reports

- To schedule reports, choose Schedule > New
- Specify options, including:
 - Batch ID
 - Printer
 - E-Mail
 - Save Report Output
 - Run Once



RF-IN-130

Batch ID. Specify the batch ID for the scheduled report. The batch ID is created by the administrator in Batch ID Maintenance (36.14.1) and determines when and how often the report will be run on the report server.

Printer. If you want to have the scheduled report printed, specify a printer to send the report to. Printers are set up for the report server by the administrator in Printer Setup Maintenance (36.13.2).

E-Mail. Enter e-mail addresses or Inbox user IDs you want to have scheduled report notifications sent to. Separate multiple entries with commas.

Save Report Output. Select this option if you want the report server to send the scheduled report output file to the document service.

Run Once. Select this option if you want to mark the scheduled report as non-permanent. A non-permanent scheduled report will run only once with the next batch run, regardless how many times the associated batch is scheduled to run. A permanent scheduled report will run every time the batch is run.

Scheduled Reports

Scheduled Reports

- To view scheduled reports, use Scheduled Report Browse (choose Schedule > View Schedule)
- To maintain a scheduled report, use Scheduled Report Maintenance
- To view scheduled report history, use Scheduled Report History

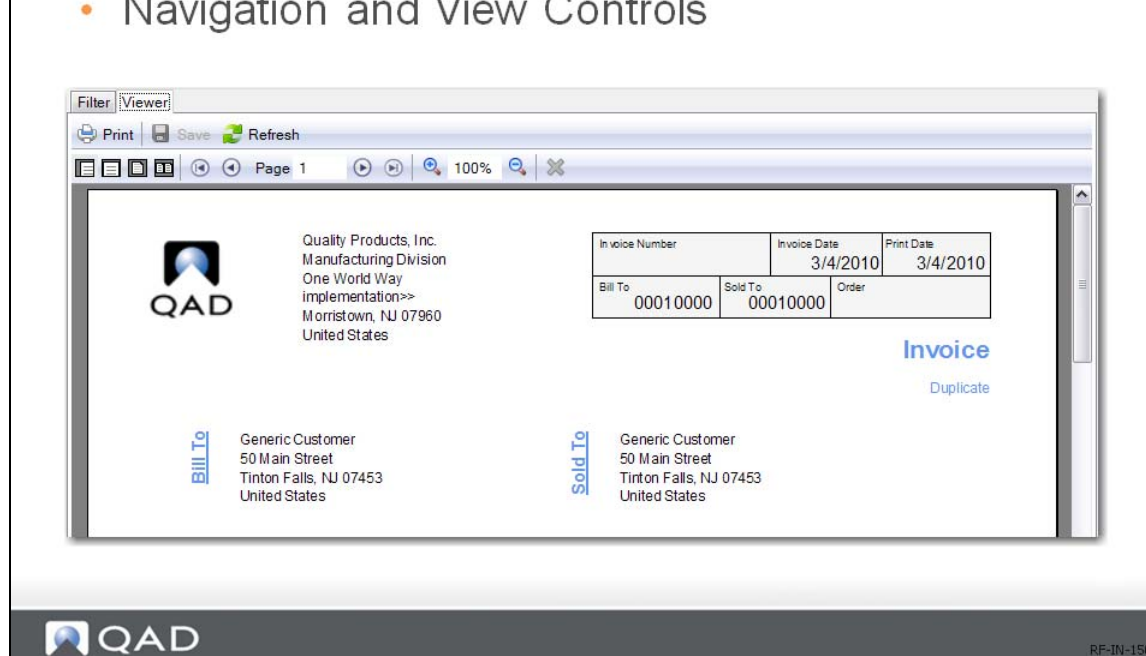


RF-IN-140

Viewer Screen

Viewer Screen

- Toolbar (Print, Save, Refresh)
- Navigation and View Controls



The Viewer screen includes the following options:

Print. Send the report to a printer.

Save. Save the report to a specified location.

Refresh. Regenerate the report using your last setting.

Actual Size. Display the report in its actual size.

Page Width. Fit the report to the width of the Viewer screen.

One Page. Display the report in a one-page view in the Viewer screen.

Two Pages. Display the report in a side-by-side two-page view in the Report Viewer window.

First Page. Jump to the first page.

Previous Page. Go to the previous page.

Next Page. Go to the next page.

Last Page. Jump to the last page.

Zoom In. Magnify the report preview size.

Size. Specify the exact report preview size.

Zoom Out. Decrease the report review size.

Cancel Rendering. Cancel rendering the report.

Reporting Framework Architecture

Reporting Framework Architecture

- High Level Overview
- Report Render Engine
- Data Source
- Layout Definition
 - Pre-Render Engine



RF-IN-160

The QAD Reporting Framework contains five key components: report render engine, report layout definition, data source, pre-render engine, and report.

- Report render engine

As the core of the solution, the report render engine takes in data set and report layout definition as inputs, then renders and produces the report output. Since the QAD Reporting Framework is a .NET solution, the report render engine can only run on the Windows operating system.

The rendering process takes place on the computer that actually runs the report. If you run a report in the QAD .NET UI on your PC, then your PC's CPU power is consumed to render the report, which helps to distribute the processing load across client machines.

- Data source

Data to be displayed on the report are queried from the underlying business system through the data source definition. The QAD Reporting Framework supports three types of data sources: generic proxy (Progress program), browse, and Financials API.

- Report layout definition

Report layout definition defines what gets displayed on the report, and where. You use Report Designer in the QAD .NET UI to define the report layout in WYSIWYG (What You See Is What You Get) fashion. You can also import and export report layout definitions as XML files, which makes it very easy for you to deploy reports or migrate them between systems, such as moving reports from the test environment to the production environment.

- Pre-render engine

The pre-render engine pre-processes the report layout definition by applying a report template to it as well as performing label translations and produces a modified report layout definition. The resultant report layout definition along with the data source are then fed into the report render engine, which generates the actual report.

- Report output

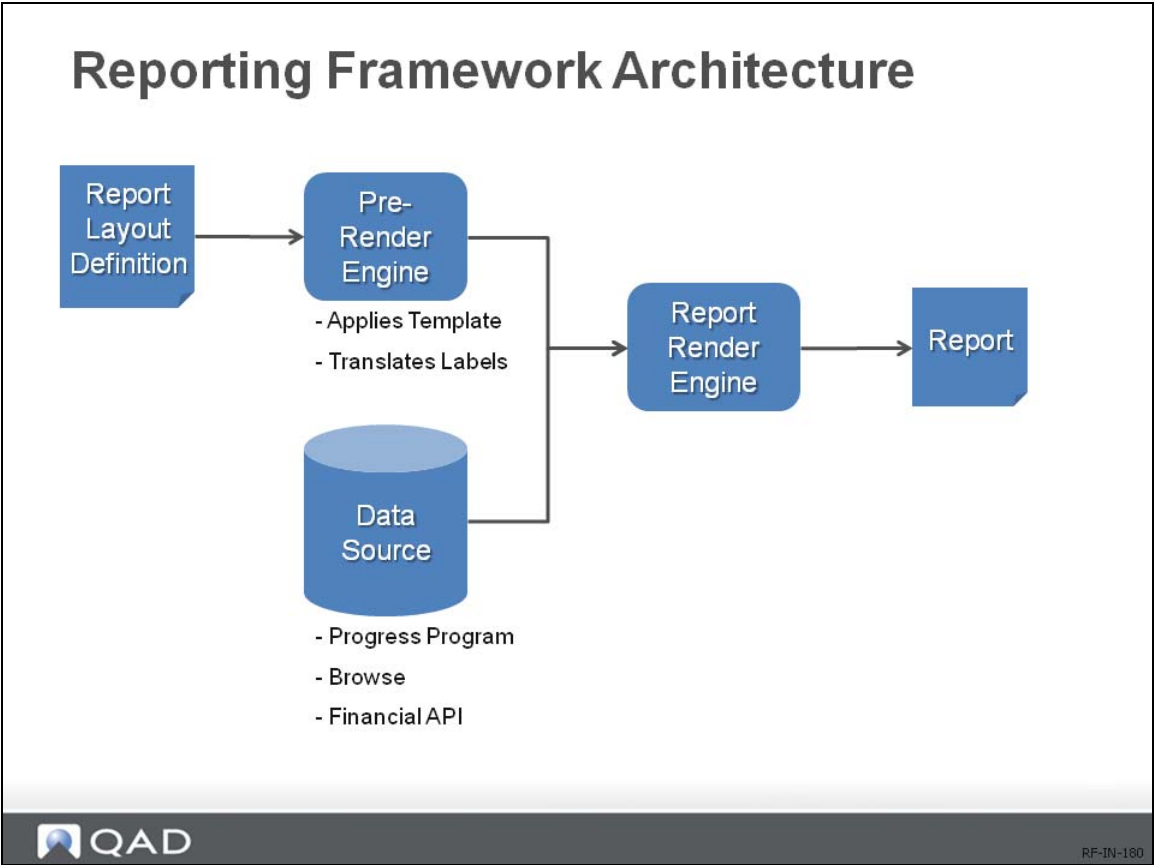
The report output can be rendered in three different formats, depending on your preference. The default format is a document displayed on the screen, which you can view by paging and zooming. You can then send it to printer if you want a hard copy. Alternately, the report can be exported into the PDF or Excel format, which you can print or save as a file.

Reporting Framework Architecture

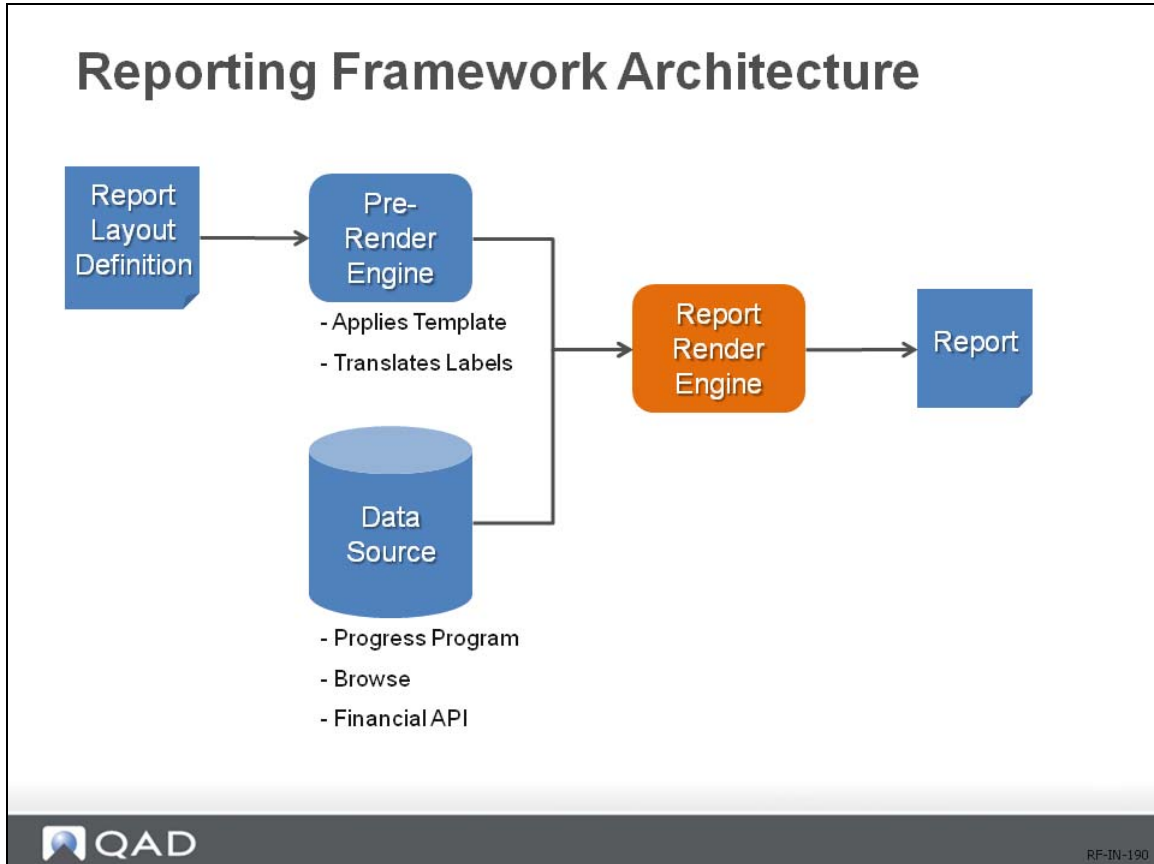
Reporting Framework Architecture

- Report Output
- Report Server and Scheduled Reports
- Installation and Deployment
 - Client
 - Server

Reporting Framework Architecture



Reporting Framework Architecture: Report Render Engine



Report Render Engine

Report Render Engine

- Third party tool (Component One)
 - No end-user license fees to QAD customers
 - <http://www.componentone.com/>
 - <http://helpcentral.componentone.com/CS/forums/>
- .NET (Windows only) solution



RF-IN-200

The report render engine takes in data set and report layout definition as inputs, then renders and produces the report output.

The report render engine is based on Component One.

The report render engine is a .NET solution: the report render engine can only run on the Windows operating system.

The rendering process takes place on the computer that actually runs the report. If you run a report in the QAD .NET UI on your PC, then your PC's CPU power is consumed to render the report, which helps to distribute the processing load across client machines.

Report Render Engine

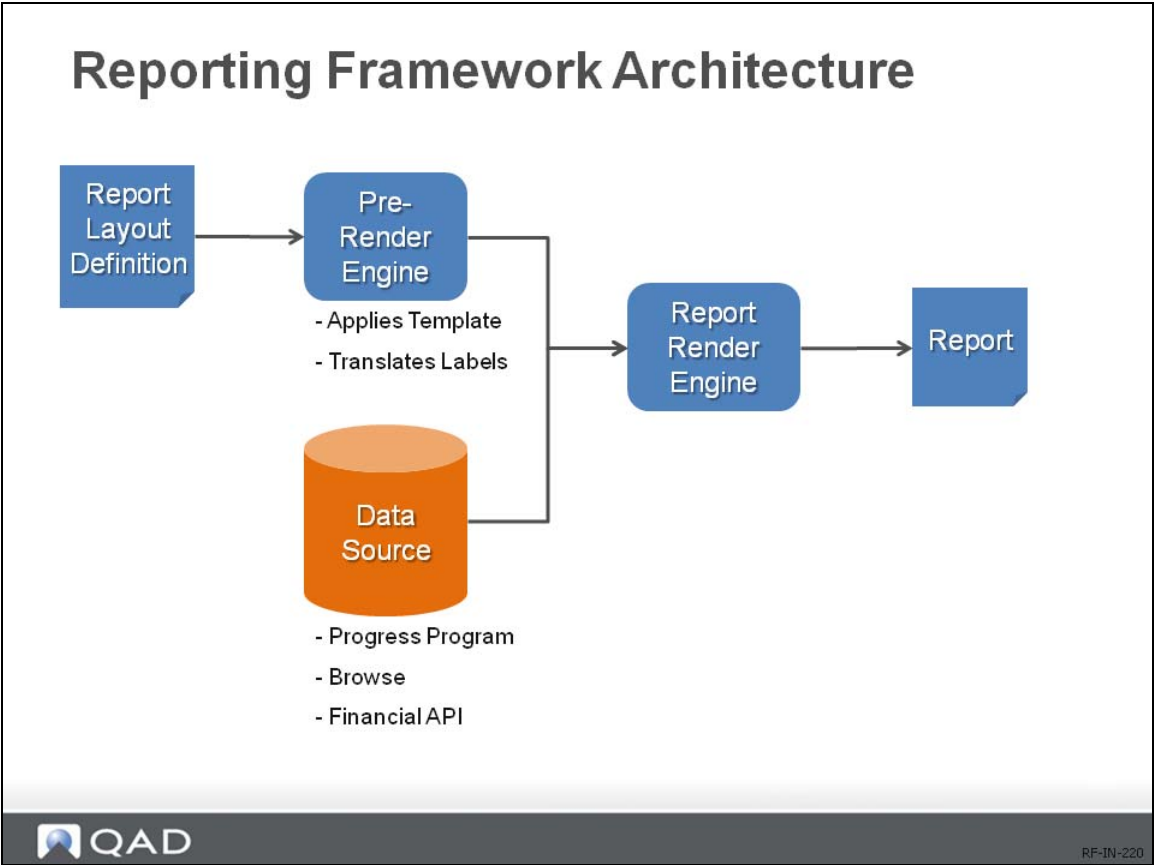
Report Render Engine

- Inputs:
 - Data set (queried from data source)
 - Report layout definition (page layout)
- Output:
 - Rendered report

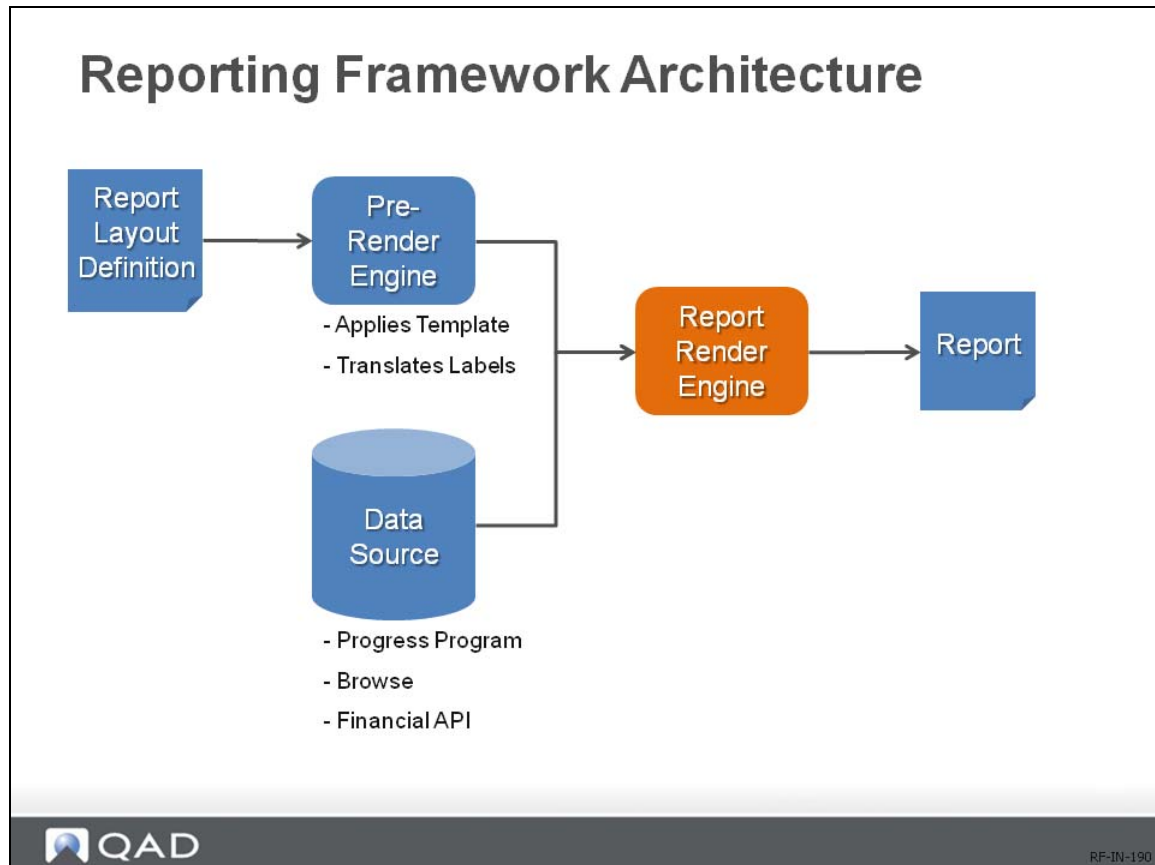


RF-IN-210

Reporting Framework Architecture: Data Source



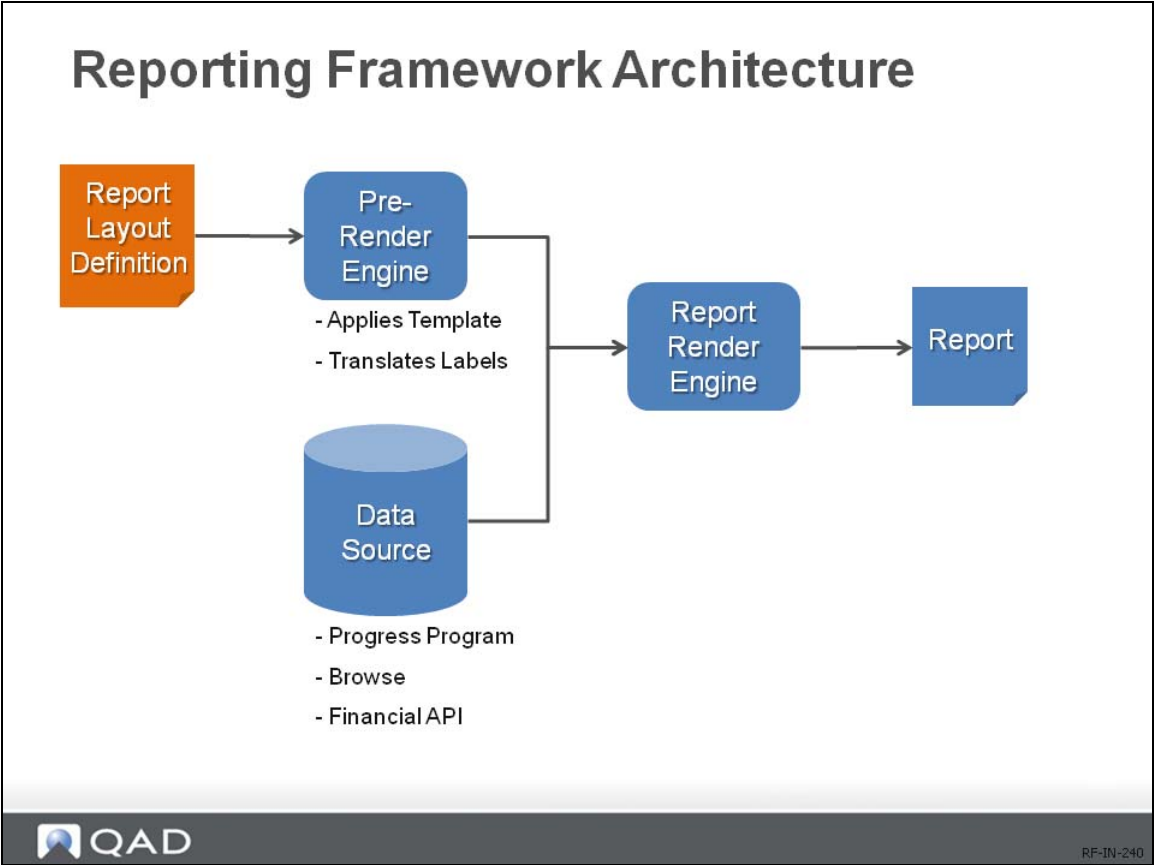
Report Data Sources



Before you create a report, you need to determine where your report data comes from. The QAD Reporting Framework supports three types of data sources: generic proxy (Progress program), browse, and QAD Financials API. Depending on which type of data source you use, you might need to perform some additional implementation steps.

Note It is even possible to create a new data source type to connect to other stores of data. This would require writing a .NET implementation of the `IDataSourceProvider` interface.

Reporting Framework Architecture: Report Layout Definition



Report Layout Definition

Report Layout Definition

- Defines what gets displayed where on the page
- Created by report designer
- Report Resource Designer program
 - Graphical drag-and-drop layout editor
 - Banded layout model (like Crystal, Jasper)
- Import/Export tool
 - Allows reports to be controlled as XML files
 - Useful for version control, deployment

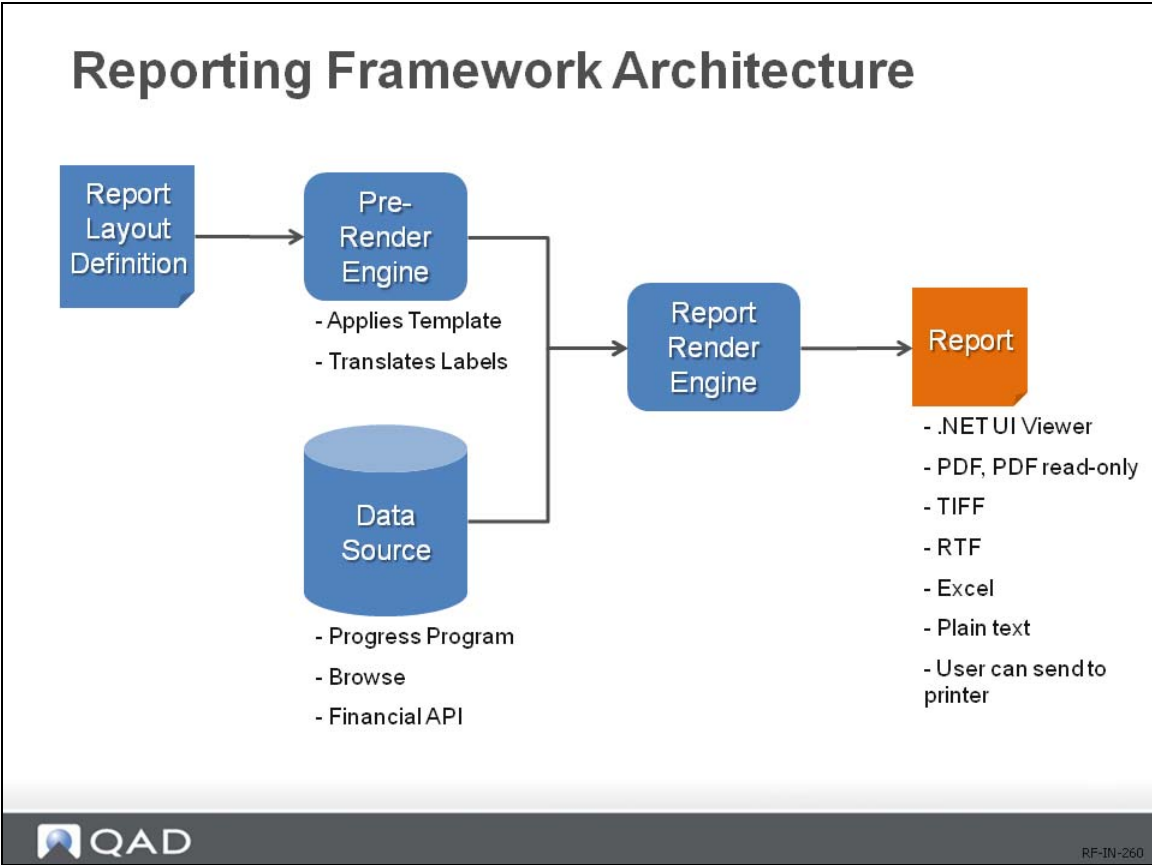


RF-IN-250

A report definition contains all the information that defines the data binding, layout, and customized formatting of a report. It is saved as an XML file that can be edited in the Report Designer, either visually or in the text editor mode.

Use the Report Resource Designer program to create report definitions.

Reporting Framework Architecture: Report

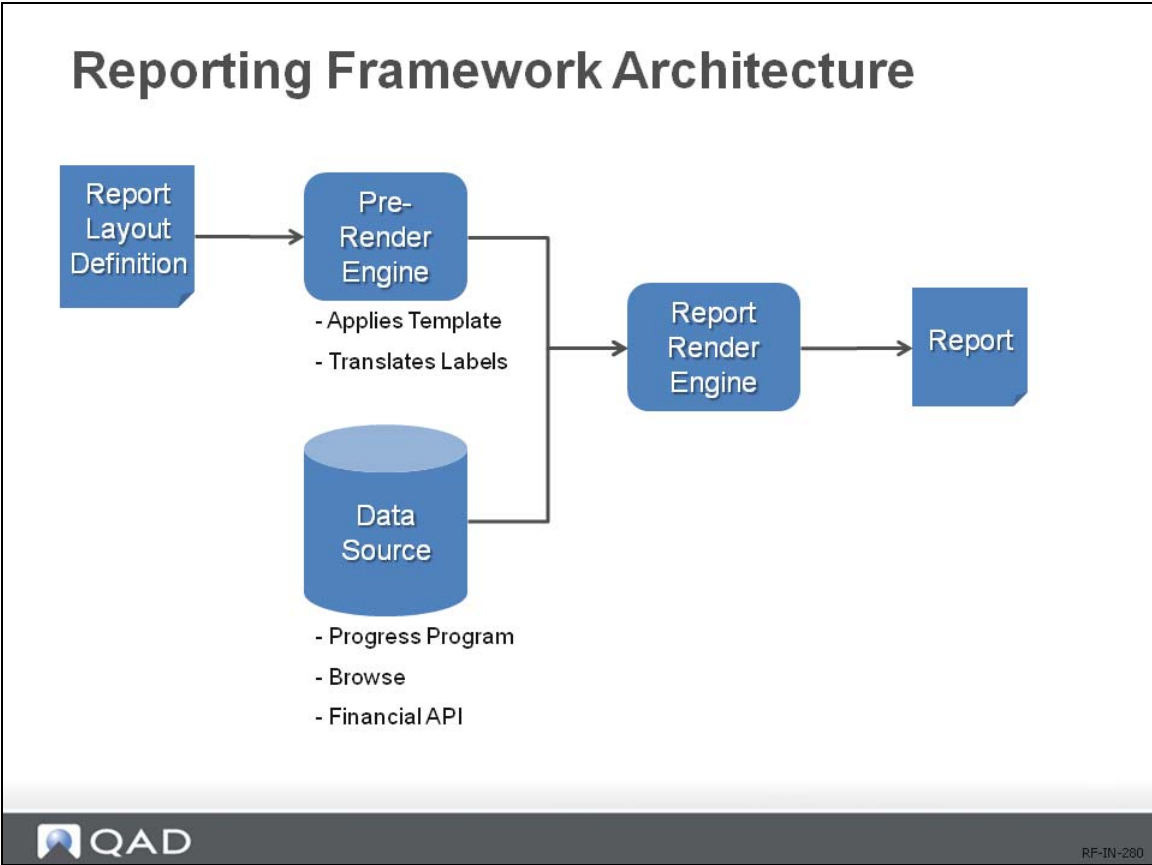


Report Output

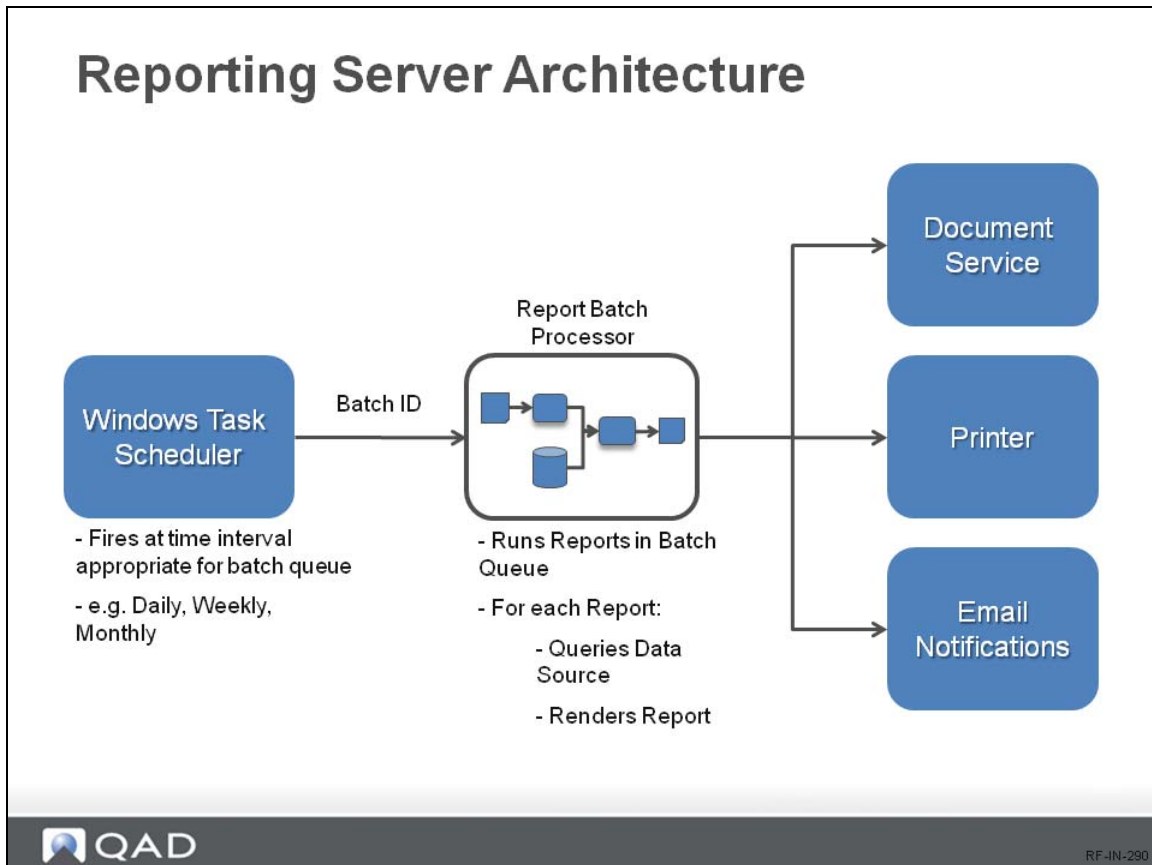
Report Output

- Reports can be rendered in various formats:
 - Document (.NET UI Viewer)
 - PDF file
 - PDF read-only file
 - TIFF file
 - RTF file
 - Excel file
 - Plain text file

Reporting Framework Architecture: Report Generation



Reporting Server Architecture: Report Batch Processor



Scheduled Reports

Scheduled Reports

- Report Server(s) run queues of reports (batches)
- Server processes
 - Non-GUI .NET UI processes
 - Standard installation of .NET UI
 - Launched from command line
- Windows Task Scheduler used to periodically launch batches (e.g. daily, monthly)
- Output to printer and/or file on web server
- Optional notifications to email, .NET UI Inbox



RF-IN-300

You can automate the process of generating routine reports by scheduling them to automatically run at specified times or intervals and have the reports sent to a specified destination, such as a printer or the document service on the report server.

To schedule reports to run at a specified time or interval, on the report server, you create a Windows scheduled task for a batch and group the reports in the batch.

The Windows Task Scheduler should be configured to launch the Report Batch Processor, which is a non-GUI instance of the QAD .NET UI launched from the command line, for a specific batch as scheduled and runs all the scheduled reports grouped in the batch. If already set up, the report outputs are sent to the QAD .NET UI document service and/or server-side printer as configured.

Multiple report servers can be set up for increased throughput and failover. The different servers can be configured to process different batches, or can even jointly process reports in the same batch. The Report Batch Processor coordinates the processing of scheduled reports with different priorities in the correct sequence across multiple report servers.

Installation and Deployment

Installation and Deployment

- Report Framework seamlessly integrated into .NET UI
- No special installation needed
- No license fees to customers
- Report Server Installation
 - Windows OS required
 - Install QAD .NET UI client in standard fashion
 - Run the server using command line (no GUI)
 - Can deploy multiple servers
 - Failover
 - Increased throughput



RF-IN-310

Report Resource Maintenance

Report Resource Maintenance

- Specify Report Code of your choice
 - Must be unique
 - QAD-shipped reports use QAD_ prefix
 - Do not modify QAD_ reports
- Specify Category (report type) as Report
- Specify Data Source Type:
 - Browse
 - Proxy
 - FinancialAPI (EE only)



RF-IN-320

Use Report Resource Maintenance to create a report resource.

Report Resource Maintenance

Report Resource Maintenance

- Specify Data Source Ref (reference) based on data source type:
- Browse ID -- if using Browse data source
- Program name (.p) -- if using Proxy data source
- Financial report query object -- if using FinancialAPI data source

Using Report Resource Maintenance



Open Report Resource Maintenance and enter the following fields. Click Next or press Enter to move to the next frame or field; click Back to return to the previous field.

Report Code. Specify a code that identifies a report resource.

Important QAD-provided built-in reports, report resources and templates all begin with “QAD_”. Do not create or modify reports, report resources, or templates with this prefix. Otherwise, your customized changes will get overwritten during system upgrades from QAD.

Category. Select one of the following report resource types for different report providers: Report for QAD .NET-based reporting and Dashboard for Cognos dashboard reports.

Note Cognos dashboard reports are currently not implemented.

Data Source Type. Specify a data source type that indicates how the report retrieves its data:

- **Browse:** The report uses browses as its data source. Both classic browses (created in Browse Maintenance) and Financials browses (created in the Financials CBF tool) are supported.
- **Proxy:** The report accesses the database through the generic proxy program.
- **Financials API:** The report retrieves data through the QAD Financials API.

Data Source Ref. Provide the reference information for retrieving data through the data source provider.

- For classic QAD ERP browses maintained in Browse Maintenance, enter `<BrowseServerType>:<QueryID>`; for example, `QAD.Browse.MfgProBrowseServer:so009`. `<BrowseServerType>` is optional. If it is not specified, the data source reference defaults to the QAD ERP browse server. In this case, `so009` is equivalent to `QAD.Browse.MfgProBrowseServer:so009`.

For Financials browses created in the Financials CBF tool, enter the following:

```
BaseAdapters.CBAdapters.QadBrowseAdapter:<BusinessComponent>.<QueryMethod>
```

Here is an example using this naming syntax:

```
BaseAdapters.CBAdapters.QadBrowseAdapter:BJournalEntry.SelectPosting
```

- For the generic proxy data source, specify a data source proxy program file name; for example, `myReport.p`.
- For the Financials API data source, specify a Financials reporting component name followed by a method name; for example, `BGLReport.GLList`, where `BGLReport` is a component name and `GLList` is a method name.

Description. Provide a description of the report resource.

Default Definition. Specify the default report definition for the report resource. When you open a report resource in Report Viewer or Report Designer, this report definition is loaded by default.

Report Resource Designer

Report Resource Designer

- Create and modify page layout designs (that is, "report layout definitions")
- Report code must already exist (see Report Resource Maintenance)
- Report can have one or more definitions
- Graphical drag-and-drop layout editor
- Organized into sections (or "bands")



RF-IN-350

Use the Report Resource Designer to create a report definition.

- 1 Type Report Resource Designer in the menu search field and press Enter.
- 2 Click the New icon on the Report Designer Toolbar. The Report Wizard window appears.

Select the report resource you previously created and click Next.

Using Report Resource Designer



To use the Report Resource Designer:

- 1 Type Report Resource Designer in the menu search field and press Enter.
- 2 Click the New icon on the Report Designer Toolbar. The Report Wizard window appears. Select the report resource you previously created and click Next.
- 3 Select a report template or select None to use the default built-in report template. Click Next.
- 4 Select a table as the report data source and click Next. All the available tables you can select as data sources are listed in a tree.
- 5 To view all the fields in a table, click the plus sign next to the table to expand the tree.
- 6 This screen offers you several options to define how the data will be organized on the page. Select the layout that best approximates what you want the final report to look like.
- 7 Select Fields into the report.
- 8 The Summary screen recaps the information you have specified for the report definition. If you want to modify the settings, click Back to return to previous steps to edit them; otherwise, click Finish to complete the basic report setup and exit Report Wizard.
- 9 When you return to the Report Designer main screen, the report displays in the visual design mode in the Design pane based on the newly created report definition. Save the report as a new report definition. You can further customize it in Report Designer.

Adding Reports to Menu

Adding Reports to Menu

- Report code must exist with at least one layout definition
- The default definition will be executed when the menu item is run
- Use Menu System Maintenance to create a standard menu item
- Exec Procedure format:

`urn:qad-report:c1:ReportCode`



RF-IN-370

Users open reports through report menu items that they have been given access to based on their role membership. Use Menu System Maintenance to create a menu item to provide access to a report resource.

In the Exec Procedure field, enter a component-based activity specified in the form of a uniform resource name (URN):

`urn:qad-report:c1:ReportCode`

Where *ReportCode* is the report code of the report resource you are creating a menu item for.

Adding Reports to Menu

Adding Reports to Menu

- Grant menu item authorization using standard menu security tools
 - SE: Menu Security Maintenance
 - EE: Role Permissions Maintain



RF-IN-380

Template Designer

Template Designer

- Used to create and modify report templates
- A report template is a mechanism by which attributes can be inherited by one or more reports
 - Similar to CSS in web page design
 - Inherited attributes include properties such as fonts, colors, and alignment
 - Fields can be inherited (for example, standard page header content, logos)

Template Designer

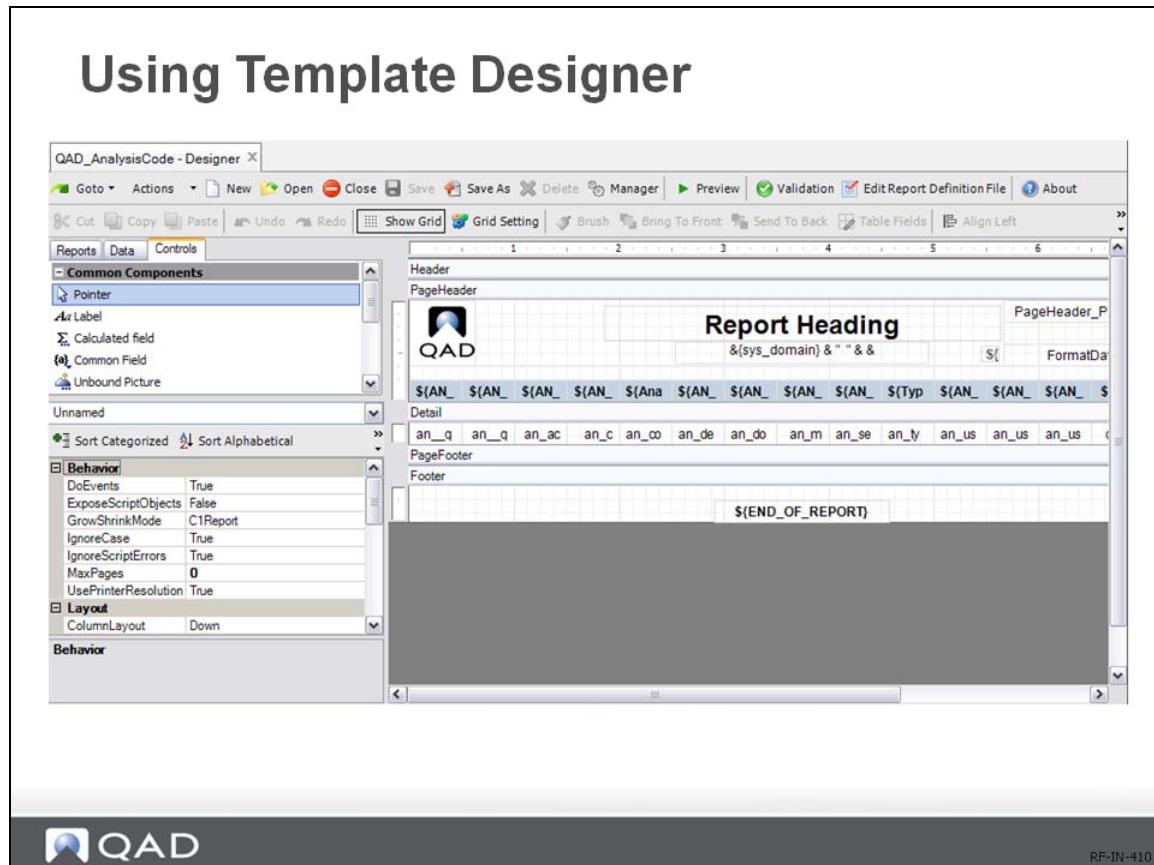
Template Designer

- Allows for mass changes to many reports by changing just a template common to them
- Useful mechanism for standardization



RF-IN-400

Using Template Designer



The Template Designer works similarly to the Report Resource Designer, but there are a few important differences.

Fields placed in template sections (except for the Detail section, to be discussed next) behave much like fields in Report Resource Designer: the fields themselves will get put onto any reports whose sections inherit from the template sections. For example, the logo, title, and other fields in the PageHeader section of the template will automatically appear as fields in all reports that have sections pointing to the template's PageHeader section.

Fields in the Detail section of a template have a completely different behavior: they do not actually appear on any reports, but instead can be used to define properties (e.g. font, fore color) that can be inherited by actual fields in reports. Report fields can specify which template field to inherit properties from.

To use the Template Designer:

- 1 Launch Template Designer. Type Template Designer in the menu search field and press Enter.
- 2 Click the New button on the toolbar.
- 3 In the Create Template dialog box, enter a unique template name and click OK.

Important QAD-provided built-in reports, report resources, and templates all begin with “QAD_”. Do not create or modify reports, report resources or templates with this prefix. Otherwise, your customized changes will get overwritten during system upgrades from QAD.

- 4 In the Design pane, create and format field classes in the same way as you work with fields when working with a report definition. Provide unique class names for the classes.
- 5 If you want to define a header, page header, page footer, and footer section class name, click the default section name and enter a new name in the (name) field in the Properties pane.
- 6 If you want to add new sections, use the following steps:
 - a Click the New Group button on the toolbar.
 - b In the Edit Group dialog box, click Add and specify the properties for the new group.
 - c Click OK.
- 7 Configure the default section and field class mapping to specify the default classes to be applied to the corresponding sections and fields in the report definition.
 - a Click the Configure button on the toolbar. The Class Configuration Form dialog box appears.
 - b Under the Section Configuration tab, specify a class name for each section type.
 - c Under the Field Configuration tab, for each data type, select a section in the template and specify a class defined within that section.
 - d Click OK.
- 8 Back in the Template Designer main screen, click the Save button on the toolbar to save the template.

Report Import/Export and Development Process

Report Import/Export and Development Process

- Report Resource Export
- Report Resource Import
- Development Process Guidelines



RF-IN-420

When a report developer saves entries in Report Resource Maintenance, or saves report layout definitions in Report Resource Designer or Template Designer, the information is saved into tables in the qadadmin database. The export and import programs provide a way to conveniently get this data into and out of the database by allowing the data to be serialized to and from local XML files.

Use Report Resource Import to import report resource data files into the system.

Use Report Resource Export to export data of specified report resources into .xml files. This function lets you back up report resources or create report resource files to be imported into another system.

Report Import/Export and Development Process

Report Import/Export and Development Process

- Report Resource Import / Export programs
 - Import report definitions from XML files to DB records
 - Export report definitions from DB records to XML files
- Allows reports to be controlled as XML files
- Useful for:
 - Backup
 - Version control
 - Deployment



RF-IN-430

It is strongly recommended that report developers regularly export the reports they are creating to XML. Besides providing data backup and providing a textual artifact that can be version controlled, exported reports can also be used to resolve problems resulting from multiple report developers inadvertently working on the same report (which should generally be avoided).

Report Import/Export and Development Process

Report Import/Export and Development Process

- Granularity: One XML file per RRO
- XML contains the following:
 - All information about the RRO
 - All report layout definitions for the RRO
 - Contains reference to data source, but not the data source itself
(Proxy.p data source programs need separate file control)

Browse-to-Report Feature

Browse-to-Report Feature

- Available on all QAD browses
- Creates a report from any running Browse
 - Alternate visualization of the browse data
- Run from browse
 - Choose Action > Report
 - Report is dynamically auto-generated
 - Report is opened in a new tab and auto-run
 - All records matching the browse search conditions will be queried for the report



RF-IN-450

Browse-to-Report Feature: Customization

Browse-to-Report Feature: Customization

- User can customize the report layout using browse settings
 - For example:
 - Column size, order and visibility
 - Summaries
 - Groups
 - Sort column
 - Limitation:
 - Chart in browse is not put into report

Report Feature: Customization

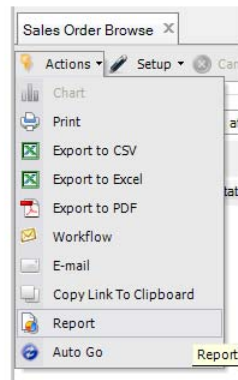
Browse-to-Report Feature: Customization

- Saving browse to Favorites menu saves settings
- Empowers any end user to perform report layout customization

Using Browse-to-Report Feature

Using Browse-to-Report Feature

- Open a browse
- Choose Actions > Report



Browse-to-Report Feature: Output

Browse-to-Report Feature: Output

Sold-To
00010000

Sales Order	Revision	Status	Line	Item Number	Unit of Measure	Quantity Ordered	Quantity
d725058d	2		1	kitds	EA	-1.0	
d725058d	2		2	D-01	EA	1.0	
d725058d	2		3	D-01	EA	4.0	
d725058f	2		1	kitds	EA	1.0	
d725058x	2		1	kit2ds	EA	1.0	
d725058x	2		2	1-BB	EA	13.0	
d725058y	2		1	kit2ds	EA	-1.0	
dm101	0		1	1-BB	EA	10.0	
dzv009	1		1	11.	EA	12.0	
dzv2	0		1	Ⓜ:~1~€~™~<~>~	EA	21.0	
dzv3	0		1	Ⓜ:~1~€~™~<~>~	EA	123.0	
dzv3	0		2	Ⓜ:~3~€~™~<~>~	EA	5,431.0	
dzv5	0		1	Ⓜ:~1~€~™~<~>~	EA	3.0	
dzv5	0		2	Ⓜ:~3~€~™~<~>~	EA	1,235,234.0	1.2

RF-IN-490

Exercise

Exercise

1. Create a new Sales Order report based on a browse data source
 - Report Resource Maintenance
For *Data Source Ref*, use so009
(Sales Order Browse)
 - Report Resource Designer
2. Test run from Report Resource Designer

Exercise

Exercise

3. Export the report to XML
 - Report Resource Export
4. Add the report to the menu
 - Menu System Maintenance
 - Role Permissions Maintain (EE)
 - To refresh menu display, log off QAD .NET UI, and then log in
 - Run it from the menu

Summary

Summary

- Report Viewer
- Report Render Engine
- Report Data Sources
- Report Layout Definition
- Report Resource Maintenance
- Template Designer
- Report Export/Import and Development Process
- Browse-to-Report Feature

Exercise Steps

Create a new Sales Order report based on a browse data source

You can use an existing browse as a source of data for a report. For example, you could use the Sales Order Browse. To see this browse, in the QAD .NET UI's left navigation panel, enter Sales Order Browse. To see where it is in the menu structure, right-click on it and choose Find In Menu. (Note that it is located under the Sales Order Menu - knowing this will come handy later when we add a new report to the menu system.) Double-click on Sales Order Browse to run the browse.

Specify the report resource

First, most likely someone will tell you the description of a browse (for example, Sales Order Browse), but in order to specify the browse as a report resource (that is, as a data source for the report), you will need to know the name (or, Browse ID) of the browse. Here are two ways to do that:

- In the QAD .NET UI's left navigation panel, locate Sales Order Browse. Right-click on it and choose Properties. Note the Aliases. The first one is the browses numbered location in the menu system, and the second is program (.p) name of the browse, such as sobr009.p. The browse name is the program name with the "br" and the ".p" removed: so009.
- Open Browse Master Browse (as its name suggests, this is a browse of browses). Under Search, on the first pull-down, choose "Description" (default is "Name"), choose "contains", and then enter "Sales Orders". Hit the Enter key or click Search. Note that the Name of "Sales Orders" is so009.

Once you know the name of the browse, you can specify it as a report resource using Report Resource Maintenance:

- 1 Open Report Resource Maintenance.
- 2 Enter a Report Code that you want to use for this new report. For example, ftp_report_01.
- 3 In the Report Resource frame, set Category to Report.
- 4 Set Data Source Type to Browse.
- 5 Set Data Source Ref to the browse name:so009 (so009 identifies Sales Order Browse).
- 6 Optionally, enter something in Description (such as "FTP Sales Order Report").
- 7 Click Next.

Now you are done specifying the report resource. (Report Resource Maintenance goes back to its original display, inviting you to enter another Report Code, but you are done with using Report Resource Maintenance for now.)

Create the report definition

You can now design the report:

- 1 Open Report Resource Designer.
- 2 Click New.
- 3 Select the report resource you have just defined: for example, select ftp_report_01.
- 4 Click Next.
- 5 Select a template. For example, select QAD_Default_Template_Browse.
- 6 Select the data source. (Under Tables, note that GetBrowseData_tt is selected so you have access to all the records accessed by Sales Order Browse.) Click Next.
- 7 Select the layout. For instance, set Orientation to Landscape and Layout to Columns. (Note that “Adjust fields to fit page” checkbox.) Click Next.
- 8 Select the field(s). For instance, select sales order number (so_mstr_so_nbr). Press the Ctrl key to select multiple items. Click > to move the selected fields into the Details box.
- 9 Click Next.
- 10 A summary of the new report is displayed, identifying the Report Template, Report Data Source, and Layout.
- 11 Click Finish.
- 12 A WYSIWYGish display of the report layout is displayed.
- 13 Click Save As. Save the report definition as ftp_report_01_definition_A.

You could edit the design now but first let's make sure what we have now actually works.

Test run from Report Resource Designer

- 1 In the Report Resource Designer, click Preview. The report Filter / Viewer is displayed.
- 2 Note the drop-down just to the left of the Run button. From the pull-down, select Document. Selecting Document will have the report display on the QAD NET UI Screen, in the Viewer tab. Note there are a variety of other output format options (PDF, Excel, etc.).
- 3 Click Run.

Export the report to XML

- 1 Open Report Resource Export.
- 2 Initially, the screen is mostly blank, but we can change that. At the bottom of the screen, in From Report, enter ftp_report_ (we are going to search for all the reports, starting with reports named “ftp_report_”).
- 3 Click Search.

- 4 Now lots of reports are listed, start with “ftp_report” and all the reports after it (most likely, a lot of reports starting with “QAD_”). We just want the ftp_reports, though. You could Uncheck All and then just select the ones you really want to export. You could also enter something in the To Report field to focus the search. Trying entering “g” in the To Report field, with “ftp_report” still in the From Report field.
- 5 Ok, that's enough fun with that. Just select the report you've created (for instance, “ftp_report_01”). Specify the Export Directory. You can select some location on your machine (click the “...” button).
- 6 Click Export.
- 7 The Status column should indicate Complete. (If there were problems, an error message would be in the Error Message column.)

Add the report to the menu

Let's add the new report (ftp_report_01, for instance) to the menu system. As this report is related to Sales Orders, let's put it under the Sales Order Menu. You might recall that the Sales Order Browse is also under the Sales Order Menu.

The QAD Enterprise Applications menu system has a numbered hierarchy. These numbers are called menu keys. You don't need to know everything about this menu system right now, but you do need to know the menu key for the Sales Order Menu so that you can put new items under it (such as your new report).

Find the Sales Order Menu in the menu system, right-click on it, and choose Properties. In the Properties window, find the Key field (yes, this is the menu key). Most likely, it is 7.1.

The Sales Order Menu typically has about 24 menu items under it. Each one has a menu key 7.1.*. For instance, Sales Order Maintenance is typically 7.1.1. and Sales Order Browse is typically 7.1.2.

Let's give our new menu item a menu key of 7.1.50. This will stay out of the way of the existing menu items but also give us some room in case we want to add other Sales Order related reports (for example, we might want to use 7.1.51, 7.1.52, etc.).

Add to menu system

- 1 Open Menu System Maintenance.
- 2 Set Language ID to US.
- 3 Click Next.
- 4 Set Menu to 7.1
- 5 Set Selection to 50 (this is how we specify a new menu item identified as 7.1.50).
- 6 Click Next.
- 7 Set Label to FTP Sales Order Report.
- 8 (Leave Name Blank)

- 9 Set Exec Procedure to: urn:qad-report:c1:ReportCode, for instance: urn:qad-report:c1:ftp_report_01

Set menu system permissions

- 1 Open Role Permissions Maintain
- 2 In Role Name, select a role that your user ID is a member of - for instance, select SuperUser. Right-click and choose Permissions.
- 3 You'll see "Secured items on menu" as an open/collapse item. Currently the check mark appears grey rather than black, indicating there are some items on the menu that are not currently accessible.
- 4 Click on the + button next to "Secured items on menu."
- 5 Click on the + button next to Customer Management, then Sales Orders / Invoices (7), then Sales Order Menu (7.1.).
- 6 Scroll down as needed to locate the menu item you have added. For instance: FTP Sales Order Report (7.1.50).
- 7 Click the checkbox on.
- 8 Click Save.
- 9 To refresh the menu system display, log off the QAD .NET UI and then log in again.
- 10 FTP Sales Order Report should now be a menu item under Sales Order Menu. You can now run the report from the menu system.

Note In Enterprise Edition (EE), you set permissions with Role Permissions Maintain. In Standard Edition (SE), you would use Menu Security Maintenance instead.

Chapter 2

Development Using Designer

Session Objectives

Session Objectives

You will learn how to:

- Create and modify report page layouts using the Report Resource Designer
- Use sub-reports to work with multi-table data sources
- Add VBScript logic for dynamic layout modification
- Use and create templates, allowing standardization and mass changes across reports



RF-DES-020

Report Designer Program

Report Designer Program

- Graphical drag-and-drop tool for editing report page layout definitions
- Only available to users belonging to the **rptDsgn** or **rptAdmin** roles (EE) or groups (SE)
- Two ways to invoke the program:
 - Report Resource Designer menu item
 - Right-click on any report menu item and click Design



RF-DES-030

Use one of the following ways to access Report Designer:

- Type Report Resource Designer in the menu search field and press Enter.
- If you have created a menu item for your report, locate it in the Applications Pane and right-click it; then choose Design from the shortcut menu. The Report Designer window appears.

Before you can access Report Designer to create a report definition, you must first create a valid report resource.

Using Report Resource Designer



In the Report Designer, you can create, load, and delete report definition files, as well as edit them either in the WYSIWYG (What You See Is What You Get) or code edit mode.

Creating a New Report Definition: Click the New button on the Report Definition toolbar. Report Wizard takes you through the process of creating a basic report definition. Click the Save icon on the toolbar to save the definition as an XML file.

Loading an Existing Report Definition File: Click the Open button on the toolbar; then in the Select Report Definition window, double-click the report definition you want to load. The Select Report Definition window also let you enter search conditions to search for the report definition you want to load.

Toolbar Functions

Toolbar Functions

- Goto
 - Report Resource Maintenance
- Actions
 - Export Metadata
 - Enable Data Import/Export
- New, Open, Close, Save, Save As, Delete
- Manager
- Preview
- Validation
- Edit Report Definition File
- About



RF-DES-050

New: Launch Report Wizard to create a new report definition.

Open. Open an existing definition.

Close. Close the current report definition.

Save. Save current report definition.

Save As. Save current report definition under a different name.

Manager Form (Report Definition Manager Form). Click the Manager button on the toolbar.

To delete an existing report definition: In the Manager window, click the report definition and then click the Delete button at the top. Confirm the deletion when prompted.

To select a different template for a report definition: Click the current template next to the report definition and select a different template from the list. The layout and formatting of the report definition will be changed after you assign a different template to it.

To set the default report definition for the report resource: Select the Is Default check box for the report definition. You must set one and only one default report definition.

When you open the report resource from the Applications menu tree, the default report definition is loaded.

Click Preview to display the current report definition in preview mode. In the preview mode, you can only navigate through the generated report.

You can export report metadata to an XML file for review and debug a report design. On the toolbar, choose Actions and click Export Metadata.

In the Save As dialog box, specify the name of the XML file and where you want to save the file. By default, the file is named after the current report with `_meta.xml` appended.

You can export the data for a report to an XML file for review, testing, and debugging purposes. You can then import an XML file containing report data and run it. The exporting and importing actions will not take place until a preview of the report is run from the Report Designer.

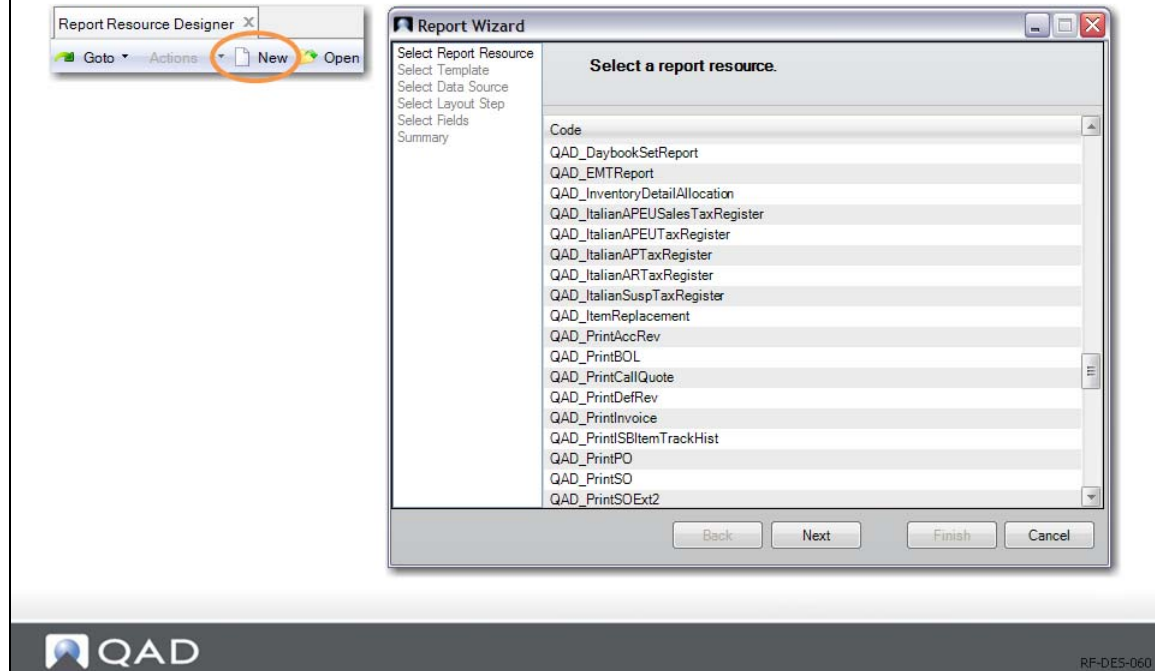
Importing report data from an XML file is a useful tool for scenarios such as:

- Repeated testing of a report whose data source takes a long time to run.
- Manually changing data values for the purpose of quickly testing report design logic, without having to change them in the live production database.

Using Report Wizard

Using Report Wizard

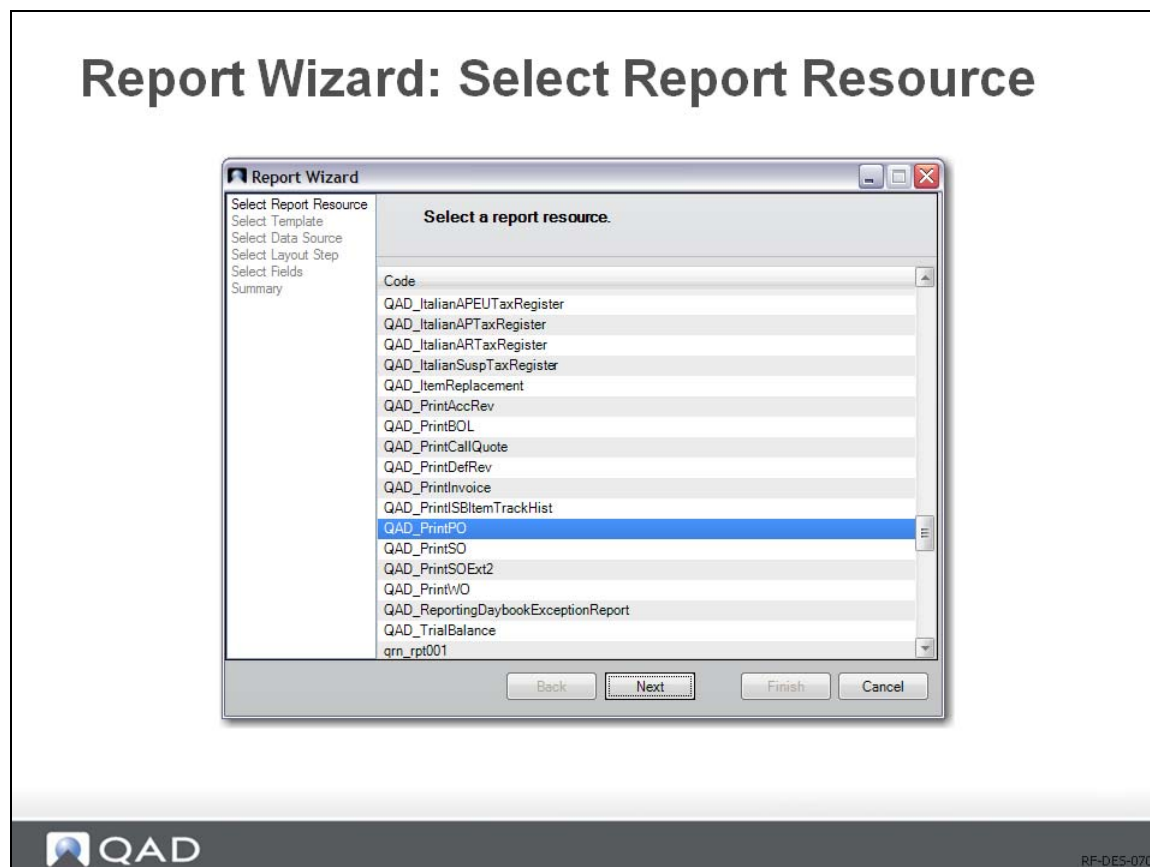
- Click New to run the New Report Wizard...



Report Wizard provides you with step-by-step instructions to build a basic report. Built-in report templates take care of most of the reporting layout and formatting for you.

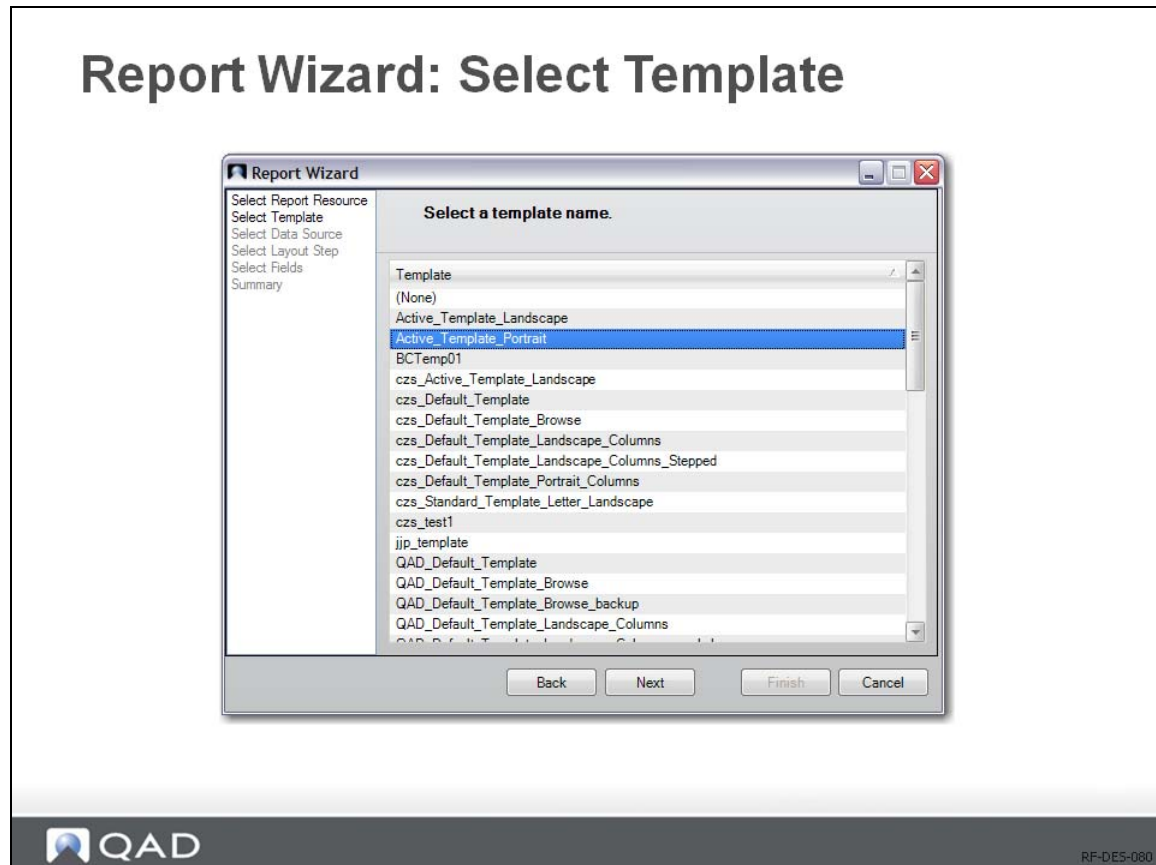
First, click New to launch the Report Wizard.

Report Wizard: Select Report Resource



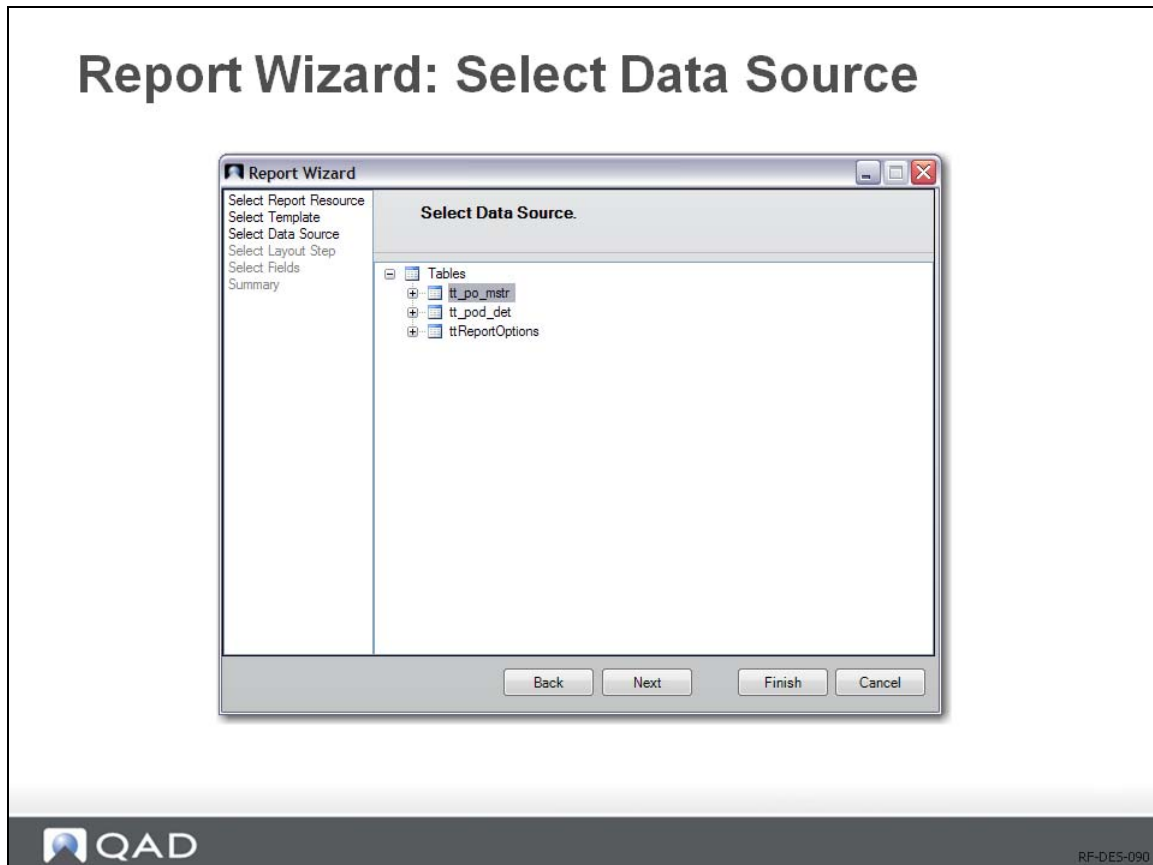
Select a report resource and click Next.

Report Wizard: Select Template



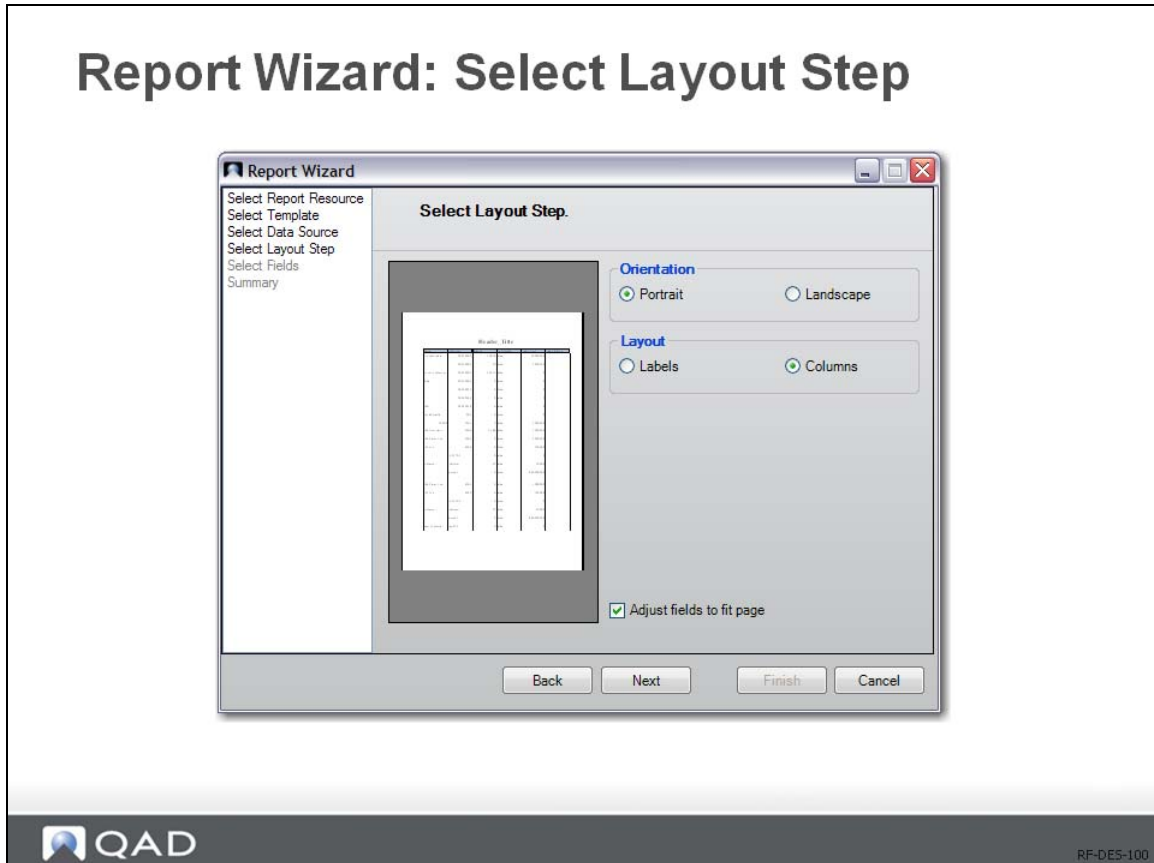
Select a template name such as `Active_Template_Portrait` and click Next.

Report Wizard: Select Data Source



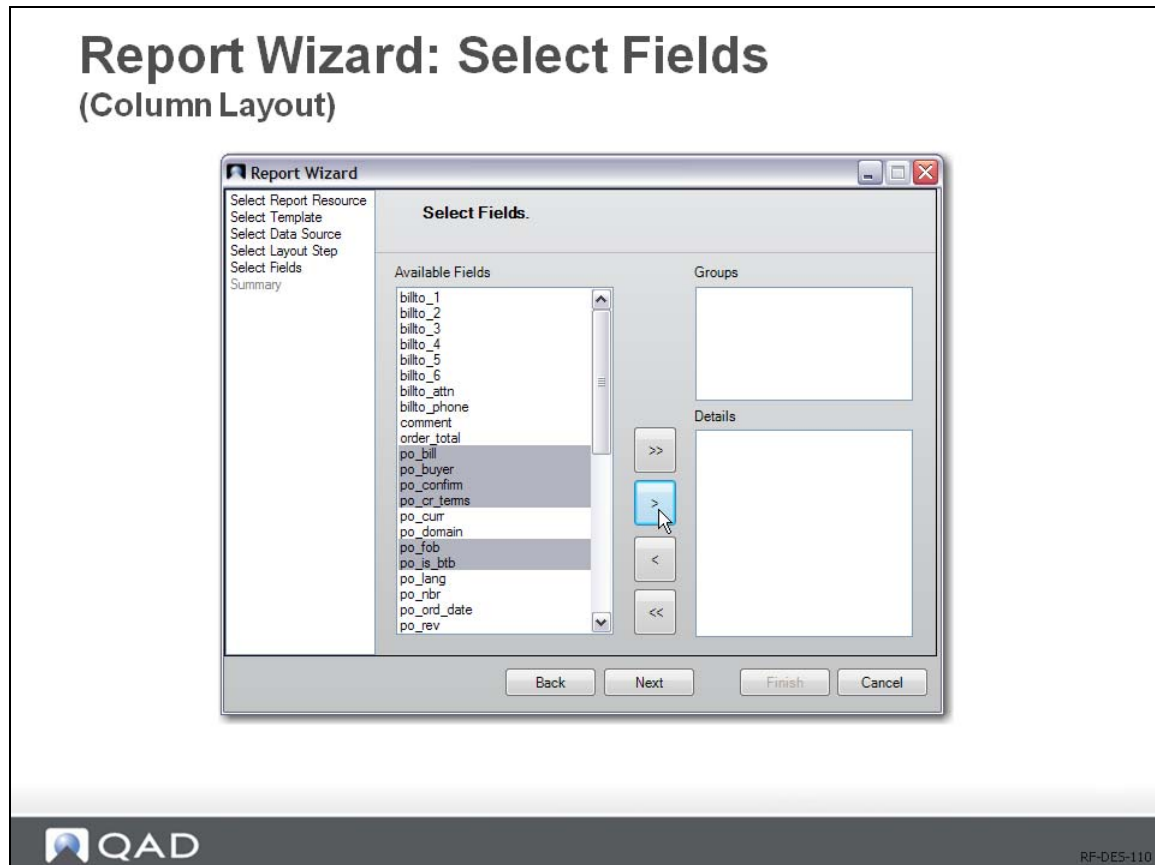
Select a data source and click Next.

Report Wizard: Select Layout Step



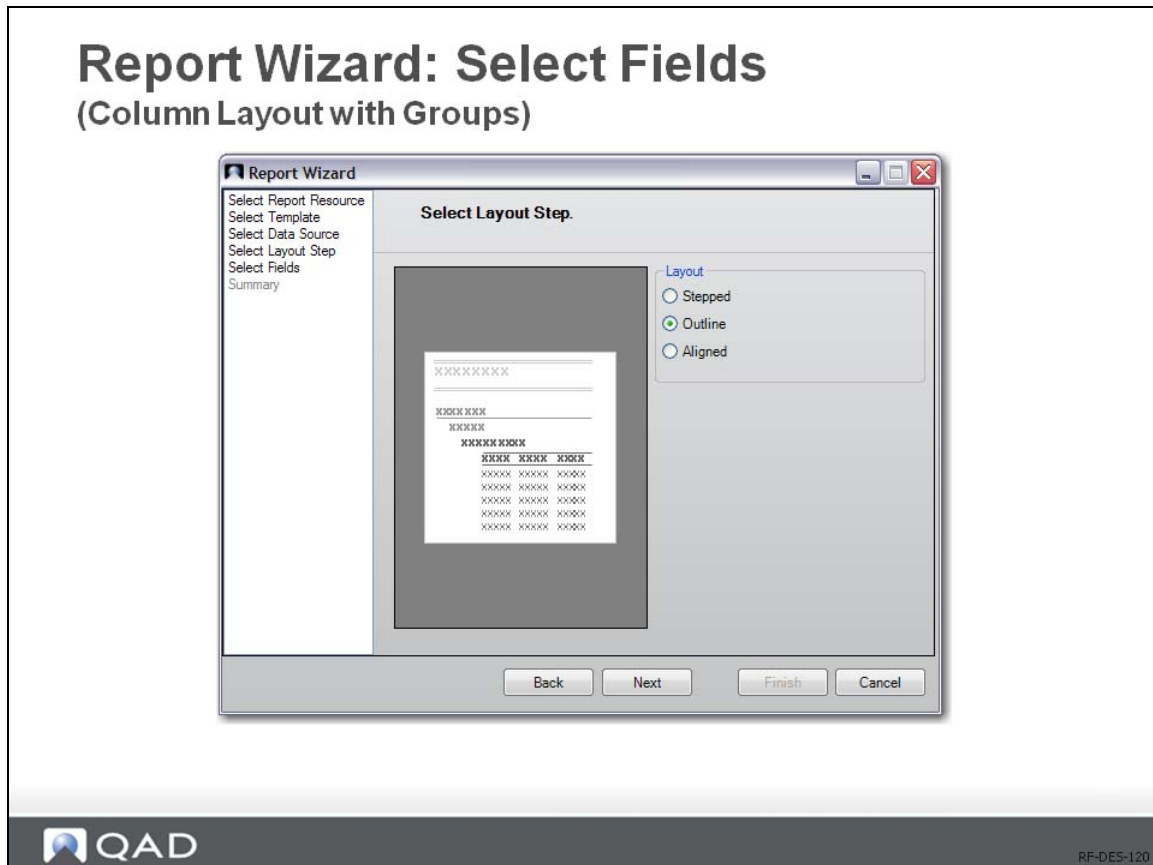
In the Select Layout Step, you can choose your report orientation as either Portrait or Landscape. Additionally, you can choose the report layout as Labels or Columns. After you click Next, the next screen will vary depending on whether you have chosen Labels or Columns.

Report Wizard: Select Fields (Column Layout)



For Column layout, you select the fields you want to use from the available fields of your data source. Optionally, you can organize the columns into groupings. This feature is similar to the “group by” display feature in browses.

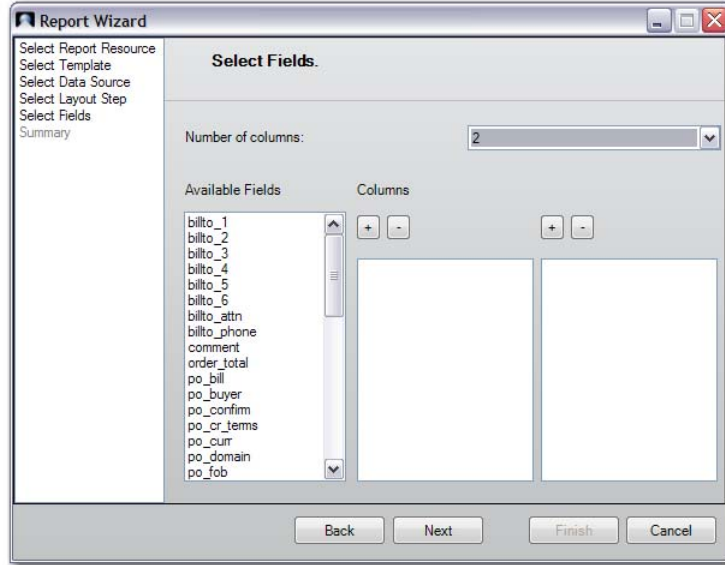
Report Wizard: Select Fields (Column Layout, Groups)



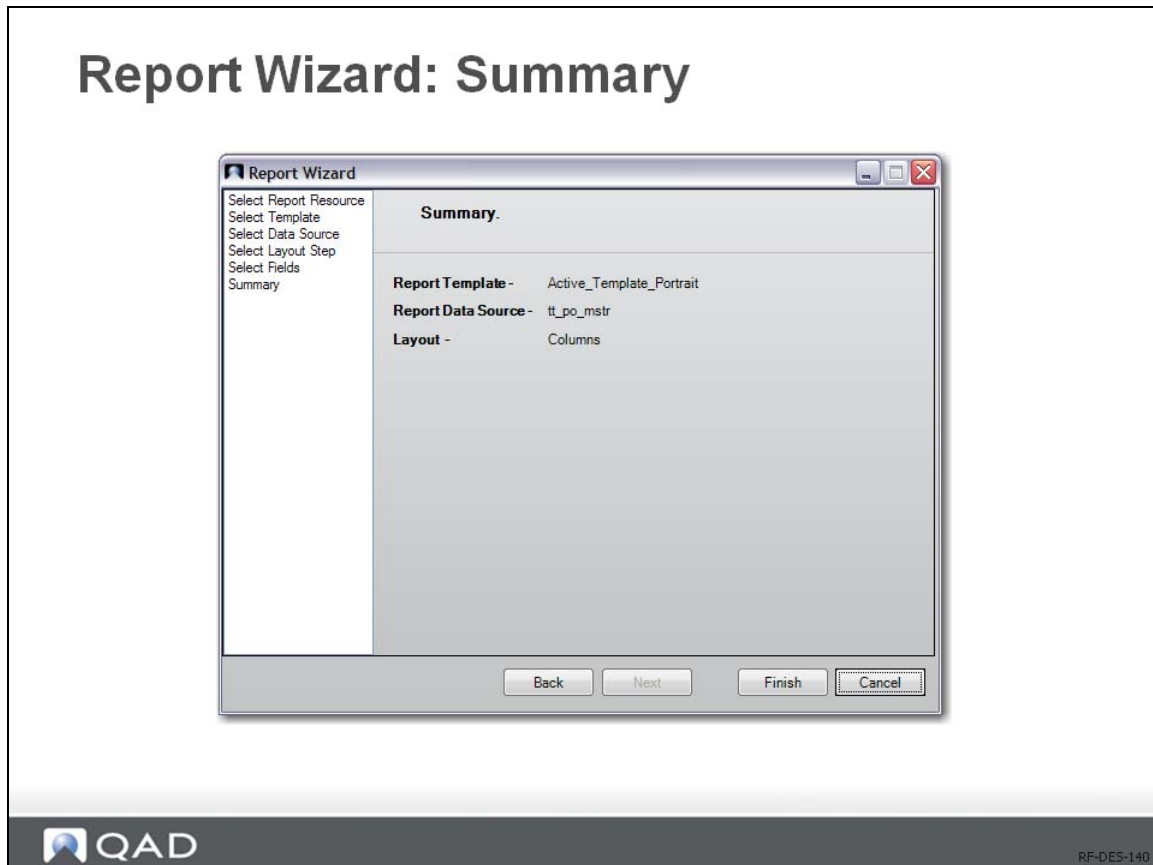
If you have chosen column layout and specified groups, you next select the type of layout you would like as Stepped, Outline, or Aligned.

Report Wizard: Select Fields (Label Layout)

Report Wizard: Select Fields (Label Layout)



Report Wizard: Summary



Upon completion, the Report Wizard's summary screen displays information about the template, data source, and layout. Finally, Click Finish.

Manually Working With Layout

Manually Working With Layout

- General Concepts
 - Sections
 - Fields
- Primary Tool Bar
- Secondary Tool Bar
 - Sub-sections for each tool



RF-DES-150

Manually Working With Layout

Manually Working With Layout

- Tool Box
 - Reports Tab
 - Create Subreport
 - Rename
 - Data Source
 - Sorting and Grouping
 - Data Tab
 - Fields
 - Parameters
 - Settings
 - Controls Tab
 - Sub-sections for each type of control
- Properties Pane

General Layout Concepts

General Layout Concepts

- Sections
 - Header: Rendered once at beginning of report
 - Page Header: Rendered at beginning of each page
 - Group Header: Rendered at start of each record group
 - Detail: Rendered once per data record
 - Group Footer: Rendered at end of each record group
 - PageFooter: Rendered at end of each page
 - Footer: Rendered once at end of report

- Fields



RF-DES-170

Fields: Report fields can be added to any section. Fields can be used for many different types of information to display:

- Can be tied to data fields in the underlying data source
- Can be tied to filter parameters or other report settings
- Can be used to display labels or other static text
- Can be used for calculated expressions
- Can be used to display images, or build simple graphics (lines, rectangles)
- Can be used to create charts
- Can be used for bar codes

Primary Tool Bar

Primary Tool Bar



- Goto
- Actions: Export Metadata, Enable Data Import/Export
- New, Open, Close, Save, Save As
- Delete
- Manager
- Preview
- Validation
- Edit Report Definition File
- About


RF-DES-180

Goto. Go to Report Resource Maintenance.

Actions. Menu of options for exporting and importing metadata and data for testing and debugging purposes.

New. Launch Report Wizard to create a new report definition.

Open. Launch Report Definition Manager to open an existing report definition.

Close. Close the current report definition.

Save. Save the current report definition.

Save As. Save the current report definition as another one.

Delete. Delete the current report definition.

Manager. Launch Report Definition Manager to delete existing report definitions, set the default report definition, and modify some of their attributes.

Preview. Display the current report definition in preview mode. In the preview mode, you can only navigate through the generated report.


Validate. Check the validity of the current report definition file. If errors are found, error messages will be displayed.

Edit Report Definition File. Open the current report definition file in code mode for editing.


About. Display Report Designer version information.

Secondary Tool Bar

Secondary Tool Bar



- Secondary tool bar includes editing functions
- Cut, Copy, Paste
- Undo, Redo
- Show Grid, Grid Setting
- Brush
- Bring To Front, Send To Back
- Multiple Object Positioning Tools...


RF-DES-190

Cut. Cut the selected objects on the report.

Copy. Copy the selected objects on the report to the clipboard.

Paste. Paste cut or copied objects from the clipboard to the currently selected area on the report.

Undo. Undo any actions you have performed on the report.

Redo. Redo the actions you have undone.

Show Grid. Toggle background grid on and off.

Grid Settings. Configure grid settings such as grid units and grid spacing.

Brush. When multiple objects are selected, apply the format of the last selected object to all other selected objects.

Bring to Front. Bring the selected object to the foreground.

Send to Back. Send the selected object to the background.

The “Multiple Object Positioning Tools” only apply to multiple objects on the report and will be grayed if only one object is selected:

Table Fields. Merge selected objects into table fields.

Align Left. Align multiple selected objects to the left boundary of the last selected object.

Align Center. Horizontally align multiple selected objects to the center of the last selected object.

Align Right. Align multiple selected objects to the right boundary of the last selected object.

Center Horizontally on Section. Horizontally position selected objects to the center of the section.

Align Top. Align multiple selected objects to the top boundary of the last selected object.

Align Middle. Vertically align multiple selected objects to the middle of the last selected object.

Center Vertically on Section. Vertically position selected objects to the center of the section.

Align Bottom. Align multiple selected objects to the bottom boundary of the last selected object.

Equal Height. Resize multiple selected objects to the same height.

Equal Width. Resize multiple selected objects to the same width.

Equal Size. Resize multiple selected objects to the same height and width.

Equal Horizontal Spacing. Reposition multiple selected objects so that they are equally spaced out horizontally.

Decrease Horizontal Spacing. Horizontally reduce spacing between multiple selected objects.

Increase Horizontal Spacing. Horizontally increment spacing between multiple selected objects.

Equal Vertical Spacing. Reposition multiple selected objects so that they are equally spaced out vertically.

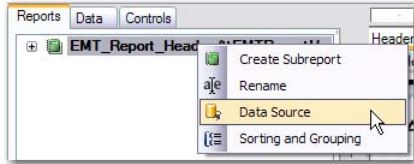
Decrease Vertical Spacing. Vertically reduce spacing between multiple selected objects.

Increase Vertical Spacing. Vertically increment spacing between multiple selected objects.

Reports Tab

Reports Tab

- Right click...



- View / Edit name of report
- Data Source
- Sorting and Grouping
- Create Sub reports (more details later)



RF-DES-200

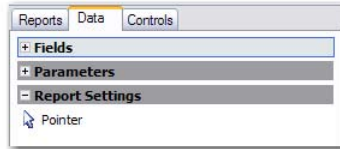
The Reports tab displays the report definition you are designing as well as all subordinate sub-reports embedded in the current report.

Right-clicking on the current report definition in this window brings up a shortcut menu that gives you access to a number of report design functions.

Data Tab

Data Tab

- Used to choose type of data field to create:
 - Fields: List of available fields from data source
 - Parameters: List of filter parameter values
 - Report Settings: List of system-supplied report settings



- Usage
 1. Click desired data field type in the tab list
 2. Move cursor to desired section
 3. Click and drag to create rectangle defining field position



RF-DES-210

The Data tab contains three groups of data: fields, parameters, and report settings. Except for the Pointer button, which is used to deselect any currently selected object, each button under these groups creates a field on the report and initializes its properties.

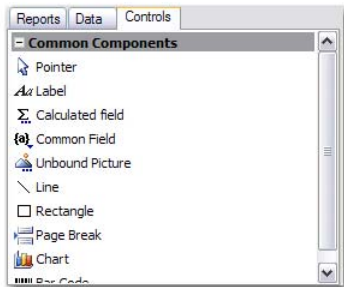
The Fields group contains all the available fields that are bound to the source record set.

The Parameters group contains all the available search condition parameters under the Filter tab in Report Viewer.

The Report Settings group contains all the available setting variables under the Settings tab in Report Viewer.

Controls Tab

Controls Tab



- Used to choose type of field control to create
- Usage
 1. Click desired field control type in the tab list
 2. Move cursor to desired section
 3. Click and drag to create rectangle defining field position



RF-DES-220

Pointer. Cancels any previous field selection to get back to the pointer tool.

Label. Static text (with provisions for linking to translated labels).

Calculated Field. Contains a VBScript expression.

Common Field. Several types of pre-defined fields.

Unbound Picture. Used to display an image from a file.

Note Image bits get embedded into report design XML.

Line, Rectangle. Simple geometric shapes.

Page Break. Inserts page break for every N rows of data.

Chart. Create simple charts tied to data source.

Bar Code. Create bar code tied to any data field.

Properties Pane

Properties Pane

- Displays properties for selected object
- Select by highlighting in Main Designer Pane (report, section, field)
- Select using drop-down box above the grid



RF-DES-230

Once an object is selected, you can use the Properties window to edit its properties.

When one or more fields are selected, the Properties window shows property values that all fields have in common, and leaves the other properties blank. If no fields are selected and you click a section (or on the bar above a section), the Section properties are displayed. If you click the gray area in the background, the Report properties are displayed.

To see how this works, click the label in the Header section and change its Font and ForeColor properties. You can also change a field's position and dimensions by typing new values for the Left, Top, Width, and Height properties.

The Properties window expresses all measurements in twips (the native unit used by Report Designer), but you can type in values in other units (in, cm, mm, pix, pt) and they will be automatically converted into twips. For example, if you set the field's Height property to 0.5 inches, the Properties window will convert it into 720 twips.

A twip (derived from TWentieth of an Imperial Point) is a typographical measurement, defined as 1/20 of a typographical point. One twip is 1/1440 inch or 17.639 μm when derived from the PostScript point at 72 to the inch, and 1/1445.4 inch or 17.573 μm based on the printer's point at 72.27 to the inch.

Properties Pane

Properties Pane

- Report-Level Properties
 - Paper Size, Margins, Width, and the Ruler
 - Check / set these before working on report
 - Can be inherited from templates
- Section-Level Properties
 - Section sizing in panel
- Field-Level Properties

Using Properties Pane

Using Properties Pane

Object dropdown list

The screenshot displays the QAD Reporting Framework interface. On the left, the Properties Pane is open to the 'PageHeader' object. The pane is organized into sections: Appearance, Design, Layout, Script, and Template. The Design section is expanded, showing properties such as (Name) set to 'PageHeader' and Visible set to 'True'. The main window shows a report layout with a header containing the QAD logo and the title '\$(EMT_NETWORK_REPORT)'. Below the header is a table with columns: \$(ITEM_NU), \$(DESCRIPTION), \$(SITE), \$(SUPPLIER), \$(NAME), and \$(THIRD_PA). The table has rows for 'pt_part', 'pt_desc1', 'si_site', 'vd_addr', 'vd_name', and 'pt_desc2'. A subreport titled 'Subreport: EMT_Report_Detail' is also visible, showing columns for trading and document information.

The various Section, Report, and Field properties are described in detail in the *Reporting Framework User Guide*.

Translated Labels

Translated Labels

- Localize reports into multiple languages
- Label terms in report design read from system's label master table
- Label terms listed in Label Master Browse
- Labels can include various formats:
 - Long label
 - Medium label
 - Short label
 - Stacked label



RF-DES-260

You can design reports that can be readily localized into multiple languages. When localized, the system dynamically reads the label master table to determine the appropriate labels to display on your reports. This table contains translated labels that can be specified with a string that specifies the required label (the label term key, for example "SALES_ORDER"). A report design can use these label term keys so that when the report is run, the system will retrieve the translated label in the appropriate language for the user.

If you attempt to translate a label that does not exist in the label master, you should create a record for it using Label Master Maintenance (36.4.17.24) so that the label can be translated and maintained.

For Label report fields, the value of the Text property will typically be literally displayed in the field when the report is rendered. However, the system can be instructed to automatically select translated labels when the report is rendered. This mode of operation is triggered when the Text property contains a value with the special $\${LABEL_TERM}$ format, which instructs the system to look up the translated label corresponding to the LABEL_TERM used.

System labels can have short, medium, and long versions, of which the system intelligently selects the largest that will fit into the report field, taking into account factors such as field width and font size. For example, if the field width is not large enough to display the large label for the given text in the given font, the medium label will be displayed automatically if it fits.

Note that the Reporting Framework is flexible about the format of the label terms. For example, you can enter the term for Sales Order as `${SALES_ORDER}` or `${Sales Order}`. The system will replace spaces with underscores and perform case-sensitive matching at run-time.

There are also some variations on this format that allow you to explicitly specify which label should be used:

- `${TERM}S` — specifies the short label
- `${TERM}M` — specifies the medium label
- `${TERM}L` — specifies the long label
- `${TERM}!` — specifies a stacked label

Note that in the case of stacked labels, the height of the field must be large enough to fit all the levels in the stacked label or the display of the label will be truncated.

For Calculated fields, the value of the Text property is evaluated as a VBScript expression and the result will be displayed in the field when the report is rendered. One special case is when the expression consists of the name of a report data field, in which case the value of that data field will be displayed when the report is rendered.

For Calculated fields, you can also specify to have integers converted into the words for the number. For instance, you can have “12” displayed as “twelve.” The format is as follows:

- `${TERM}N` — specifies integer to words conversion

Smart Label Selection

Smart Label Selection

- Triggered by entry in a label-type field
- For system label terms:
 - `#{TERM}` - displays longest label that fits
 - `#{TERM}S` - displays short label
 - `#{TERM}M` - displays medium label
 - `#{TERM}L` - displays long label
 - `#{TERM}!` - displays stacked label

Number-to-Word Translator

Number-to-Word Translator

- Triggered by special entry in a calculated report field:
 - `${DataFieldName}N`
 - Data field must be an integer
- Example
 - Data field contains value of 273
 - Report output will display “two hundred seventy three”
- Uses language of the .NET UI user
 - Limitations: no support for Chinese, Polish or Korean. Japanese does not use actual symbols (words are spelled out in English letters)

Exercise

Exercise

1. Create a second layout definition for Sales Order report created in the previous session:
 - Group data by sales order number
 - Structure layout to have a master-detail page appearance
 - Header fields listed in Group Header
 - Detail (line) fields indented below header fields
2. Make the new layout the default (so that it executes from the menu)
 - Manager tool from Designer program
3. Export the report to XML
 - Report Resource Export

Sub-Reports

Sub-Reports

- Data Source Considerations
- One report can only access one data table
- If there are multiple data tables in the data source, then multiple reports are required – use Sub-Reports



RF-DES-300

Sub-reports are regular reports contained in a field in another report (the main report). Sub-reports are usually designed to display detail information based on a current value in the main report, in a master-detail scenario. You can create multiple subreports in a main report.

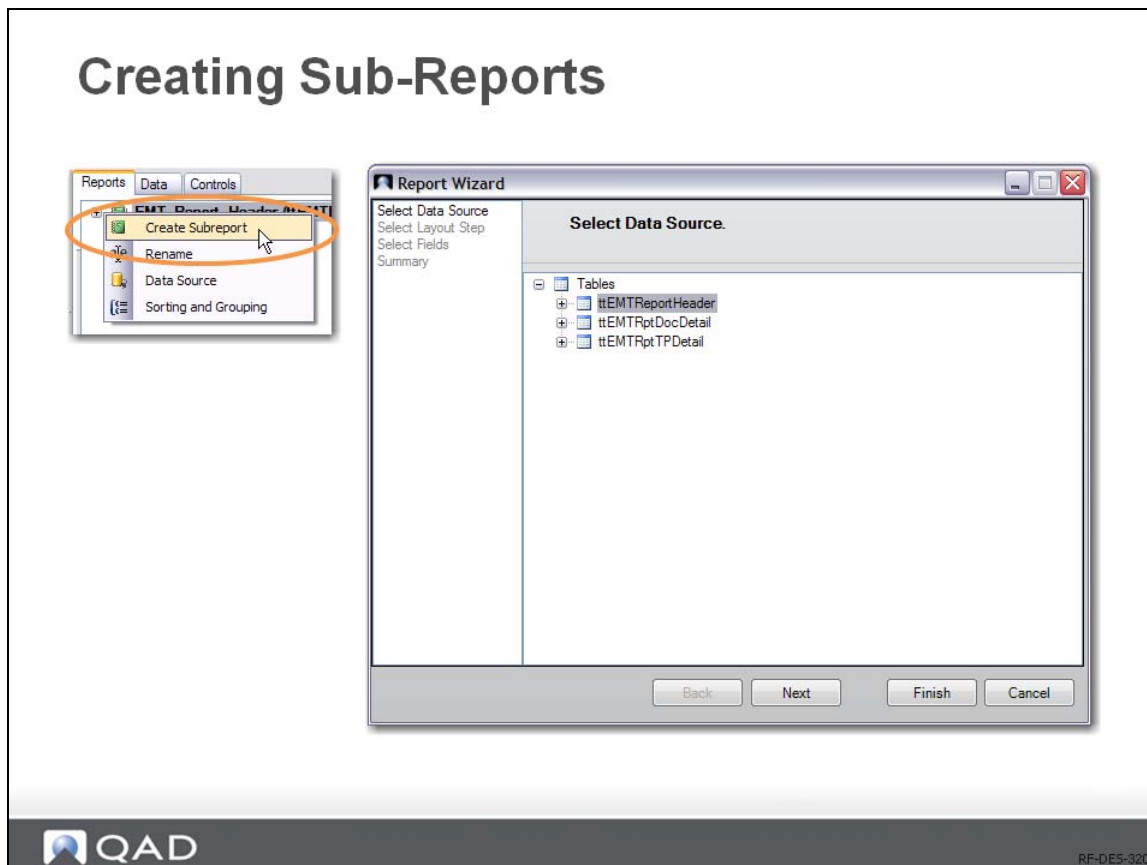
For example, the main report contains product categories and the sub-report in the Detail section contains product details for the current category.

Sub-Reports

Sub-Reports

- How to Create
 - Right-click on report in Reports tab – select “Create Subreport”
 - New report Wizard launches
 - After Wizard, click and drag mouse on the desired section to create the sub-report field
 - Link the data fields of parent and child tables
 - Right-click the sub-report field and choose “Link Subreport”
 - Edit sub-report as desired
 - Right-click the sub-report field and choose “Edit Subreport”
 - Or, double-click the sub-report in the Reports tab

Creating Sub-Reports



In the Reports tab, choose Create Subreport and then from the Report Wizard, select the data source.

Using Sub-Reports: Tips

Using Sub-Reports: Tips

- Usually placed in detail section of parent report
 - Sub-report will then run once per record
 - Sub-report data rows will be subset of rows in sub-report's data table
 - Filtered by link condition
- Generally avoid use of Page Header/PageFooter sections in sub-report
- VBScript variables defined in parent report can be visible in child sub-reports
 - Only if the parent's ExposeScriptObjects property is set to true

Using Sub-Reports: Tips

Using Sub-Reports: Tips

- Nesting is allowed
 - Sub-reports can contain sub-reports
 - Reports can contain many sub-reports
 - Report Tree not limited in size
- Common error: forgetting to link sub-report
 - Results in a large number of records output in the report
 - Every record in the sub-report table displayed for each record in the parent table
 - Cartesian product)

Sub-Reports: Exercise

Sub-Reports: Exercise

1. Create a master-detail Sales Order report, using a sub-report with a 2-table data source
 - Report Resource Maintenance
 - Report Code: SalesOrderSubReport
 - Data source: Proxy- UserGuideSampleReport.p
 - Need to first install it into the system
 - Report Resource Designer
 - Use sub-report for detail layout
2. Export the report to XML
 - Report Resource Export

VBScript Logic

VBScript Logic

- Event Model
 - VBScript code blocks can be placed into event hooks at Report and Section levels.
 - VBScript code blocks will fire when corresponding event occurs during report rendering
 - Can access field and section properties
 - e.g. header.visible = false
 - e.g. soldto_attnLbl.ForeColor = rgb(255,0,0)



RF-DES-360

Section-level Script settings:

OnFormat. Enter a Visual Basic script that will be executed during the formatting stage of report rendering.

OnPrint. Enter a Visual Basic script that will be executed in the final stage of rendering, after all report field values have been filled.

Report-level Script settings:

OnClose. Enter a Visual Basic script that will be executed when the report finishes rendering.

OnError. Enter a Visual Basic script that will be executed when an error occurs.

OnNoData. Enter a Visual Basic script that will be executed when the report has no data.

OnOpen. Enter a Visual Basic script that will be executed when the report starts rendering.

OnPage. Enter a Visual Basic script that will be executed each time a new page is created.

Design Philosophy for using VBScript:

Less is more! Try to use as little as possible, since it can add to the cost of maintaining the report in the future. VBScript used too much can add a lot of complexity to the report layout design.

VBScript is often necessary for achieving certain dynamic layout goals, for example changing a file's color or visibility based on its data value.

Some reports might need to change the properties of many fields at a time based on data values, even moving their location dynamically. Instead of the brute-force approach of using conditional statements with lots of fine-grained field property setting VBScript statements, it is often preferable to instead use two or more sub-reports, each containing one of the different layout options, and using a single VBScript statement to conditionally determine which sub-report to make visible. This is a more maintainable approach that leads to report designs that are easier to read, understand, and enhance in the future.

Script Logic

VBScript Logic

- Use in Calculated Fields
 - Linking to data source fields
 - Built-in function finder, variables (Page, Pages,
- Design Philosophy
 - General rule: Use as little VBScript as needed
 - VBScript makes reports harder to maintain, and harder to convert to other rendering technologies
 - Dynamic Layout Options
 - VBScript to select sub-reports vs. VBScript to control fine-grained fields

VBScript Examples

VBScript Examples

- Filtering out data row based on some data value:
 - Put VBScript block in Detail section's OnFormat event:

```
if ucase(pt_mstr_pt_part_type) = "FINGOOD" then
    Detail.Visible = false
else
    Detail.Visible = true
endif
```

- Changing a field's color based on its data value:
 - Put VBScript block in Detail section's OnPrint event:

```
if (pt_price >= 100) then
    'color the field text RED
    [pt_priceCtl].ForeColor = "#FF0000"
else
    'color the field text BLACK
    [pt_priceCtl].ForeColor = "#000000"
endif
```

VBScript: Exercise

VBScript: Exercise

1. Perform the techniques for the two VBScript examples given previously.
 - Filtering out data row based on some data value
 - Changing a field's color based on its data value

VBScript: Exercise

VBScript: Exercise

2. Add summary calculation fields to sum / count field values across groups.
 - Use the grouped Sales Order report
 - sum the quantity ordered for each order
 - count the number of lines for each order
 - count the number of lines across ALL orders
 - Use Calculated Field type in the appropriate section
 - Use aggregate script functions (e.g. Sum()) on appropriate data field

Report Templates

Report Templates

- A template is a special kind of report that cannot be directly rendered
- Allows report attributes to be defined in one place (the template) and applied to many reports



RF-DES-410

A report template is special kind of report definition that cannot be rendered directly by itself, but instead can be used to control certain aspects of the rendering of other reports. When designing a report, a template can be specified (optionally) in which case the report can inherit many kinds of attributes from the template, such as field colors and fonts. If at a later time these attributes are changed in the template, those changes will be seen in every report that is using that template.

Any given report can inherit from at most one template, but a given template can be used to control any number of reports. Thus templates enable report developers to making changes in a single place (the template) which will have a mass effect on many reports. This is a powerful tool that can assist the report development process in many ways, such as reducing initial development time, enforcing common standards across reports, and quickly implementing future changes to these standards.

There are three general types of report properties that can be governed by templates:

- Top-level report properties (e.g. paper size, margins)
- Section properties (e.g. The back color of the PageHeader section)
- Field properties (e.g. The font and ForeColor of fields)

In addition to inheriting properties, reports can also inherit fields from a template. For instance, a template might contain fields in the page header that display date, time, domain, and a corporate logo. These fields will be automatically added to all reports using that template.

Templates can be edited in the Template Designer, which is very similar to the Report Designer program.

Elements in the report template represent classes or styles that can be applied to corresponding elements in a report definition based on a class-mapping relationship.

A field defined in the report template represents a field class identified by a unique class name. When the field class is applied to a field in a report definition, most of its properties are carried over to the field so that the field takes on the same formatting and layout.

The header, page header, page footer, footer, or a group section defined in the report template represents a section class identified by a unique class name. When the section class is applied to a section in a report definition, it is virtually copied over to the report definition complete with all the elements in it.

Report Templates

Report Templates

- Changes made to one template will affect all reports that use that template
- A report can reference at most one template
- QAD-shipped templates have a QAD_ prefix and can get overwritten in upgrades.
 - Should not be modified, but copies can be made and modified.




RF-DES-420


Report Parameter Maintenance

Report Parameter Maintenance


Report Code: QAD_CDocumentReportCheque
Param Name: sys_search_criteria_display
Extend:

Value Settings

Value Type: Constant 
Value: NONE

2nd Type: 
2nd Value:

Operator Allowed:
Operator Editable:
Operator:

 RF-DES-RPM_1

Adding Search Criteria Report Parameter

Adding Search Criteria Report Parameter

1. Open Report Parameter Maintenance.
2. In Report Code, choose the resource your report is using.
3. In Param Name, choose `sys_search_criteria_display`.
4. Check the Extend checkbox.
5. Continue to the Value Settings frame and set Value to Header, Footer, or None
6. Click Next through the remaining frames of the program to close it.
7. Run the report.



RF-DES-RPM_2

Template Behavior

Template Behavior

- Templates define “Classes” that can be applied to objects in a report definition.
- Section Classes: sections in the report will inherit:
 - Most properties of the section class
 - BackColor, Visible, CanGrow, CanShrink, ForcePageBreak, KeepTogether, Repeat, OnFormat, OnPrint
 - All fields in the section class
 - Only for Header, PageHeader, PageFooter, and Footer sections

Template Behavior

Template Behavior

- Field Classes: fields in the report will inherit:
 - Most properties of the field class
 - Properties depend on field type. For example: Font, BackColor, ForeColor, Height, ...
- Report-level properties can be inherited from templates

Template Designer

Template Designer

- Tool to edit templates
 - Accessible in QAD .NET UI
 - Requires user membership in rptAdmin or rptDsgn
- Similar to Report Resource Designer, but has some differences
 - Detail Section Behavior:
 - Fields in detail section do not get passed to reports
 - Detail fields are only used to define field classes
- Template Import/Export
 - Similar to report import/export
 - Templates can be serialized as XML

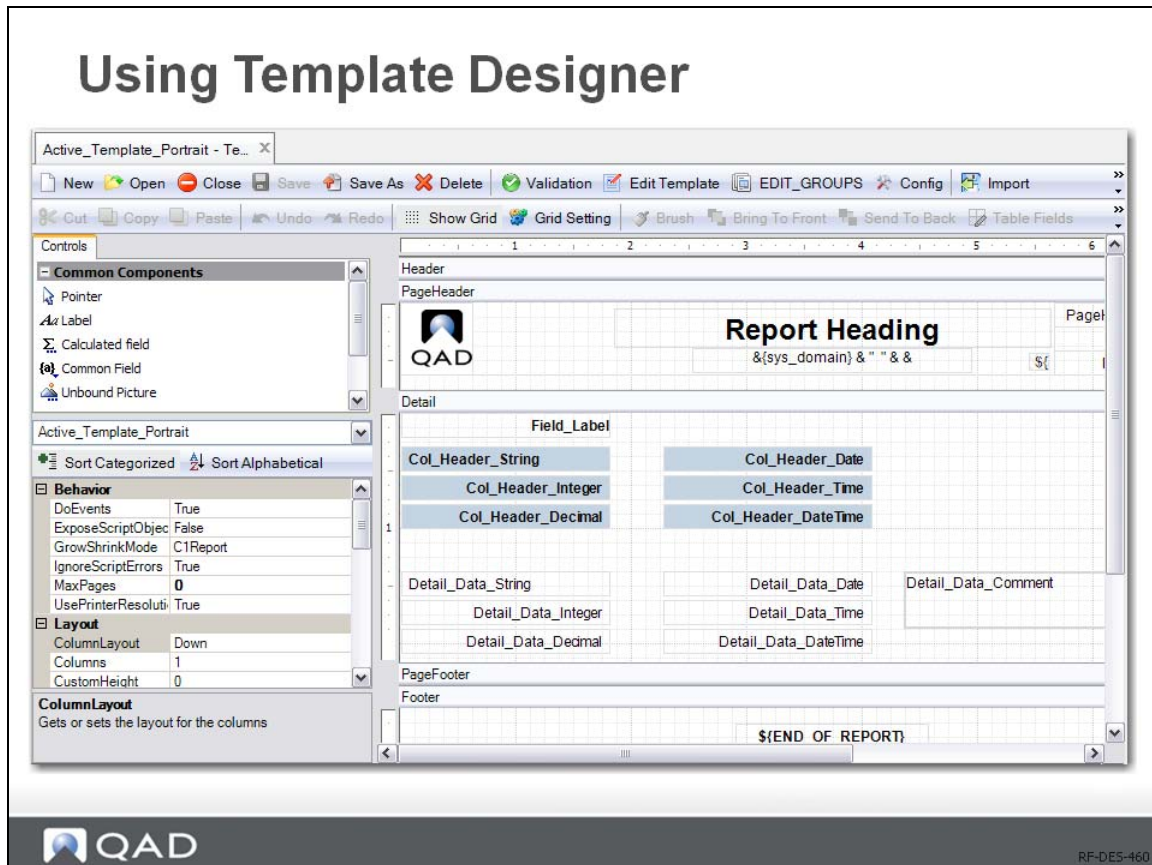


RF-DES-450

The Template Designer is almost identical to Report Designer, except for the following differences:

- The Toolbox only displays the Controls tab which contains a limited set of control components that can be used in the template: Label, Calculated Field, Common Field, Unbound Pictures, Line, Rectangle, and Chart.
- The toolbar contains the following buttons that are specific to Template Designer: Edit Template, Add Group, Config, Import, Export.

Using Template Designer



- 1 Launch Template Designer. Type Template Designer in the menu search field and press Enter.
- 2 Click the New button on the toolbar.
- 3 In the Create Template dialog box, enter a unique template name and click OK.

Important QAD-provided built-in reports, report resources, and templates all begin with “QAD_”. Do not create or modify reports, report resources or templates with this prefix. Otherwise, your customized changes will get overwritten during system upgrades from QAD.

- 4 In the Design pane, create and format field classes in the same way as you work with fields when working with a report definition. Provide unique class names for the classes.
- 5 If you want to define a header, page header, page footer, and footer section class name, click the default section name and enter a new name in the (name) field in the Properties pane.
- 6 If you want to add new sections, use the following steps:
 - a Click the New Group button on the toolbar.
 - b In the Edit Group dialog box, click Add and specify the properties for the new group.
 - c Click OK.

- 7** Configure the default section and field class mapping to specify the default classes to be applied to the corresponding sections and fields in the report definition.
 - a** Click the Configure button on the toolbar. The Class Configuration Form dialog box appears.
 - b** Under the Section Configuration tab, specify a class name for each section type.
 - c** Under the Field Configuration tab, for each data type, select a section in the template and specify a class defined within that section.
 - d** Click OK.
- 8** Back in the Template Designer main screen, click the Save button on the toolbar to save the template.

Applying Templates to Report Designs

Applying Templates to Report Designs

- When designing a report, there are two ways to specify a template:
 - Can assign a template when Wizard is run to create a new report definition (“New” tool button runs Wizard)
 - Can assign a template to an already-created report definition using “Manager” tool button



RF-DES-470

After you design a report template, you can apply it to multiple report definitions to enforce a consistent look and feel across all these reports.

Applying Templates to Report Designs

Applying Templates to Report Designs

- Class Property
 - Use at section and field levels to bind section/field to desired template section/field
 - New report Wizard auto-assigns many classes
 - Report-level properties automatically linked to template (2010 release and higher)



RF-DES-480

Class Property

Class Property

The screenshot displays the QAD Reporting Framework software interface. The main window shows a report design with a header, page header, group header, detail, group footer, page footer, and footer. The report content includes the QAD logo, a sales order number field, and a table with columns for sales order number, ship-to, and due date. The footer contains the text `$(END_OF_REPORT)`.

The left sidebar contains a tree view with the following structure:

- Reports
 - Unnamed (GetBrowseData_it)
- so_mstr_so_nbrLbl1
- Sort Categorized | Sort Alphabetical
- Layout
 - Anchor: Top
 - Bounds: 0, 0, 2910, 300
 - CanGrow: False
 - CanShrink: False
 - ForcePageBreak: None
 - KeepTogether: False
 - ZOrder: 0
- template
 - Class: Col_Header_String
- Class
 - Gets or sets the template.

The **template** section is highlighted with an orange circle, and a mouse cursor is pointing at the **Class** property, which is set to `Col_Header_String`.

The bottom of the interface features the QAD logo on the left and the text "RF-DES-490" on the right.

Overriding Template Attributes

Overriding Template Attributes

- Properties inherited from template can be manually modified
 - Modified property will be completely decoupled from template
 - Once decoupled, changes made to template will not affect decoupled property
 - **To re-couple template property:**
 - Click on property value in property grid
 - Click on “Use Default Value” button at top of property grid
 - Value will be refreshed from template
 - Property will be re-coupled to template

Overriding Template Attributes

Overriding Template Attributes

- Fields inherited from template section classes can be modified
 - Modified field will be completely decoupled from template
 - Once decoupled, changes made to template will not affect de-coupled field
 - To re-couple all template fields (and properties) for the section:
 - Change section's template class to None
 - Change it again to the original class
 - All template fields will be refreshed from template.

Template Development Guidelines

Template Development Guidelines

- When editing templates, be careful when saving since changes can affect many reports across the system
- QAD Active templates (2010 release and higher)



RF-DES-520

Starting with the QAD Enterprise Applications 2010 release, the QAD standard reports included with the product use one of the following templates:

- `Active_Template_Landscape` —the template typically used by QAD standard reports with landscape orientation.
- `Active_Template_Portrait` — the template typically used by QAD standard reports with portrait orientation.

Additionally, four QAD standard templates are included:

- `QAD_Standard_Template_A4_Landscape` — the QAD standard template for the A4 (210 x 297 mm, or 8.27 x 11.69 inches) paper size with landscape orientation.
- `QAD_Standard_Template_A4_Portrait` — the QAD standard template for the A4 paper size with portrait orientation.
- `QAD_Standard_Template_Letter_Landscape` — the QAD standard template for the Letter (8.5 x 11 inches) paper size with landscape orientation.
- `QAD_Standard_Template_Letter_Portrait` — the QAD standard template for the Letter paper size with portrait orientation.

As shipped, the contents of Active_Template_Landscape are identical to QAD_Standard_Template_Letter_Landscape, and the contents of Active_Template_Portrait are identical to QAD_Standard_Template_Letter_Portrait. If template customizations are required, they can be done in the Active templates (which the QAD reports reference), and the original QAD_ templates should be kept unmodified in case any changes might need to be rolled back.

Careful management of these templates is highly recommended.

Special Template for Browse Reports

Special Template for Browse Reports

- Template name:
QAD_Default_Template_Browse
- Used for reports created on-the-fly from running .NET UI Browsers
 - Any end user can invoke such a report from a browse using Actions > Report
- Customers can modify this template to tailor cosmetic appearances of browse reports
 - Back up any changes using export tool since this template will be overwritten in upgrades



RF-DES-530

Templates: Exercise

Templates: Exercise

1. Use Report Resource Designer to modify (or create) a report definition that does not have a template specified
 - Use the Manager tool to specify a template
 - Assign classes to sections and fields
 - Override some field properties for a data field
 - Override some field properties from a template-inherited field
2. Use New Report Wizard to specify a template and Create a new report definition
 - Examine the class assignments made by the Wizard, changing if desired

Templates: Exercise

Templates: Exercise

3. Modify the template used in exercise 1, using Template Designer:
 - Modify some field colors / fonts, including for overridden fields. Re-run the report and verify that changes only take effect for non-overridden fields.
 - Re-couple overridden fields to the template, and re-run the report.
 - Add a field to PageHeader section, and change BackColor of that section
 - Open report in the Designer and run
 - Is it as expected?
 - Remove template class for PageHeader section and re-assign it.
 - Does the report run differently now?

Summary

Summary

In this section you have learned how to:

- Create and modify report page layouts using the QAD Reporting Framework's Designer program
- Use sub-reports to work with multi-table data sources
- Add VBScript logic for dynamic layout modification
- Use and create templates, allowing standardization and mass changes across reports



RF-DES-560

Exercise Steps

Create a second layout definition for Sales Order report

To create a second layout definition for the sales order report created in the previous session, you want to:

- Group data by sales order number
- Structure layout to have a master-detail page appearance
- Header fields listed in Group Header
- Detail (line) fields indented below header fields

So you do the following:

- 1 Open Report Resource Designer.
- 2 Click New.
- 3 Select the report resource you defined in Session 1: for example, select ftp_report_01.
- 4 Click Next.
- 5 Select a template. For example, select QAD_Default_Template_Browse.
- 6 Select the data source. (Under Tables, note that GetBrowseData_tt is selected so you have access to all the records accessed by Sales Order Browse.)
- 7 Click Next.
- 8 Select the layout. For instance, set Orientation to Landscape and Layout to Columns. (Note that "Adjust fields to fit page" checkbox.) # # # Click Next.
- 9 Select the customer field (so_mstr_so_cust). Drag it to the Group box.
- 10 Select the remaining fields and move them into the Details box.
- 11 Click Next. A summary of the new report is displayed, identifying the Report Template, Report Data Source, and Layout.
- 12 Click Finish.
- 13 A WYSIWYGish display of the report layout is displayed.
- 14 Click Save As. Save the report definition as ftp_report_01_definition_B. (This is a new definition - the definition you created in Session 1 was ftp_report_01_definition_A.)
- 15 Test the report by clicking Preview and then Run.

Make a New Layout the Default

To make the new layout the default (so that it executes from the menu):

- 1 In Report Resource Designer, click Manager.
- 2 Change the selected Default Definition from ftp_report_01_definition_A to ftp_report_01_definition_B.
- 3 Now try running the report you put on the menu system in Session 1. The report should generate results based on the new definition (ftp_report_01_definition_B), which groups sales order data by customer.

Export the report to XML

- 1 Open Report Resource Export.
- 2 Initially, the screen is mostly blank. At the bottom of the screen, in From Report, enter ftp and in To Report, enter g. This will list all the reports that start with “ftp”.
- 3 Click Search.
- 4 Select the report you've created (for instance, “ftp_report_01”). Specify the Export Directory. You can select some location on your machine (click the ... button).
- 5 Click Export.
- 6 The Status column should indicate Complete. (If there were problems, an error message would be in the Error Message column.)

Create a new user in EE and Assign Different Language

This can be a useful way to check your use of label terms.

You need to use the following:

- User Maintenance
- User Domain / Entity Access Maintain
- Role Membership Maintain

Next, with User Maintenance, change the language of the user you've created.

Create a Report Containing a Sub-report

- 1 Using the Report Resource Designer, create a new report using QAD_SourceNetworkReport.
- 2 To move things along, just set Template to (None).
- 3 First, use ttReportHeader as the table that will be the data source for the report.
- 4 Set the Orientation to Landscape and the Layout to Columns.
- 5 For Select Fields, move only the unique_id field to Details.
- 6 Run the report to check what you have so far.

- 7 You might want to tweak the positioning of the unique_id field listing so it is left-justified and make the unique_id list bold or somehow more prominent.
- 8 Now, back in the Report Resource Designer, in the Reports tab, right-click on Unnamed(ttReportHeader), and choose Create Subreport.
- 9 Choose ttReportDetail, click Next, and click Next.
- 10 For Select Fields, select all of them and put them into the Details box.
- 11 Click Next and click Finish.
- 12 Now in the Reports tab, select SubReport1(ttReportDetails).
- 13 Move the cursor to the Main Designer window. Move it to the Details section. The cursor turns into a cross.
- 14 Click-drag the cursor to place a field containing the subreport.
- 15 Make room for the subreport field by dragging the lower Details boundary down a bit.
- 16 Expand the field containing the subreport so there is room for all that data.
- 17 Preview and run to see the resulting report.
- 18 We forgot to establish the master-detail relationship by linking: this could have been painful if we were dealing with a lot of data...
- 19 Link subreport based on unique ID field by selecting the field containing the subreport, right-click, and choose Link Subreport.
- 20 Set Container report field (master) to: unique_id.
- 21 Set the subreport field (child) to: unique_id. These must match so as to establish the connection between the two reports.
- 22 Now Preview and run the report again.
- 23 You can now also edit the subreport by right-clicking on the field containing it and choosing Edit Subreport (and so on).

Modify a Template and See How It Affects a Set of Reports

Let's modify the template used by all the reports generated directly from browses in the system.

Note that when modifying these templates, it is a very good idea to save them and back them up.

- 1 Open any browse. For example, open Sales Order Browse.
- 2 From the browse, choose Actions > Report. Run the report.
- 3 Note the layout and appearance of the resulting report.
- 4 Now, open the Template Designer.
- 5 Select QAD_Default_Template_Browse.

- 6 In the Main Designer window, click on the field boundary for the title of the report (the rectangle around Browse_Report_Title)
- 7 In the Properties panel, under Appearance, select the BackColor field.
- 8 By default, BackColor is transparent. Change it to be some other color, such as Silver.
- 9 Save the template.
- 10 Now open any browse - for example, open Sales Order Browse.
- 11 From the browse, choose Actions > Report. Run the report.
- 12 Now the resulting report will have the color you selected as the background color in the report title.
- 13 Try running another report from some other browse (for instance, Item Browse). You should see the same new background color in the report title.

Chapter 3

Progress Data Source Program

Session Objectives

Session Objectives

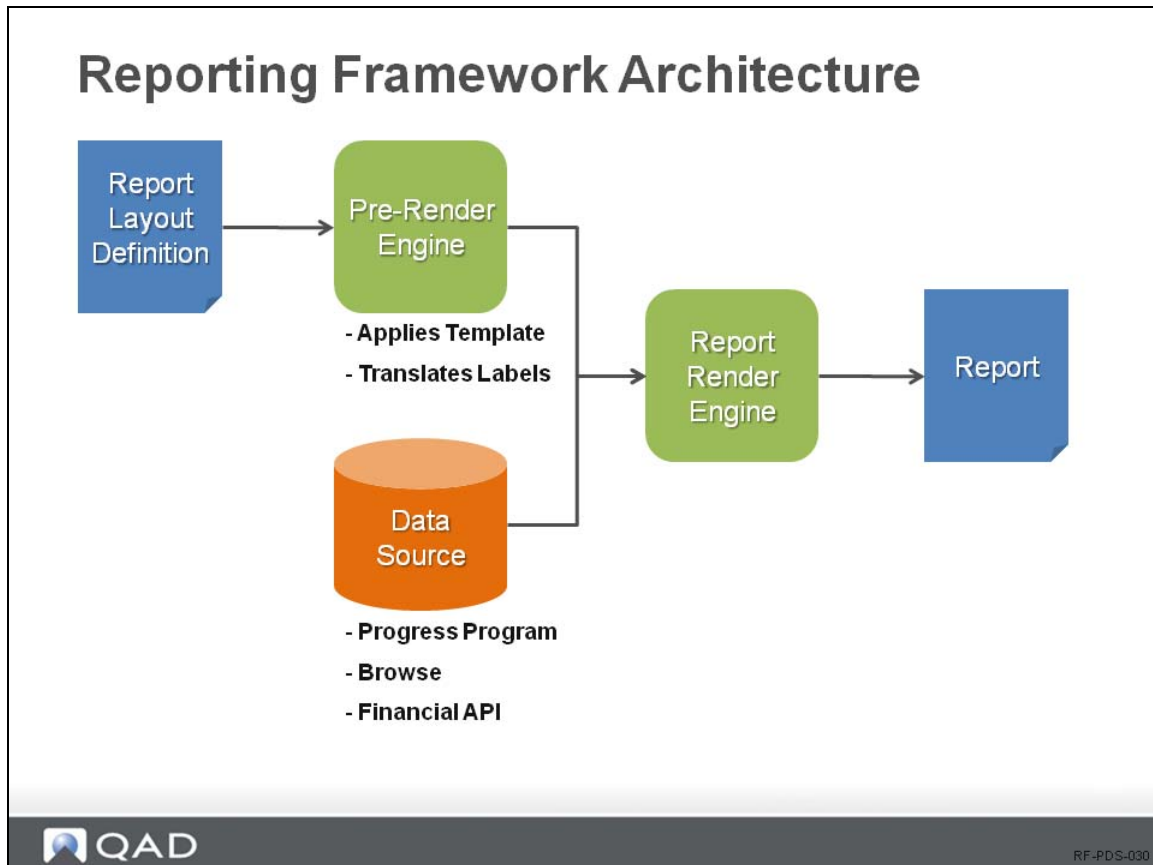
You will learn how to:

- Develop a Progress “Proxy” Data Source Program
- Deploy the data source program into the run-time environment



RF-PDS-020

Reporting Framework Architecture



There are three types of data sources that a report can use: Browse, FinancialAPI, and Progress Program. Our focus in this session is on developing and deploying a Progress Program – a proxy data source program.

Developing a Progress Data Source Program

Developing a Progress Data Source Program

- Program runs on the .NET UI Progress Application Server tier
- Two types of client requests must be supported
 - Metadata request
 - Information about data set tables and fields
 - Field attributes drive client behavior on prompt page
 - Data request
 - Query business data, restricting by filter conditions



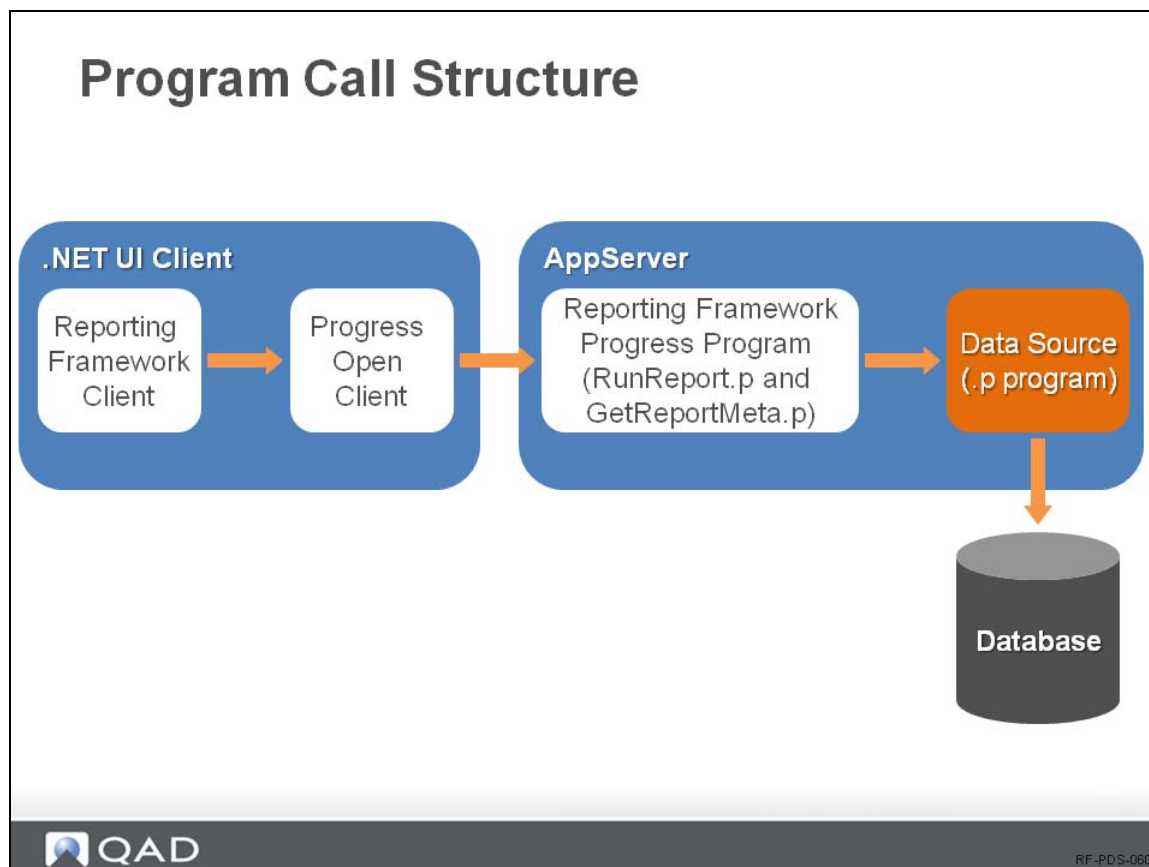
RF-PDS-040

Developing a Progress Data Source Program

Developing a Progress Data Source Program

- Generic entry programs: RunReport.p, GetReportMeta.p
 - All client requests go through GetReportMeta.p for metadata
 - All client requests go through RunReport.p for data query
 - RunReport.p and GetReportMeta.p delegate the call to the specified data source program

Program Call Structure



This diagram illustrates the context in which a Progress data source program is run.

Progress Data Source Program Calls

Progress Data Source Program Calls

- A data source program requires four blocks of code to be written:
 - 1. A data set definition
 - Consists of one or more temp-tables that define the data set structure for the report
 - Tables can be related (e.g. master-detail)
 - 2. Statements to empty the temp-table(s) in the data set
 - 3. Metadata definition that defines attributes for the fields in the data set
 - 4. Data retrieval logic that populates the data set according to filter conditions



RF-PDS-070

A data source program comprises four blocks of code:

- A data set definition (consisting of one or more temp-tables) that defines the data set structure for the report
- Statements to empty the temp-table(s) in the data set
- Metadata definition that defines attributes for the fields in the data set
- Data retrieval logic that populates the data set

Data Source Program Code Sections

A data source program comprises four blocks of code:

- A data set definition (consisting of one or more temp-tables) that defines the data set structure for the report
- Statements to empty the temp-table(s) in the data set
- Metadata definition that defines attributes for the fields in the data set
- Data retrieval logic that populates the data set

In order for the program to be invoked properly by RunReport.p, it must have a specific structure. The following sample code illustrates this structure, showing where the four blocks of code should be inserted:

```
/* UserGuideReportSkeleton.p - Progress Report Data Source program skeleton */
```

```

/* Copyright 2012 QAD Inc. All rights reserved.                                     */
/*                                                                                   */

{mfsubdirs.i}
{{&US_BBI}mfdeclre.i}
{{&US_BBI}gplabel.i}

{com/qad/shell/report/dsReportRequest.i}
{com/qad/shell/report/ReportConstants.i}

/* Report data set definition block */

/* TODO Insert your data set code here */

/* Main Block */
define input parameter runReport as logical no-undo.
define input parameter reportHandle as handle no-undo.
define input parameter dataset for dsReportRequest.
define output parameter dataset-handle phReportResults.

{com/qad/shell/report/reporting.i}

define variable bufferName as character no-undo.

/* Empty temp-table block */

/* TODO empty the temp-table(s) */

for first ttReportRequest no-lock:

    run FillMetaData.

    if runReport then do:
        run RunReport
        (output dataset-handle phReportResults).
    end.

end.

/* Metadata definition block */
procedure FillMetaData:

    /* TODO Insert your metadata code here */

end procedure.

/* Data retrieval logic block */
procedure RunReport:

    define output parameter dataset-handle phReportResults.

    /* TODO Insert your data retrieval code here */

    /* Assign phReportResults to your dataset here */
    /* phReportResults = dataset <your dataset>:handle. */

end procedure.

/* This delete is to prevent a memory leak of the data set */
delete object phReportResults no-error.

```

The code contains several comments starting with `TODO`. These comments indicate where the code blocks should be inserted.

For more information, see *QAD Reporting Framework User Guide*, Appendix A, Implementing a Progress Data Source Program.

Overall Code Structure - Page 1

Overall Code Structure – Page 1

```
(mfsubdirs.i)
({US_BBI}mfdeclre.i)
({US_BBI}gplabel.i)

(com/qad/shell/report/dsReportRequest.i)
(com/qad/shell/report/ReportConstants.i)

/* Report data set definition block */

/* TODO Insert your data set code here */

/* Main Block */
define input parameter runReport as logical.
define input parameter reportHandle as handle.
define input parameter dataset for dsReportRequest.
define output parameter dataset-handle phReportResults.

(com/qad/shell/report/reporting.i)

define variable bufferSize as character no-undo.

/* Empty temp-table block */

/* TODO empty the temp-table(s) */

for first ttReportRequest no-lock:
  run FillMetaData.

  if runReport then do:
    run RunReport
      (output dataset-handle phReportResults).
  end.
end.
```

Continued ...



RF-PDS-080

Overall Code Structure - Page 2

Overall Code Structure – Page 2

... Continued

```
/* Metadata definition block */
procedure FillMetaData:

    /* TODO Insert your metadata code here */

end procedure.

/* Data retrieval logic block */
procedure RunReport:

    define output parameter dataset-handle phReportResults.

    /* TODO Insert your data retrieval code here */
    /* Assign phReportResults to your dataset here */
    /* phReportResults = dataset dsReportResults:handle. */

end procedure.

/* This delete is to prevent a memory leak of the data set */
delete object phReportResults no-error.
```

1. Data Set Definition Block - Example

1. Data Set Definition Block - Example

```

/* Temp-table definition block */
define temp-table ttSalesHeader
  field so_nbr like so_mstr.so_nbr
  field so_cust like so_mstr.so_cust
  field so_ord_date like so_mstr.so_ord_date
  field sales_order_slspn1 like so_mstr.so_slspn[1]
  field sales_order_slspn2 like so_mstr.so_slspn[2]
  field sales_order_slspn3 like so_mstr.so_slspn[3]
  field sales_order_slspn4 like so_mstr.so_slspn[4]

  index SalesHeaderIdx is primary so_nbr
.

define temp-table ttSoLine
  field sales_order_number like so_mstr.so_nbr
  field sales_detail_line like sod_det.sod_line
  field sales_detail_item like sod_det.sod_part
  field sales_detail_unit_measure like sod_det.sod_um
  field sales_detail_due_date like sod_det.sod_due_date

  index SoLineIdx is primary sales_order_number sales_detail_line.
.

define dataset dsReportResults for ttSalesHeader, ttSoLine
  data-relation drLine for ttSalesHeader, ttSoLine
  relation-fields (so_nbr, sales_order_number)

```



RF-PDS-100

Note If you want to use some fields in the temp-table as search fields, you must use the same field names in the temp-tables as those in the database.

Note Indexes specified for the temp tables can be used to define unique constraints and the default sort order of the records.

Note There must be a dataset named dsReportResults defined for temp-tables for the program to work, even if they have no relations.

2. Table Emptying Code - Example

2. Table Emptying Code - Example

```
/* Empty temp-table block */  
empty temp-table ttSalesHeader no-error.  
empty temp-table ttSoLine      no-error.
```



RF-PDS-110

Each temp-table in the report data set should be emptied before the main program logic is executed. This is necessary since temp-table contents could still be cached on the Application Server from previous client requests.

ReportHelper.p Program

ReportHelper.p Program

- Performs tasks common to all data source programs
 - Initializes Metadata data set
- Provides utility functions / procedures
 - Makes specific data source program code more succinct:
 - Functions to assist in creating metadata
 - Functions to assist with building dynamic filter condition strings for 'where' clause
- Runs as a persistent procedure
 - Handle is passed into data source program as an input parameter



RF-PDS-120

3. Metadata Definition Code Block

3. Metadata Definition Code Block

- Needs to fill the metadata output data set
- Define Buffer Header (for each table)
 - Use `CreateBufferHeader` from `ReportHelper.p`
- Define Field Metadata
 - Use `CreateField` from `ReportHelper.p` (and similar procedures)
 - Specify Lookups
 - Translation Considerations
 - Field Labels
 - Value Lists
 - Logical-Typed `fieldFormat`

Metadata: Using CreateBufferHeader

Metadata: Using CreateBufferHeader

- CreateBufferHeader procedure
 - Creates the metadata that describes a table
 - Must be run once for each temp-table in the report data set
- Parameters:
 - tableName
 - tableLabel
- Example:

```
run CreateBufferHeader in reportHandle
  ("ttSalesHeader", "Sales Orders").
```



RF-PDS-140

Table 3.1
CreateBufferHeader Parameters

Seq.	Name	Input/ Output	Data Type	Description
1	tableName	Input	Character	Temp-table name
2	tableLabel	Input	Character	Label displayed in Report Designer

Metadata: Using CreateField

Metadata: Using CreateField

- `CreateField` procedures
 - Create the metadata that describes a field
 - Three variants:
 - `CreateField` – allows programmer to explicitly pass in values of each metadata parameter.
 - `CreateFieldForDBField` - can be used if field is similar to one in business database
 - Some metadata parameters will be determined by the system based on database field.
 - e.g. field label, data type, field format, and lookup name
 - `CreateFieldLikeDBField` - similar to `CreateFieldForDBField`
 - Allows field name of the report field to be different from that of the database field.



RF-PDS-150

Either of three predefined procedures can be used to create field metadata. The procedures are similar and have the same effect of creating a metadata record for one field. A description of each is given below.

`CreateField` — this variant allows the programmer to explicitly pass in the values of each metadata parameter.

`CreateFieldForDBField` — this variant can be used if the field is similar to one in the business database, in which case some of the metadata parameters will be determined by the system based on the database field. For example, the field label, data type, field format, and lookup name will be driven by the database field.

`CreateFieldLikeDBField` — this variant is almost identical to `CreateFieldForDBField` except that it allows the field name of the report field to be different from that of the database field.

(The following sections list the input parameters for each of these three procedures.)

Metadata: CreateField

Metadata: CreateField

CreateField input parameters include:

- bufferName
- fieldName
- fieldLabel
- dataType
- fieldFormat
- lookupName
- isSearchField
- isReadOnlySearch
- isVisible
- isSingleEntry
- isOperatorChangeable
- isRequiredCondition
- isEditable
- defaultValue
- defaultOperator
- defaultValueType
- defaultValue2
- defaultValueType2



RF-PDS-160

Table 3.2
CreateField Parameters

Name	Input/Output	Data Type	Description
bufferName	Input	Character	Temp-table name
fieldName	Input	Character	Report field name
fieldLabel	Input	Character	User-facing label for this field, as it will appear on the prompt page of the report viewer
dataType	Input	Character	Progress datatype of the field
fieldFormat	Input	Character	Progress format for the field. In the case of logical-typed fields, the field format string is used to specify the user-facing labels for the true and false values the field can hold; the syntax is: <code><trueLabel>/<falseLabel></code> . For example, "Debit/Credit". Note: in a multi-language system, these values should not be hard coded but instead dynamically translated using the <code>getLabel</code> function. The format strings will be used in search condition values on the prompt page, as well as data values in the report output.
lookupName	Input	Character	Program name of the lookup program (if any) that will be invoked from the lookup icon for this field on the prompt page of the report viewer. For example, "gp1u340.p".

Name	Input/Output	Data Type	Description
isSearchField	Input	Logical	Whether this field should be a search field on the prompt page
isReadOnlySearch	Input	Logical	Whether this field is read-only on the prompt page
isVisible	Input	Logical	Whether this field is visible in Report Designer
isSingleEntry	Input	Logical	Always set this to False
isOperatorChangeable	Input	Logical	Whether the operator can be changed on the prompt page
isRequiredCondition	Input	Logical	Whether the field is mandatory on the prompt page
isEditable	Input	Logical	Whether the field can be edited on the prompt page
defaultValue	Input	Character	Default value of the first search field
defaultOperator	Input	Character	Default operator of the first search field
defaultValueType	Input	Character	Default value type of the first search field
defaultValue2	Input	Character	Default value of the second search field
defaultValueType2	Input	Character	Default value type of the second search field

Metadata: CreateFieldForDBField

Metadata: CreateFieldForDBField

CreateFieldForDBField parameters include:

- bufferName
- tableName
- fieldName
- isSearchField
- isReadOnlySearch
- isVisible
- isSingleEntry
- isOperatorChangeable
- isRequiredCondition
- isEditable
- defaultValue
- defaultOperator
- defaultValueType
- defaultValue2
- defaultValueType2



RF-PDS-170

Table 3.3
CreateFieldForDBField Parameters

Name	Input/Output	Data Type	Description
bufferName	Input	Character	Temp-table name
tableName	Input	Character	QAD ERP database table name
fieldName	Input	Character	QAD ERP database field name
isSearchField	Input	Logical	Whether this field should be a search field on the prompt page
isReadOnlySearch	Input	Logical	Whether this field is read-only on the prompt page
isVisible	Input	Logical	Whether this field is visible in Report Designer
isSingleEntry	Input	Logical	Always set this to False
isOperatorChangeable	Input	Logical	Whether the operator can be changed on the prompt page
isRequiredCondition	Input	Logical	Whether the field is mandatory on the prompt page
isEditable	Input	Logical	Whether the field can be edited on the prompt page
defaultValue	Input	Character	Default value of the first search field
defaultOperator	Input	Character	Default operator of the first search field
defaultValueType	Input	Character	Default value type of the first search field
defaultValue2	Input	Character	Default value of the second search field
defaultValueType2	Input	Character	Default value type of the second search field

Metadata: CreateFieldLikeDBField

Metadata: CreateFieldLikeDBField

CreateFieldLikeDBField parameters include:

- bufferName
- fName
- tableName
- fieldName
- isSearchField
- isReadOnlySearch
- isVisible
- isSingleEntry
- isOperatorChangeable
- isRequiredCondition
- isEditable
- defaultValue
- defaultOperator
- defaultValueType
- defaultValue2
- defaultValueType2



RF-PDS-180

Table 3.4
CreateFieldLikeDBField Parameters

Name	Input/Output	Data Type	Description
bufferName	Input	Character	Temp-table name
fName	Input	Character	Field name in the temp-table (report field name)
tableName	Input	Character	QAD ERP database table name
fieldName	Input	Character	QAD ERP database field name
isSearchField	Input	Logical	Whether this field should be a search field on the prompt page
isReadOnlySearch	Input	Logical	Whether this field is a read-only on the prompt page
isVisible	Input	Logical	Whether this field is visible in Report Designer
isSingleEntry	Input	Logical	Always set this to False
isOperatorChangeable	Input	Logical	Whether the operator can be changed on the prompt page
isRequiredCondition	Input	Logical	Whether the field is mandatory on the prompt page
isEditable	Input	Logical	Whether the field can be edited on the prompt page
defaultValue	Input	Character	Default value of the first search field
defaultOperator	Input	Character	Default operator of the first search field
defaultValueType	Input	Character	Default value type of the first search field
defaultValue2	Input	Character	Default value of the second search field
defaultValueType2	Input	Character	Default value type of the second search field

Metadata: CreateField - Example

Metadata: CreateField - Example

```
/* The so_nbr field is set as a search field, so the seventh parameter is True. */  
run CreateField in reportHandle  
  (bufferName,  
   "so_nbr",  
   getLabel("SALES_ORDER"),  
   "character",  
   "x(8)",  
   "gplu239.p",  
   true,  
   false,  
   true,  
   false,  
   true,  
   false,  
   true,  
   "",  
   {&ParameterOperator_Equals},  
   {&ParameterValue_Type_Constant},  
   "",  
   {&ParameterValue_Type_Constant}).
```

Metadata: CreateFieldForDBField - Example

Metadata: CreateFieldForDBField - Example

```
/* The following field will have some of its metadata attributes */  
/* driven from the so_cust database field. */  
  
run CreateFieldForDBField in reportHandle  
  (bufferName,  
   "so_mstr",  
   "so_cust",  
   true,  
   false,  
   true,  
   false,  
   true,  
   false,  
   true,  
   "",  
   {&ParameterOperator_Equals},  
   {&ParameterValue_Type_Constant},  
   "",  
   {&ParameterValue_Type_Constant}).
```

Metadata: CreateFieldLikeDBField - Example

Metadata: CreateFieldLikeDBField - Example

```
/* The following field will be called order_date in the report
even though some */
/* of its metadata attributes will be driven from the
so_ord_date database field. */

run CreateFieldLikeDBField in reportHandle
(bufferName,
 "order_date",
 "so_mstr",
 "so_ord_date",
 true,
 false,
 true,
 false,
 true,
 false,
 true,
 "",
 {&ParameterOperator_Equals},
 {&ParameterValue_Type_Constant},
 "",
 {&ParameterValue_Type_Constant}).
```

Metadata: Lookup Syntax

Metadata: Lookup Syntax

- Short-hand Notation:
 - Can be used if the desired lookup is a browse defined by Browse Maintenance
 - Example:
“gplu239.p” – denotes browse gp239 as lookup
- Complete Notation:
 - Must be used if the browse is from a different server implementation (e.g. EE Financial query)
 - `<lookup provider type>:<lookupID>:<lookup return field>:<lookup filter field>`
 - Example:
 - `BaseLibrary.Lookup.BLFLookupProvider:BJournalSAO.SelectJournal:tcJournalCode:tJournal.JournalCode`

Metadata: Value Lists

Metadata: Value Lists

- Allows values for a *character*-typed field to be restricted to a discrete set of values
- Each *value* can be accompanied with a user-facing *label* that can appear in two places:
 - List of values in filter condition (viewer prompt page)
 - Report output document
 - The *value* is what is stored in the DB
- Use the `valueList` field metadata attribute
- Syntax:
 - <Label 1>,<DB value 1>,<Label 2>,<DB value 2>, ...

Metadata: Value Lists - Example

Metadata: Value Lists - Example

- Note: SetFieldMetaParam is a procedure in ReportHelper.p that can be used to set field metadata attributes
- One of the CreateField procedures must first have been called to create the record

```
define variable statusValueList as character no-undo.  
statusValueList = "New,1,In Process,2,Completed,3".  
  
run SetFieldMetaParam in ReportHandle(  
    bufferName,  
    "status",  
    "valueList",  
    statusValueList)  
.
```

Metadata: Translations

Metadata: Translations

- Most translations for reports are done in the page layout design (Report Resource Designer)
- There are also some areas that may require translated text in the metadata returned by a data source program:
 - Field Labels - fieldLabel metadata attribute
 - Value Lists - valueList metadata attribute
 - Logical-Typed value labels - fieldFormat metadata attribute

Metadata: Translations

Metadata: Translations

- Use `global_user_lang` shared variable to determine language to use for getting translated labels
- Use `getLabel()` utility function in `ReportHelper.p`
 - Input: term key string
 - Output: translated long label, using `global_user_lang`
 - Example: `getLabel("SALES_ORDER")`
 - Function definition:

```
FUNCTION getLabel returns character (  
input pTerm as character):
```

4. Data Retrieval Logic Code Block

4. Data Retrieval Logic Code Block

- Used to populate the report data set tables
- Must be implemented in procedure `RunReport`
- Need to Use Dynamic Query
 - To handle filter conditions from client request
 - `FillQueryStringVariable` helper procedure
- Query Iteration and Temp Table Population

Data Retrieval - Defining the Query

Data Retrieval – Defining the Query

```
define variable queryString as character no-undo.  
define variable hSOQuery as handle.  
define query SOQuery for so_mstr.  
  
hSOQuery = query SOQuery:handle.  
  
queryString = "for each so_mstr no-lock "  
  + " where so_mstr.so_domain = " + QUOTER(global_domain).  
  
run FillQueryStringVariable in reportHandle (input  
  "ttSalesHeader", input "so_nbr", input-output queryString).  
  
run FillQueryStringVariable in reportHandle (input  
  "ttSalesHeader", input "so_cust", input-output queryString).  
  
run FillQueryStringVariable in reportHandle (input  
  "ttSalesHeader", input "order_date", input-output  
  queryString).  
  
queryString = queryString + ":".  
  
hSOQuery:query-prepare(queryString).  
hSOQuery:query-open().  
hSOQuery:get-next().
```

FillQueryStringVariable Procedure

FillQueryStringVariable Procedure

- Helper procedure from ReportHelper.p
- Used to add dynamic filter conditions to the query string
- Will construct a where-clause segment and append it to the query string with a logical and
 - Inspects client request data set to read filter conditions



RF-PDS-290

The FillQueryStringVariable function will get the search conditions sent from the client request (each consisting of the search field, operator, and value entered by the user) and then construct the corresponding where clause fragment and add it to the query string dynamically at run time.

The following lists the input parameters for the FillQueryStringVariable function:

Name	Input/Output	Data Type	Description
bufferName	Input	Character	Temp-table name
fieldName	Input	Character	Field name in the temp-table
queryString	Input-Output	Character	Dynamic query string

Note For the function to work, the temp-table fields defined as search fields in the metadata definition should use exactly the same names as those in the database.

Note If other static filter conditions are required in the query string, they can be added in place of the “where true” part of the query string in the above example. For example, if we wanted this report to filter its query to only include orders whose so_site value is “SITE1”, we could use the following statement to begin the query string:

```
queryString = "for each so_mstr no-lock where so_site = " + QUOTER("SITE1").
```

The `QUOTER()` function returns the input string wrapped in quotes, and is useful when specifying text in a dynamic query string that must appear quoted in the final query string.

If static sorting is required, a “by” clause can be added at the end of the dynamic query string. For example, the following change to our above example will cause records to be sorted primarily by `so_cust`:

```
queryString = queryString + " by so_cust:".
```

FillQueryStringVariable Procedure

FillQueryStringVariable Procedure

- Call signature:

```
PROCEDURE FillQueryStringVariable:  
    define input parameter bufferName as character no-undo.  
    define input parameter fieldName as character no-undo.  
    define input-output parameter queryString as character  
no-undo.
```

- Input parameters include:
 - bufferName (temp-table name)
 - fieldName (field name in temp-table)
 - queryString (dynamic query string)

Static where-clause Conditions

Static where-clause Conditions

- Static filter conditions can be hard coded into beginning of query string
 - Example: Filter query to only include orders for “SITE1”

```
queryString = "for each so_mstr no-lock
where so_site = "
+ QUOTER("SITE1").
```

- If no static conditions are needed, add a dummy one ("where true") so that and-ed dynamic conditions are appended properly
- For static sorting, add a ‘by’-clause at the end of the query string
 - Example: Sort records by customer

```
queryString = queryString + " by so_cust:".
```



RF-PDS-310

NOTE: The QUOTER() function returns the input string wrapped in quotes, and is useful when specifying text in a dynamic query string that must appear quoted in the final query string.

GetFilterValue Function

GetFilterValue Function

- Helper procedure from ReportHelper.p
- Used to retrieve filter values from input request
- Call signature:

```
FUNCTION GetFilterValue returns character  
  (input ipBufferName as character,  
   input ipFieldName as character,  
   input defaultValue as character):
```

GetFilterValue Function

GetFilterValue Function

- Input parameters include:
 - ipBufferName (temp-table name)
 - ipFieldName (field name in temp-table)
 - defaultValue (value to return if none found in request)
- Returns: Field value from filter conditions in request

GetFilterValue - Example

GetFilterValue - Example

```
define var orderLineFilterValue as integer no-undo.  
  
orderLineFilterValue =  
integer(GetFilterValue("ttReportOptions", "sod_line",  
"1")).
```

Data Retrieval - Query Iteration

Data Retrieval – Query Iteration

- Iterate over the query records and populate the temp tables of the report data set
- Example...

Data Retrieval - Query Iteration Example

Data Retrieval – Query Iteration Example

```
repeat while not hSOQuery:query-off-end:
  create ttSalesHeader.
  assign
    ttSalesHeader.so_nbr = so_mstr.so_nbr
    ttSalesHeader.so_cust = so_mstr.so_cust
    ttSalesHeader.so_ord_date = so_mstr.so_ord_date
    ttSalesHeader.sales_order_slspn1 = so_mstr.so_slspn[1]
    ttSalesHeader.sales_order_slspn2 = so_mstr.so_slspn[2]
    ttSalesHeader.sales_order_slspn3 = so_mstr.so_slspn[3].
    ttSalesHeader.sales_order_slspn4 = so_mstr.so_slspn[4].

  for each sod_det no-lock
    where sod_det.sod_nbr = so_mstr.so_nbr:

    create ttSoLine.
    assign
      ttSoLine.sales_order_number = sod_det.sod_nbr
      ttSoLine.sales_detail_line = sod_det.sod_line
      ttSoLine.sales_detail_item = sod_det.sod_part
      ttSoLine.sales_detail_unit_measure = sod_det.sod_um
      ttSoLine.sales_detail_due_date = sod_det.sod_due_date.
    end.
  hSOQuery:get-next().
end. /* Repeat query */
```



RF-PDS-360

Progress Data Source Deployment

Progress Data Source Deployment

- Progress data source programs are deployed on the .NET UI Progress application server tier
 - `<desktop source code directory>/com/qad/shell/report/reports`
 - `<desktop source code directory>` usually is: `/qad/web/server/docs/<ENVname>/ebdesktop2/<WEBAPPNAME>/`
- Compile the program and save
 - Must be connected to qadddb and qadadm DBs
 - Can use the `mkdt` program in `<desktop source code directory>`
 - `./mkdt compile`



RF-PDS-370

Note If you prefer to physically locate your Progress data source programs in another directory path, that is fine. Just be sure to realize that at run-time the reporting system will attempt to locate the program using the path `com/qad/shell/report/reports/<DataSourceRef>`, where `<DataSourceRef>` is the value in the Data Source Ref field in Report Resource Maintenance. Therefore, make sure that you name your file accordingly, put your alternate directory into the application server's propath, and place your data source program in a `com/qad/shell/report/reports` subdirectory under your alternate location.

Progress Data Source Deployment: Propath Example

Progress Data Source Deployment Propath Example

- Example propath for compilation:

```
propath = propath + "," +  
"/qad/web/server/docs/93/ebdesktop2/dev93ui/com/qad/shell/report/  
reports".  
  
propath = propath + "," +  
"/qad/web/server/docs/93/ebdesktop2/dev93ui".  
  
propath = propath + "," +  
"/qad/web/server/docs/93/ebdesktop2/dev93ui/com/qad/shell/report".  
  
propath = propath + "," +  
"/qad/mfgpro/93/stage".
```



RF-PDS-380

Exercise

Exercise

1. Install and compile the sample .p data source programs given in the course materials
 - HelloWorldReport.p
 - SimpleItemReport.p
 - TimeDelayReport.p
 - UserGuideSampleReport.p



RF-PDS-390

Exercise

Exercise

2. Import the XML layout definitions for the sample reports
 - Report Resource Import
3. Modify the SimpleItemReport.p program to add a Unit of Measure field
 - Add field to .p to contain pt_mstr.pt_um
 - Add field to page layout in Designer
 - Add it to the SimpleItemReport_Grouped layout

Summary

Summary

In this section, you have learned how to...

- Develop a Progress “Proxy” Data Source Program
- Deploy the data source program into the run-time environment

Exercise Steps

Define Report Definition for HelloWorldReport.p

(The following is based on QAD training environments.)

Get an example program (such as HelloWorldReport.p), and copy it to

```
Z:\qadapps\qea\qadui\
```

In the training environment, proxy report source directory is:

```
Z:\qadapps\qea\qadui\com\qad\shell\report\reports
```

Where Z: is mapped to \\qaddemo\dr01

In the QAD .NET UI, launch the Progress Editor (mgedit)

Enter:

```
compile /dr01/qadapps/qea/qadui/HelloWorldReport.p save into
/dr01/qadapps/qea/qadui/com/qad/shell/report/reports.
```

...and run it (F1).

Now use Report Resource Maintenance to define the report code for the proxy data source, identifying the source as HelloWorldReport.p

Finally, use Report Resource Designer to generate a simple report.

Example Code: HelloWorldReport.p

```
/* HelloWorldReport.p - OPERATIONS BY WORK ORDER REPORT          */
/* Copyright 1986-2012 QAD Inc., Carpinteria, CA, USA.          */
/* All rights reserved worldwide. This is an unpublished work.  */
/*                                                                */
/* Simple example report datasource program.                     */
/*                                                                */
/* $Id: HelloWorldReport.p 12860 2008-10-06 02:07:55Z czs $     */
/*-Revision end-----*/

{mfdeclre.i}
{gplabel.i}

{com/qad/shell/report/ReportConstants.i}
{com/qad/shell/report/dsReportRequest.i}

/*-----*/
/* ***** Report Table Definition ***** */
/*-----*/

/* This temp table name is arbitrary, but the dataset it belongs to should be */
/* named dsReportResults.                                                    */
/* Any temp tables in the dsReportResults dataset will be returned to the    */
/* report client at runtime.                                                  */

define temp-table ttHelloWorld
  field count as integer
  field hello as character
.

define dataset dsReportResults
```

```

    for ttHelloWorld
    .

/*=====*/
/* ***** Main routine ***** */
/*=====*/

define input parameter runReport as logical.
define input parameter reportHandle as handle.
define input parameter dataset for dsReportRequest.
define output parameter dataset-handle phReportResults.

{com/qad/shell/report/reporting.i}

define variable bufferName as character no-undo.

empty temp-table ttHelloWorld no-error.

for first ttReportRequest no-lock:

    /* Get the meta data about the fields */
    run FillMetaData.

    if runReport then do:
        run RunReport
            (output dataset-handle phReportResults).
    end.
end.

/*=====*/
/* ***** Procedures ***** */
/*=====*/

/* Meta Data */

/* This section defines metadata attributes for the fields which controls */
/* their behavior as search fields and sets other properties. */
/* The helper procedure which is invoked to set the metadata has the */
/* following input parameters: */
/* PROCEDURE CreateField: */
/*   define input parameter bufferName as character. */
/*   define input parameter fieldName as character. */
/*   define input parameter fieldLabel as character. */
/*   define input parameter dataType as character. */
/*   define input parameter fieldFormat as character. */
/*   define input parameter lookupName as character. */
/*   define input parameter isSearchField as logical. */
/*   define input parameter isReadOnlySearch as logical. */
/*   define input parameter isVisible as logical. */
/*   define input parameter isSingleEntry as logical. */
/*   define input parameter isOperatorChangeable as logical. */
/*   define input parameter isRequiredCondition as logical. */
/*   define input parameter isEditable as logical. */
/*   define input parameter defaultValue as character. */
/*   define input parameter defaultOperator as character. */
/*   define input parameter defaultValueType as character. */
/*   define input parameter defaultValue2 as character. */
/*   define input parameter defaultValueType2 as character. */

procedure FillMetaData:

    bufferName = "ttHelloWorld".
    run CreateBufferHeader in reportHandle
        (bufferName, "Hello World").

    run CreateField in reportHandle
        (bufferName,
         "count",
         "Count",
         "integer",
         "9999",

```

```

    " ",
    false,
    false,
    true,
    false,
    false,
    false,
    false,
    " ",
    "Equals",
    "Constant",
    " ",
    "Constant"
  ).

run CreateField in reportHandle
(bufferName,
"hello",
"Hello",
"character",
"x(40)",
" ",
false,
false,
true,
false,
false,
false,
false,
" ",
"Equals",
"Constant",
" ",
"Constant"
).

end procedure.

procedure RunReport:

  define output parameter dataset-handle phReportResults.

  /* This is the section where the output report dataset gets populated */

  define variable cnt as integer initial 1 no-undo.

  do while cnt <= 10:
    create ttHelloWorld.
    assign
      ttHelloWorld.count = cnt
      ttHelloWorld.hello = "Hello world !"
    .

    cnt = cnt + 1.
  end.

  phReportResults = dataset dsReportResults:handle.
end procedure.

```


Chapter 4

Reporting Framework Administration

Session Objectives

Session Objectives

You will learn how to:

- Configure the QAD Reporting Framework
- Set up user authorization for report development and admin programs
- Manage reports on the system menu
- Administer scheduled report servers
- Recent enhancements



RF-ADMIN-020

System Configuration: client-session.xml

System Configuration: client-session.xml

- Several reporting-related settings must exist in the client-session.xml configuration file on the QAD .NET UI home server
- File location:
 - <TomcatInstallDir>/webapps/qadhome/configurations/<config>/client-session.xml
- Reporting-related entries:
 - ReportDataSourceProviders
 - Tells system how to load data source implementation classes
 - Allows custom implementations to be added
 - SMTP settings
 - For email notifications from scheduled report runs



RF-ADMIN-030

You can write a new custom data source provider implementation. For example, you might have a non-QAD data store that you want to report against. Write a .NET class that implements the `IDataSourceProvider` interface, deploy it in the QAD .NET UI environment, and register it by adding a `<Provider>` section to `client-session.xml`. The `Name` is arbitrary (and will appear in the data source options in `ReportResourceMaintenance`), and the `Assembly` and `Class` name must match the DLL and class name (with full namespace) that you developed.

Report Data Source Provider Settings

Report Data Source Provider Settings

- Registers report data source provider implementations
- The following is the entry as currently shipped:

```
<ReportDataSourceProviders>
  <Provider>
    <Name>GenericProxy</Name>
    <Assembly>QAD.Plugin.Reports.ReportFramework</Assembly>
    <Class>QAD.Plugin.Reports.ReportFramework.GenericProxyDatasourceProvider</Class>
  </Provider>
  <Provider>
    <Name>Browse</Name>
    <Assembly>QAD.Browse</Assembly>
    <Class>QAD.Browse.BrowseDatasourceProvider</Class>
  </Provider>
  <Provider>
    <Name>FinancialAPI</Name>
    <Assembly>BaseLibrary</Assembly>
    <Class>BaseLibrary.Reports.BLFDatasourceProvider</Class>
  </Provider>
</ReportDataSourceProviders>
```



RF-ADMIN-040

You can write a new custom data source provider implementation. For example, you might have a non-QAD data store that you want to report against.

- 1 Write a .NET class that implements the `IDatasourceProvider` interface
- 2 Deploy it in a DLL in the .NET UI environment
- 3 Add a generalized code entry for it: `FieldName=rptres_datasource_type`, `Value=any integer` (safest to pick one >100 to avoid future QAD conflicts), `Comments:arbitrary`; will appear in the code lookup in RR Maint.
- 4 Register it by adding a `<Provider>` section to `client-session.xml`. The `Name` should match the integer chosen for the code value in step 3, and the `Assembly` and `Class` name must match the DLL and class name (with full namespace) that you developed.

Data Source Provider Settings - Change For Financials

Data Source Provider Settings – Change For Financials

- The entry for the FinancialAPI changed between 2009 EE and 2009.1 EE releases
 - Entry for 2009 EE:

```
<Provider>  
  <Name>FinancialAPI</Name>  
  <Assembly>BaseLibrary</Assembly>  
  <Class>BaseLibrary.Reports.BLFDatasourceProvider</Class>  
</Provider>
```

- Entry for 2009.1 EE:

```
<Provider>  
  <Name>FinancialAPI</Name>  
  <Assembly>BaseAdapters.Reporting</Assembly>  
  
<Class>BaseAdapters.Reporting.DatasourceProvider.BLFDatasourceProvider</Class>  
</Provider>
```



RF-ADMIN-050

rptAdmin and rptDsgn Roles / Groups

rptAdmin and rptDsgn Roles / Groups

- These roles are used to grant access to report development and admin programs
- **rptAdmin** membership allows access to all of the reporting programs
- **rptDsgn** membership allows access to a subset of the programs
 - Appropriate for report developers
- Admin Programs for User Roles (EE):
 - Role View, Role Create, User Role View, Role Membership Maintain
- Admin Programs for User Groups (SE):
 - User Group Maintenance



RF-ADMIN-060

You must create two roles — rptAdmin and rptDsgn — in Role Create for the report administrator and report designer/developer respectively, and then assign them to the particular user IDs you would like to perform the associated activities. These roles add another layer of security that controls access to some activities within the programs. Since the activity-level controls are hard-coded in the reporting programs, you will not be able to perform certain activities within these programs if you create roles with other names for the report administrator and report designer/developer.

Note Make sure you use the correct capitalization for the roles.

Program Authorization Matrix

Reporting Program	rptAdmin	rptDsgn
Report Designer	Allow	Allow
Template Designer	Allow	Allow
Report Resource Import	Allow	Allow
Report Resource Export	Allow	Allow
Scheduled Report Maintenance	Allow	
Report Resource Maintenance	Allow	Allow
Report Parameter Maintenance	Allow	Allow
Personal use Filter Maintenance	Allow	Allow
Admin User Filter Maintenance	Allow	
Report Settings Restore	Allow	

You must grant the rptAdmin and rptDsgn roles access to the following programs based on this table:

Table 4.1
Program-Role Access Control Matrix

Reporting Program	rptAdmin	rptDsgn
Report Designer	Allow	Allow
Template Designer	Allow	Allow
Report Resource Import	Allow	Allow
Report Resource Export	Allow	Allow
Scheduled Report Maintenance	Allow	
Report Resource Maintenance	Allow	Allow
Report Parameter Maintenance	Allow	Allow
Personal User Filter Maintenance	Allow	Allow
Admin User Filter Maintenance	Allow	
Report Settings Restore	Allow	

For more information on common access security features, see [QAD Security and Controls User Guide](#).

Adding a Report to the Menu

Adding a Report to the Menu

- Use standard Menu System Maintenance
 - Exec Procedure must be a URN of the format:
`urn:qad-report:c1:<ReportCode>`
- If the report has multiple layout definitions, the one marked as the *default* will execute from the menu.
- Menu security is done in standard fashion
 - EE: Role Permissions Maintain
 - SE: Menu Security Maintenance



RF-ADMIN-080

The ReportCode value in the URN must match the one entered into Report Resource Maintenance for the required report.

Exercise

Exercise

1. Locate client-session.xml on your system
2. View the file (use `vi` or `more`)
 - Locate the `ReportDataSourceProviders` section
 - Locate the SMTP section
3. Launch *User Role View*
 - Determine which users are in the `rptAdmin` and `rptDsgn` groups



RF-ADMIN-090

Exercises include the following:

- 1 Locate the client-session.xml file on your system.
- 2 View the file (use `vi` or `more`):
 - Locate the `ReportDataSourceProviders` section
 - Locate the SMTP section
- 3 Launch User Role View and determine which users are in the `rptAdmin` and `rptDsgn` groups.

Scheduled Reports

Scheduled Reports

- Report Server(s) run queues of reports (batches)
- Server processes
 - Non-GUI .NET UI processes
 - Standard installation of .NET UI
 - Launched from command line
- Windows Task Scheduler used to periodically launch batches (for example: daily, monthly)
- Output to printer and/or file on web server
- Optional notifications to email, .NET UI Inbox with output as attachment or link to the output



RF-ADMIN-100

Note QRF scheduled report servers should not be confused with the financial report daemons in the EE product. The latter is separate from the QRF and is used in conjunction with the Crystal Reports for EE financials.

Scheduled Reports: Installation and Deployment

Scheduled Reports

Installation and Deployment

- Report Framework seamlessly integrated into .NET UI
- No special installation needed
- No license fees to customers



RF-ADMIN-110

Scheduled Reports: Installation and Deployment

Scheduled Reports

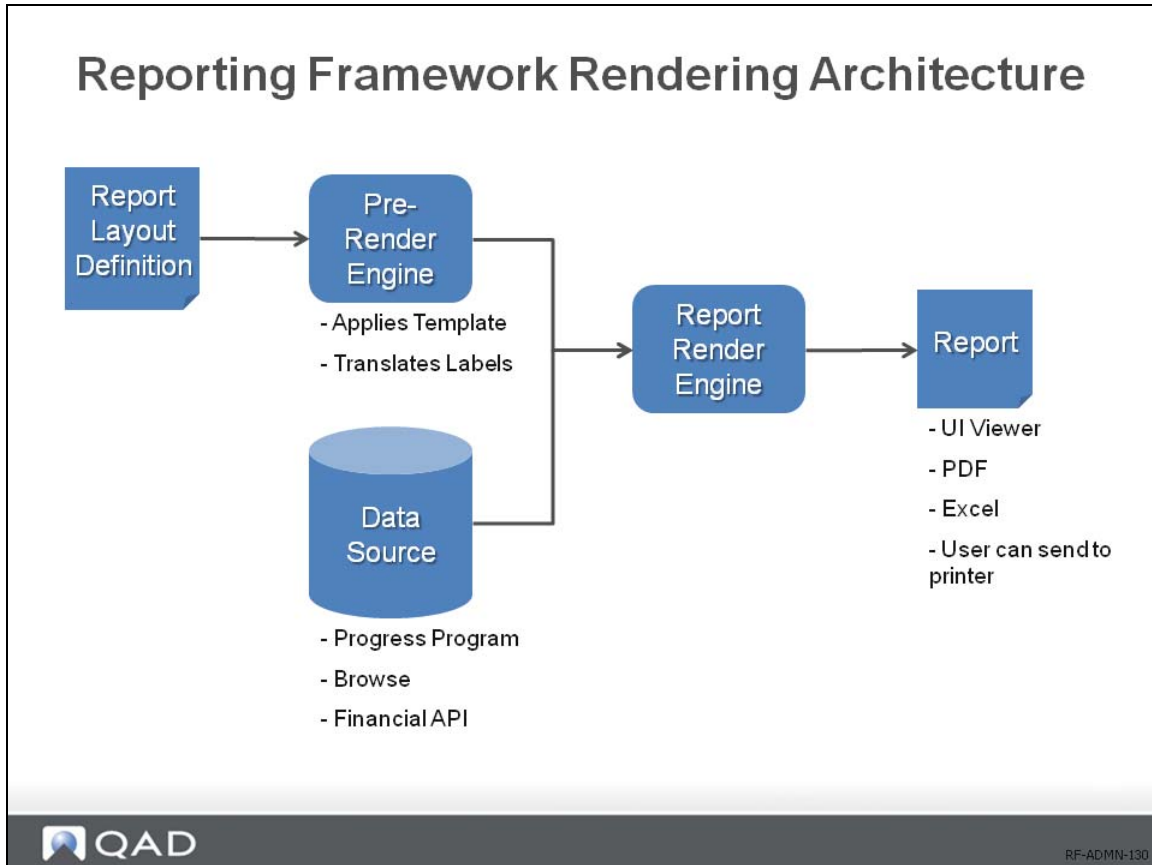
Installation and Deployment (continued)

- Report Server Installation
 - Windows OS required
 - Install .NET UI client in standard fashion
 - Run the server using command line (no GUI)
 - Can deploy multiple servers
 - Failover
 - Increased throughput

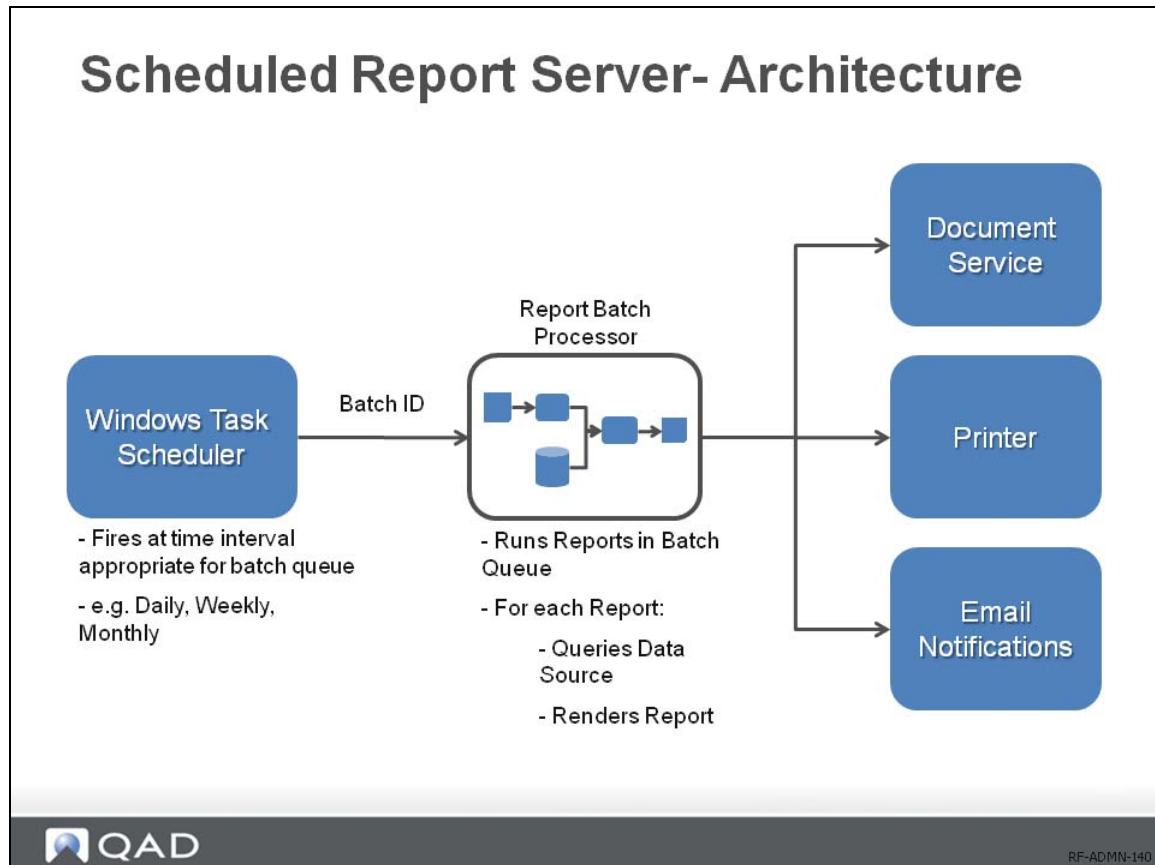


RF-ADMIN-120

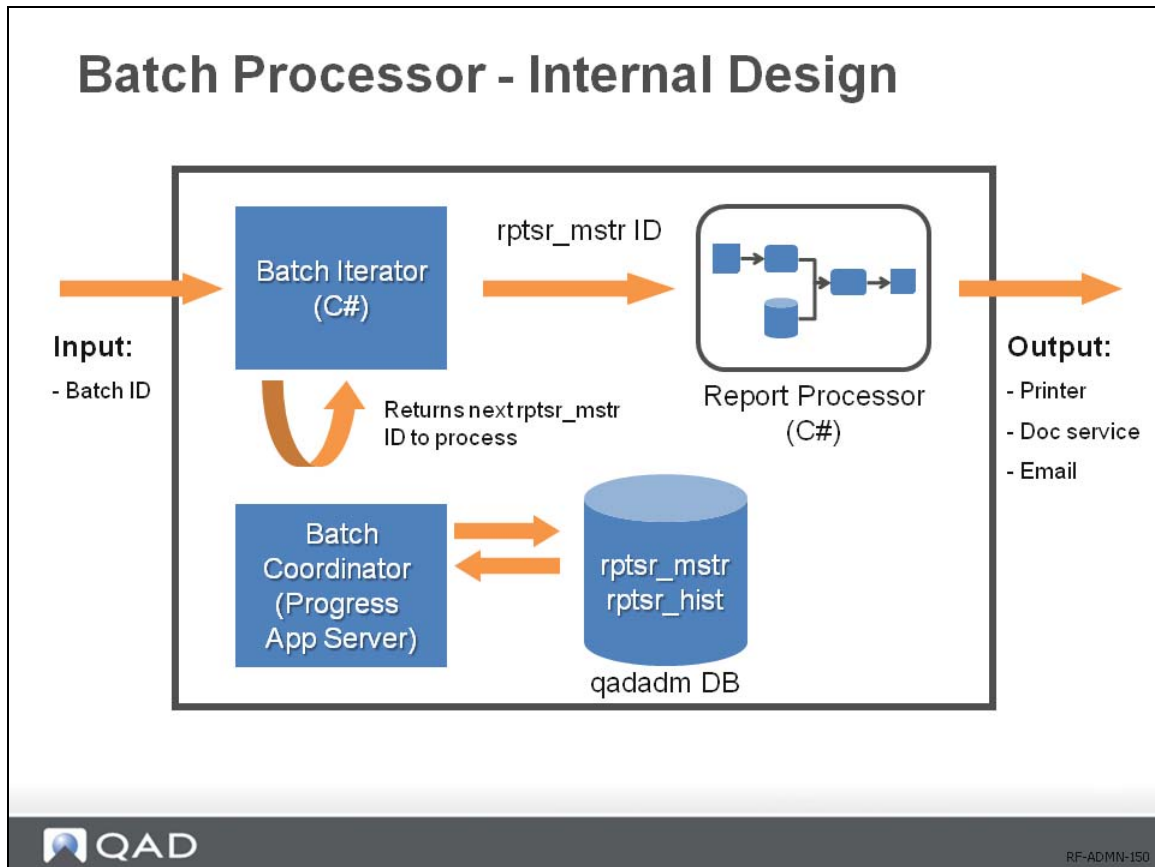
Reporting Framework: Rendering Architecture



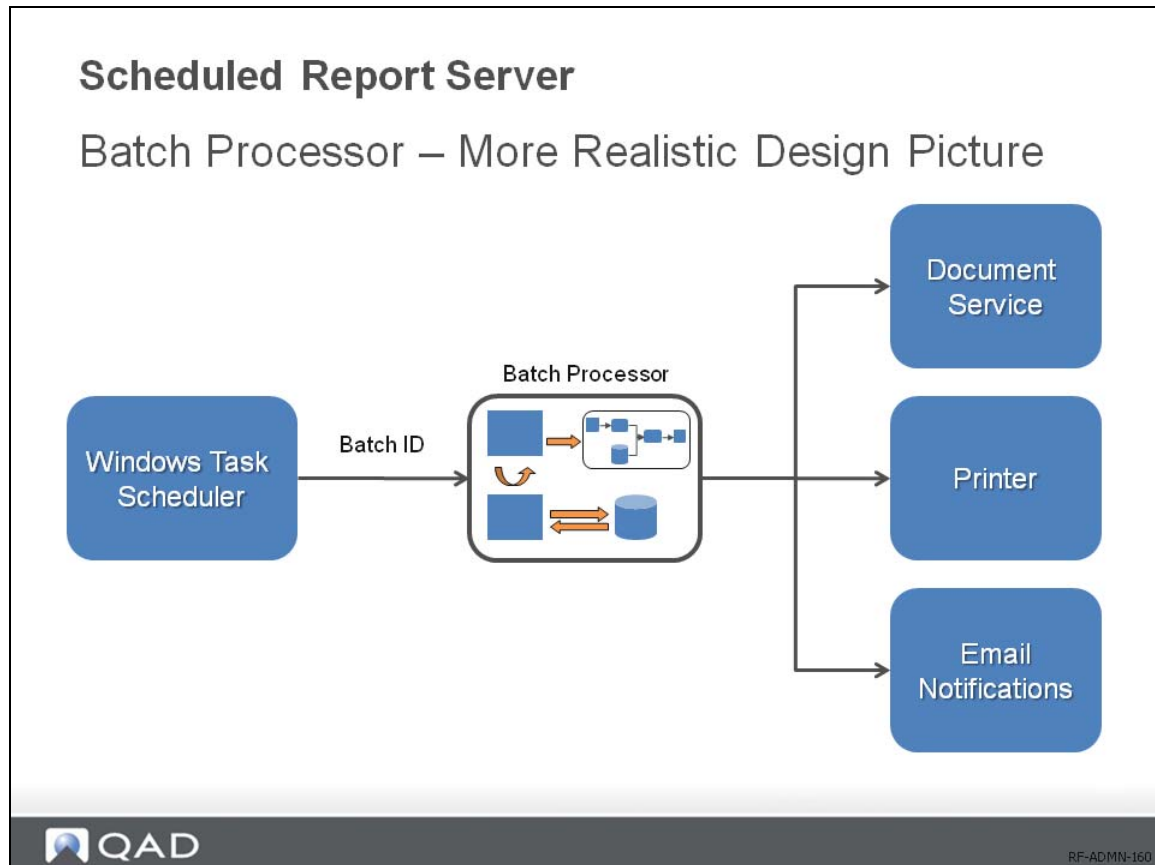
Scheduled Report Server - Architecture



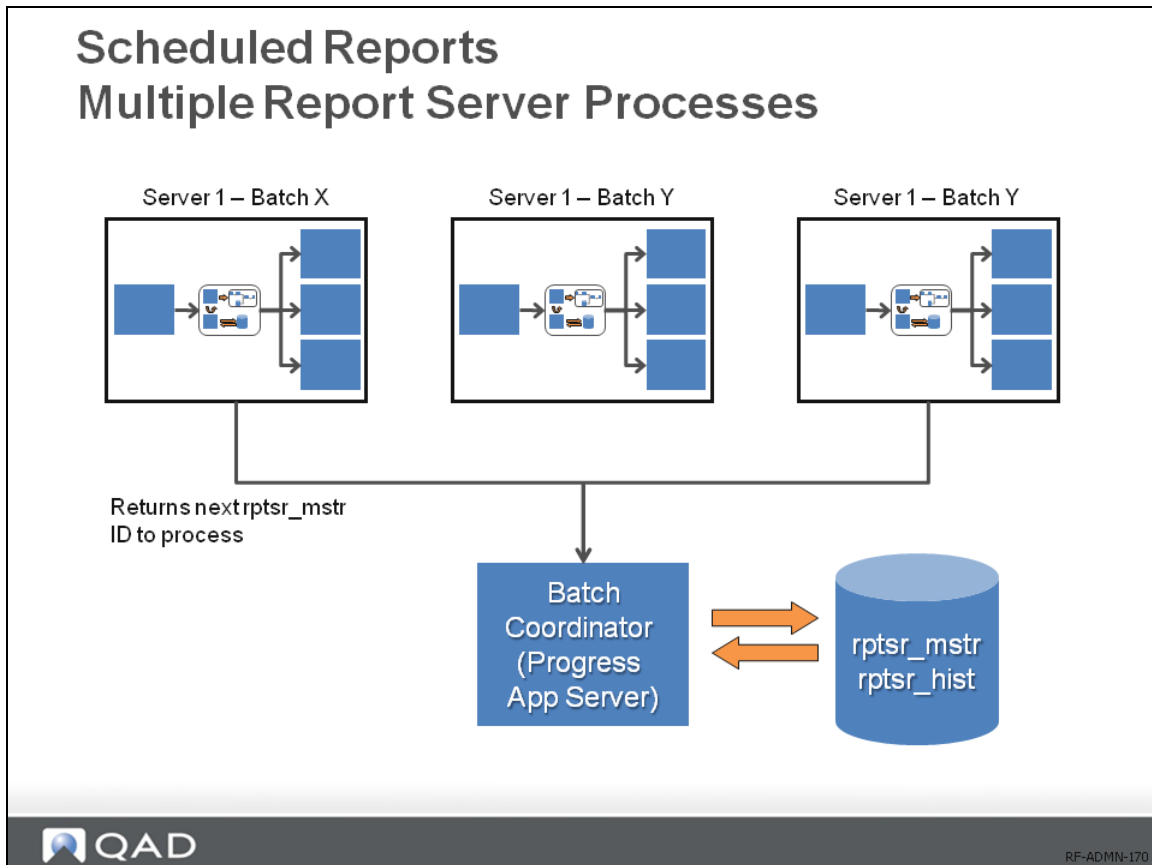
Batch Processor - Internal Design



Scheduled Report Server: Batch Processor Design



Scheduled Reports: Multiple Report Server Processes



When multiple report server processes are run, they can be either on the same machine or different machines.

The physical partitioning can be determined by examining batch frequency and number of reports. Report rendering is a CPU-intensive process.

Any one report server process is linked to a single Batch ID.

It is allowed to have multiple server processes handling the same Batch ID.

Scheduled Reports Status

Scheduled Report *Status*

- Scheduled report records in the batch queues have a `rptsr_mstr.rptsr_status` field containing current status
- Used by batch processor to coordinate the processing of the batch
- Status Values:
 - **NEW:** Newly created and has never run
 - **WAITING:** Initialized to run in current batch run
 - **RUNNING:** Currently running
 - **COMPLETED:** Finished running, with no errors
 - **ERROR:** Finished running, with errors

Scheduled Report Administration: Batch ID Maintenance

Scheduled Report Administration

Batch ID Maintenance

- Same batch DB table used (bc_mstr) as for legacy MFG/PRO reports
 - Different detail table: rptsr_mstr (one record per scheduled report)
 - Each batch ID can logically a queue of scheduled reports (as rptsr_mstr records)
- Same *Batch ID Maintenance* program as for legacy MFG/PRO reports

Batch ID Maintenance

Go To Actions Copy Print Preview Attach

Batch ID: breport1

Batch ID:

Permanent:

Priority:

Scheduled Report Administration: Configuring Windows Task Scheduler

Scheduled Report Administration

Configuring Windows Task Scheduler

- Every time the scheduler fires, a report server process will be launched for a specific Batch ID
 - Can have nightly, weekly, monthly batches
- Configure the Task Scheduler to use a command-line to launch the report server:
 - For local parameter file reference:
QAD.Client.exe -param.url:file:///c:/params.pf
 - For remote parameter file reference:
QAD.Client.exe -param.url:http://somehost/rpt/params.pf
 - Example params.pf file:

```
-silent  
-config-name:test  
-user:mfg  
-password:(blank)  
-workspace:Domain1.1000  
-report-batch:nightly1  
-enable:qad.plugin.services  
-enable:qad.plugin.reports  
-enable:qad.plugin.reportserver  
-report-mode:batch
```

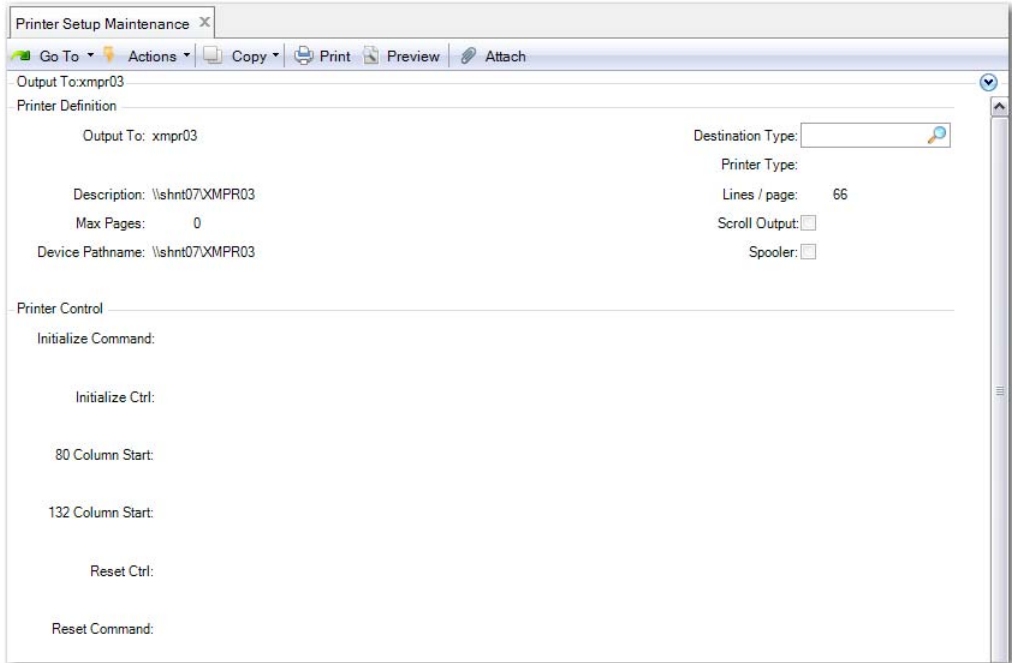
Printer Setup

Printer Setup


- Scheduled Report Printers:
 - Set up as Windows printers on the report server machine
 - Allows the .NET report server processes to use them
 - Set up in client-session.xml

```
<ReportServer>
  <Printer>
    <UNCPath>\\machineA\printerA</UNCPath>
    <Description>Description of printer (optional)</Description>
  </Printer>
  <Printer>
    <UNCPath>\\machineB\printerB</UNCPath>
    <Description>Description of printer (optional)</Description>
  </Printer>
</ReportServer>
```

Using Printer Setup Maintenance



The screenshot displays the 'Printer Setup Maintenance' window. The title bar includes a close button and the text 'Printer Setup Maintenance'. Below the title bar is a menu bar with 'Go To', 'Actions', 'Copy', 'Print', 'Preview', and 'Attach'. The main content area is divided into two sections: 'Printer Definition' and 'Printer Control'.
Printer Definition:
Output To: xmpr03
Description: \\shnt07\XMPR03
Max Pages: 0
Device Pathname: \\shnt07\XMPR03
Destination Type: [Searchable dropdown]
Printer Type:
Lines / page: 66
Scroll Output:
Spooler:
Printer Control:
Initialize Command:
Initialize Ctrl:
80 Column Start:
132 Column Start:
Reset Ctrl:
Reset Command:

 RF-ADMIN-220

Email SMTP Settings

Email SMTP Settings

- Configures the .NET UI system to tell it what SMTP server to use for sending email
- Use the following entry in client-session.xml:

Template:

```
<Configuration>
...
<!-- SMTP server host name -->
<Smtp.Host>SMTPHostname</Smtp.Host>
<!-- SMTP port name -->
<Smtp.Port>SMTPPortNumber</Smtp.Port>
<!-- SMTP from email address -->
<Smtp.From>E-Mail</Smtp.From>
<!-- SMTP username -->
<Smtp.Username>SMTPUsername</Smtp.Username>
<!-- SMTP password -->
<Smtp.Password>SMTPPassword</Smtp.Password>
<!-- SMTP use SSL -->
<Smtp.UseSSL>>false</Smtp.UseSSL>
...
</Configuration>
```

Example:

```
<Configuration>
...
<Smtp.Host>smtp.mycompany.com</Smtp.Host>
<Smtp.Port>25</Smtp.Port>
<Smtp.From>Report Server <joedoe@qad.com></Smtp.From>
<Smtp.Username>admin</Smtp.Username>
<Smtp.Password>123</Smtp.Password>
<Smtp.UseSSL>>false</Smtp.UseSSL>
...
</Configuration>
```



Email Template File

Email Template File

- Allows administrator to customize the email notifications sent from report servers
- File located on home server:
report-email-template.txt
- File location:
<TomcatInstallDir>/webapps/qadhome/configurations/<config>/storage/reports/
- Example Contents:

```
[SUBJECT]
Scheduled Report Completed: {$RRO_DESC}
[BODY]
A scheduled report from QAD Enterprise Applications has completed:

Report:      {$RRO_DESC} ({$RRO_CODE})
Description: {$SR_DESC}
Link to Report: {$REPORT_FILE_LINK}
```



RF-ADMIN-240

Available dynamic variables to use in the email template:

`{$RRO_CODE}` : Report Code (as entered in Report Resource Maintenance).

`{$RRO_DESC}` : Description (as entered in Report Resource Maintenance).

`{$SR_DESC}` : Scheduled Report Description (as entered in the New Schedule Report form).

`{$REPORT_FILE_LINK}` : URL Link to output PDF file (if file output was specified when scheduled report was requested).

Scheduled Report Administration

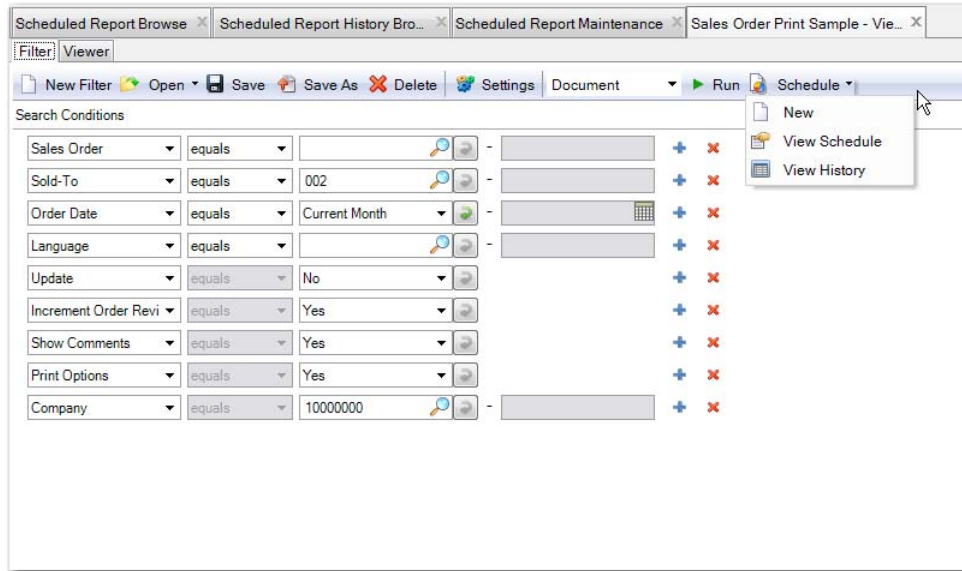
Scheduled Report Administration

- **Any End User**
 - Can create and monitor scheduled reports from the report viewer screen
 - Visibility restricted to the current report open in viewer
- **Administrator**
 - Scheduled Report Browse
 - Shows current status of all scheduled reports in each batch
 - Scheduled Report History Browse
 - Shows all past scheduled report runs
 - Scheduled Report Maintenance
 - Allows viewing and editing of scheduled report properties
 - e.g. changing priorities, deleting

End User Report Scheduling

End User Report Scheduling

Scheduling and monitoring can be done from the viewer



Scheduled Report Browse

Scheduled Report Browse

- Shows current status of scheduled reports in each batch

The screenshot shows the 'Scheduled Report Browse' application window. It features a search bar at the top with a 'Search' button and a 'Clear All' button. Below the search bar, there are navigation controls for 'Viewing 1 - 30 of 30' and 'Records per page: 100'. The main area contains a table with the following columns: Domain, Batch ID, Priority, Create Date, Create Time, Status, Report Code, Active, Permanent, and Last Star.

Domain	Batch ID	Priority	Create Date	Create Time	Status	Report Code	Active	Permanent	Last Star
Domain1	ci01	0	7/12/2009	22:42:47	COMPLETE	hzyrpt004	Yes	Yes	
Domain1	ci01	0	7/12/2009	22:53:42	COMPLETE	hzyrpt004	Yes	Yes	
Domain1	ci01	0	7/12/2009	22:58:08	COMPLETE	hzyrpt004	Yes	Yes	
Domain1	ci01	0	7/12/2009	23:10:28	COMPLETE	hzyrpt004	Yes	Yes	
Domain1	ci01	0	7/12/2009	23:11:51	COMPLETE	hzyrpt004	Yes	Yes	
Domain1	ci01	0	7/12/2009	23:12:41	COMPLETE	hzyrpt004	Yes	Yes	
Domain1	ci01	0	7/12/2009	23:15:34	COMPLETE	hzyrpt004	Yes	Yes	
Domain1	ci01	0	7/12/2009	23:18:07	COMPLETE	hzyrpt004	Yes	Yes	
Domain1	ci01	0	7/12/2009	23:19:55	COMPLETE	hzyrpt004	Yes	Yes	
Domain1	ci01	0	7/12/2009	23:20:42	COMPLETE	hzyrpt004	Yes	Yes	
Domain1	ci01	0	7/12/2009	23:22:19	COMPLETE	hzyrpt004	Yes	Yes	
Domain1	ci01	0	7/12/2009	23:23:25	COMPLETE	hzyrpt004	Yes	Yes	
Domain1	ci01	0	7/12/2009	23:24:49	COMPLETE	hzyrpt004	Yes	Yes	
Domain1	ci01	0	7/13/2009	00:02:33	NEW	hzyrpt001	Yes	Yes	
Domain1	cn	0	7/16/2009	20:04:29	COMPLETE	ynh_rpt001	Yes	Yes	
Domain1	cn01	0	7/12/2009	23:43:22	COMPLETE	hzyrpt001	Yes	Yes	
Domain1	cn01	0	7/16/2009	17:36:34	COMPLETE	hzyrpt001	Yes	Yes	

Scheduled Report History Browse

Scheduled Report History Browse

- Shows past scheduled report runs

The screenshot shows a web application window titled "Scheduled Report History Browse". The interface includes a search bar with a "Search" button and a "Clear All" button. Below the search bar, it indicates "Viewing 1 - 100 of 134" records and "Records per page: 100". The main content is a table with the following columns: Domain, Batch ID, Start Date, Start Time, Status, Report Code, and URL. The table contains 15 rows of data, with the first row highlighted in blue.

Domain	Batch ID	Start Date	Start Time	Status	Report Code	URL
Domain1	ci01	7/10/2009	00:42:26	ERROR	hzyrpt004	
Domain1	ci01	7/10/2009	00:55:44	ERROR	hzyrpt004	
Domain1	ci01	7/10/2009	01:07:38	COMPLETE	hzyrpt004	http://coli48.qad.com:8888/qadzinfandeleee/webdav/configurations/test/storag
Domain1	ci01	7/10/2009	01:13:53	COMPLETE	hzyrpt002	http://coli48.qad.com:8888/qadzinfandeleee/webdav/configurations/test/storag
Domain1	ci01	7/12/2009	23:48:44	COMPLETE	hzyrpt004	http://coli48.qad.com:8888/qadzinfandeleee/webdav/configurations/test/storag
Domain1	ci01	7/12/2009	23:49:08	COMPLETE	hzyrpt004	http://coli48.qad.com:8888/qadzinfandeleee/webdav/configurations/test/storag
Domain1	ci01	7/12/2009	23:49:18	COMPLETE	hzyrpt004	http://coli48.qad.com:8888/qadzinfandeleee/webdav/configurations/test/storag
Domain1	ci01	7/12/2009	23:49:28	COMPLETE	hzyrpt004	http://coli48.qad.com:8888/qadzinfandeleee/webdav/configurations/test/storag
Domain1	ci01	7/12/2009	23:49:39	COMPLETE	hzyrpt004	http://coli48.qad.com:8888/qadzinfandeleee/webdav/configurations/test/storag
Domain1	ci01	7/12/2009	23:49:49	COMPLETE	hzyrpt004	http://coli48.qad.com:8888/qadzinfandeleee/webdav/configurations/test/storag
Domain1	ci01	7/12/2009	23:49:59	COMPLETE	hzyrpt004	http://coli48.qad.com:8888/qadzinfandeleee/webdav/configurations/test/storag
Domain1	ci01	7/12/2009	23:50:09	COMPLETE	hzyrpt004	http://coli48.qad.com:8888/qadzinfandeleee/webdav/configurations/test/storag
Domain1	ci01	7/12/2009	23:50:20	COMPLETE	hzyrpt004	http://coli48.qad.com:8888/qadzinfandeleee/webdav/configurations/test/storag
Domain1	ci01	7/12/2009	23:50:29	COMPLETE	hzyrpt004	http://coli48.qad.com:8888/qadzinfandeleee/webdav/configurations/test/storag
Domain1	ci01	7/12/2009	23:50:40	COMPLETE	hzyrpt004	http://coli48.qad.com:8888/qadzinfandeleee/webdav/configurations/test/storag
Domain1	ci01	7/12/2009	23:50:51	COMPLETE	hzyrpt004	http://coli48.qad.com:8888/qadzinfandeleee/webdav/configurations/test/storag

Scheduled Report Maintenance

Scheduled Report Maintenance

Allows viewing and editing of scheduled report properties

The screenshot shows a web browser window with the title "Scheduled Report Maintenance". The browser's address bar displays "Batch ID: ci01". The page content includes the following fields and controls:

- Batch ID: ci01
- Domain: Domain1
- Batch Details section:
 - Schedule ID: 9
 - Report Code: hzyrpt004
 - Priority:
 - Permanent:
 - Status:
 - Alert:
 - Interval:
 - Timeout:
 - Description:
- Batch ID: (with a search icon)
- Create Date: 07/12/09 22:42:47
- Last Run Date: 07/12/09 23:48:44
- Host Name: HZY-SH
- Process ID: 6624
- Version: 31

Scheduled Reports: Exercise

Scheduled Reports: Exercise

1. Create a new batch ID
 - Batch ID Maintenance
2. Create several scheduled reports in the batch
 - Run report from menu, use Schedule button
 - Make sure some have RunOnce=false, SaveFile=true
 - Use View Schedule button to review
3. Configure and run a scheduled report server process
 - From DOS prompt
 - From Windows Task Scheduler
 - Start > Control Panel > Scheduled Tasks
 - Monitor batch runs using *Scheduled Report Browse* and *Scheduled Report History Browse*
4. Modify batch using Scheduled Report Maintenance
 - Delete one of the scheduled reports from the batch
 - Change some priority values to change order of running
 - Run batch again and see if the behavior is as expected



RF-ADMIN-300

- 1 Create a new batch ID
 - Batch ID Maintenance
- 2 Create several scheduled reports in the batch
 - Run report from menu, use Schedule button
 - Make sure some have RunOnce=false, SaveFile=true
 - Use View Schedule button to review
- 3 Configure and run a scheduled report server process
 - a A) From DOS prompt
 - b B) From Windows Task Scheduler
 - Start > Control Panel > Scheduled Tasks
 - Monitor batch runs using Scheduled Report Browse and Scheduled Report History Browse
- 4 Modify batch using Scheduled Report Maintenance
 - Delete one of the scheduled reports from the batch
 - Change some priority values to change order of running
 - Run batch again and see if the behavior is as expected

Scheduling Report from API

Scheduling Report from API

- Schedule report from progress 4GL or .NETUI code
 - Scheduled Report API
 - .NET Run-Report API
- Scheduled Report API uses QAD Reporting Framework Source
- .NET Run-Report API used to invoke reports synchronously from within a .NET UI Program
- Scheduled Report API requires Service Interface Layer (available only in EE)



RF-ADMM-310

Scheduled Report API

In addition to scheduling reports from the user interface, an API is available that can schedule reports programmatically from Progress 4GL or .NET code. If you are interested in doing development with this API, you should order the free QAD product called QAD Reporting Framework Source, which contains API supporting materials and examples and other resources useful for report developers.

In addition to the Scheduled Report API, there is also the .NET Run-Report API, which can be used to invoke reports synchronously from within a .NET UI program.

Note The Scheduled Report API requires the Service Interface Layer, which is only available in Enterprise Edition.

Service Mode

Service Mode

- QAD Reporting Framework Service (Windows Service)
- Runs virtual batch called QADSVC
- Essential for Report Bursting
- Started using the standard Windows Service Manager tool
- Broker launched by (QADReportingFrameworkService.exe)
- Agents by (QAD.Client.exe)



RF-ADMIN-320

Service mode is a continuously running Windows Service (called the QAD Reporting Framework Service) that runs reports as soon as they are scheduled. The service only processes reports that were scheduled with a special batch ID: a virtual batch named QADSVC. This special batch is called virtual because it does not need to be created using Batch ID Maintenance. However, you can create a batch named QADSVC using Batch ID Maintenance, in which case it appears on the list of available batch IDs in the scheduled report form (and is therefore visible to users).

The service is especially useful to support cases where the application that needs to run a report is not running in a QAD .NET UI process (for example, a Progress program running on a Linux machine). This service is also essential to the infrastructure of the new report bursting capability.

The service mode runs as a Windows Service on one or more Windows computers. Its architecture is similar to that of the batch mode: multiple server processes can be run on any number of machines for scalability and failover.

The QAD Reporting Framework Service is started using the standard Windows Services Manager tool (choose Control Panel|Administrative Tools). When started, a broker process is launched (QADReportingFrameworkService.exe), which in turn launches one or more agent processes (QAD.Client.exe). The agent processes perform the actual report execution. If the service is paused, the broker in turn pauses each agent. If the service is stopped, the broker terminates each agent process before terminating itself.

Note When stopping or pausing the service, any agents that are currently running reports are not interrupted; the stop or pause command takes effect after the currently running report is completed. If a long-running report needs to be canceled, this can be accomplished using the QadRFSAdmin program, which is described below.

The underlying mechanism used to process reports in the Reporting Service is similar to that used by batch mode report servers. The configuration settings for report e-mail notifications, printers, and output file routing are shared for use in both server modes. Likewise, Scheduled Report Browse and Scheduled Report History Browse can be used to monitor service mode activity in the QADSVC batch.

There is one important difference in the behavior of Service Mode compared to Scheduled Batch Mode operation: the Run Once setting is ignored for Service Mode reports because they are always run only once. When a Service Mode report has been run successfully, its scheduled report record gets deleted (and therefore disappears from view on the Scheduled Report Browse).

Note If a Service Mode report run fails, its status is changed to ERROR. Administrators can attempt a rerun of that report by changing the status back to NEW using Report Resource Maintenance. (They can navigate a link to this by right-clicking the ID field on Scheduled Report Browse.)

The installation of the QAD Reporting Framework Service is similar to that of a batch mode server since it is also bundled with the QAD .NET UI client installation. However, some additional configuration steps must be completed before starting the service, which are described in the next section.

The number of agent processes spawned by the service is specified in the configuration file. It is generally advised to have more than one agent running, so that if an agent is busy running a long report, other agents are available to process other reports simultaneously. If too many agents are running, the server machine may experience high CPU and/or memory usage, degrading its performance. If this occurs, modify the configuration file to define fewer agents and restart the service. For greater throughput and failover, several machines can be used, each one running a Reporting Framework Service against the same environment.

Note The Reporting Framework Service broker communicates with its agents using interprocess communication (IPC) channels. The name of the channel used for a particular agent is determined by the agent's name as chosen in the configuration file. These names must be unique on a given machine. When running Reporting Framework Services on multiple machines, however, it is OK to have the same agent name on different machines.

Setting Up QAD Reporting Framework Service

- 1 Install the QAD .NET UI client on the report server machine, which must be a Windows machine. Test the install by running the QAD .NET UI client and logging in.
- 2 Register the QAD Reporting Framework Service as a Windows Service:
 - Open a DOS command window (choose Start|Run, enter cmd, and click OK).
 - Navigate (cd) to the qad.plugin.reportserver plug-in directory (under the .NET installation plug-ins directory; typically in a location such as C:\Program Files\QAD\QAD Enterprise Applications 2012 EE\plugins\qad.plugin.reportserver)

- Run `install-qad-reporting-service.bat`

The QAD Reporting Framework Service is now registered as a Windows Service.

3 Configure the server. This consists of entering some settings into the XML service configuration file:

- From a DOS window or File Explorer, go to the `config` subdirectory under the `qad.plugin.reportserver` plug-in directory (typically in a location such as `C:\Program Files\QAD\QAD Enterprise Applications 2012 EE\plugins\qad.plugin.reportserver\config`).
- Make a copy of the example XML service configuration file for editing: copy `QadReportingFrameworkServiceConfig.example.xml` to `QadReportingFrameworkServiceConfig.xml`.
- Edit `QadReportingFrameworkServiceConfig.xml` in a text editor (such as Notepad). It has two main sections: one to configure the broker, and one to configure one or more agents that the broker manages. Each agent is a process that actually renders reports; it is recommended to have at least two agents for good throughput. An example of a two-agent configuration file is shown below:

```
<QadReportingFrameworkServiceConfig>
  <Broker>
    <Channel>qad.reporting.framework.service</Channel>
    <Log.File>QadReportingFrameworkService.log</Log.File>
    <Log.Level>INFO</Log.Level>
  </Broker>
  <Agents>
    <Agent>
      <Name>QADSVC-1</Name>
      <StartupParameters>
        <Config-Name>configuration_name</Config-Name>
        <User>user_id</User>
        <Password>(blank)</Password>
        <Workspace>USA.USACO</Workspace>
        <Enable>qad.plugin.financials</Enable>
      </StartupParameters>
    </Agent>
    <Agent>
      <Name>QADSVC-2</Name>
      <StartupParameters>
        <Config-Name>configuration_name</Config-Name>
        <User>user_id</User>
        <Password>(blank)</Password>
        <Workspace>USA.USACO</Workspace>
        <Enable>qad.plugin.financials</Enable>
      </StartupParameters>
    </Agent>
  </Agents>
</QadReportingFrameworkServiceConfig>
```

The file entries should be edited as follows:

- All of the `<Broker>` settings can be the same default values as in the above example.
- The `<Agent>` `<Name>` values can be the same as in the above.
- The `<Agent>` `<StartupParameters>` must be modified to point to the QAD .NET UI server system. The details are as follows:
 - `<Config-Name>` is the name of the desired configuration to connect to (same as the drop-down list at the bottom of the QAD .NET UI login screen).
 - `<User>` is the QAD .NET UI user ID for the service to connect as.

- <Password> is the QAD .NET UI password for the service to connect with (if null, use (blank), as in the above example).
- <Workspace> is the workspace key (*Domain.Entity*).
- The line containing <Enable>qad.plugin.financials</Enable> must be present for Enterprise Edition (EE) systems, and must be deleted for Standard Edition (SE) systems.

4 Start the service:

- Open the Windows Service manager program (choose Control Panel|Administrative Tools|Computer Management, and then choose Services And Applications|Services from the left pane).
- A list of Windows Services is displayed on the right pane. Locate the service called QAD Reporting Framework Service, right-click on it, and select Properties.
- When the Properties form appears, select the Log On tab and select the option for logging on as This Account. Enter the account username (for instance, QAD\username) and password. Click OK to save the changes; the Properties form closes.
- Right-click on the QAD Reporting Framework Service line and select Start to start the service. It should become fully operational for report processing within about 20 seconds. Similarly, you can later select Stop or Pause to stop or pause the service.

5 Inspect the log files for proper startup. The log files are located in the

qad.plugin.reportserver subdirectory called logs (typically in a location such as C:\Program Files\QAD\QAD Enterprise Applications 2012 EE\plugins\qad.plugin.reportserver\logs).

- Open the broker log file (QadReportingFrameworkService.log) and check for errors.
- Open the agent log files (for instance, QADSVC-1.log, QADSVC-2.log) to see if any agent errors have occurred. These look like standard QAD .NET UI log files.
- Check service status details:
 - From a DOS window, cd to the qad.plugin.reportserver plug-in directory (under the QAD .NET UI client installation plug-ins directory; typically in a location such as C:\Program Files\QAD\QAD Enterprise Applications 2012 EE\plugins\qad.plugin.reportserver)
 - Run QadRFSAdmin.exe Status and you will see a status report similar to that shown below. If the broker and agent statuses are all running, then the system is responding as expected. If one or more agents have a status of not responding in the log file, it could mean a problem has occurred. Try checking the status again a few seconds later. If it is still not correct, double-check the log files.

```
>QadRFSAdmin.exe Status
Broker Status:
  Broker Status : RUNNING
  Process ID    : 1848
  IPC Channel   : qad.reporting.framework.service
  Config File   : C:\Program Files\QAD\QAD Enterprise Applications 2012
EE\plugins\ReportFramework\Build\Debug\plugins\qad.plugin.reportserver\config\QadReporting
FrameworkServiceConfig.xml
  Log File      : C:\Program Files\QAD\QAD Enterprise Applications 2012
EE\plugins\ReportFramework\Build\Debug\plugins\qad.plugin.reportserver\logs\QadReportingFr
ameworkService.log
  Log Level     : DEBUG
  Agents (2):
```

```

Agent Status           : RUNNING
Agent Process ID      : 2952
Agent IPC Channel     : qad.reporting.framework.service.QADSV-1
Agent Start Time      : 10/25/2011 1:06:04 PM
Agent Memory Info     : 48066 K (Mem Usage), 42573 K (VM Size)
Agent Last Report Action: Started listening at 2011-10-25 13:06:04
Agent Pending Request : PENDING_REQUEST_NONE
Agent Command Line    : C:\Program Files\QAD\QAD Enterprise Applications 2012
EE\plugins\ReportFramework\Build\Debug\container\QAD.Client.exe -
param.url:file:///C:/Program Files/QAD/QAD Enterprise Applications 2012
EE/plugins/qad.plugin.reportserver/config/QADSV-1.pf

```

```

Agent Status           : RUNNING
Agent Process ID      : 5096
Agent IPC Channel     : qad.reporting.framework.service.QADSV-2
Agent Start Time      : 10/25/2011 1:06:06 PM
Agent Memory Info     : 50003 K (Mem Usage), 42471 K (VM Size)
Agent Last Report Action: Started listening at 2011-10-25 13:06:06
Agent Pending Request : PENDING_REQUEST_NONE
Agent Command Line    : C:\dev-
trunk\plugins\ReportFramework\Build\Debug\container\QAD.Client.exe -
param.url:file:///C:/Program Files/QAD/QAD Enterprise Applications 2012
EE/plugins/qad.plugin.reportserver/config/QADSV-2.pf

```

6 Test the service by scheduling a report to be processed by the service.

- Log on to the QAD .NET UI system.
- Open any report in the system that uses the QAD Reporting Framework. If none are on the menu, open Report Resource Designer, open any report, and then click the Preview button to run it from the designer.
- Click the Schedule|New tool button to schedule the report:
 - Enter any *Description*.
 - Enter a blank *Batch ID* (this defaults to the virtual service batch QADSV-1).
 - Leave *Printer* blank for basic testing purposes. If printer testing is desired, reporting printers must be defined in the QAD .NET UI home server configuration file (`client-session.xml`) before they appear on this form.
 - Check the *Save Report Output* option to have the scheduled report output saved on the web server.
 - *Run Only Once* can be selected or not since it is ignored (all service-mode servers run the reports only once regardless of this setting).
 - An *Email* address can be specified, but e-mailing will only succeed if SMTP has been set up on the .NET UI home server configuration file (`client-session.xml`).
- Click the Schedule|View Schedule tool button to check the report's status in the batch (it will exist in the batch queue until it has been run successfully).
- Click the Schedule|View History tool button to check the report run results (which will not exist until the report has finished being run). Right-click on the URL column and select Scheduled Report Result to drill down to the output file, which is displayed in the QAD .NET UI below the history browse grid.
- If any errors are encountered, or the report shows up in the queue but never gets run, check the reporting service log files as described in step 5 on page 215.

Administering the Reporting Framework Service

Routine administration of the service can be done using the standard Windows Service Manager tool, which allows functions such as start, stop, pause, and resume. It also allows the service to be automatically started at machine startup and other options.

There is also a QAD administrative utility that provides finer control over the service. It is a command line program called `QadRFSAdmin.exe` that is typically run from a DOS command prompt. It is located in the report server plug-in folder under the QAD .NET UI client installation folder. For example:

```
C:\Program Files\QAD\QAD Enterprise Applications\plugins\  
qad.plugin.reportserver
```

To run the command line administration program, open a DOS command window, cd to the above folder, and then run `QadRFSAdmin.exe`.

The usage is as follows:

```
QadRFSAdmin [-f <config file>] <command> [param]  
Valid commands:  
QadRFSAdmin StartBroker  
QadRFSAdmin StopBroker  
QadRFSAdmin StartAgents  
QadRFSAdmin StopAgents  
QadRFSAdmin PauseAgents  
QadRFSAdmin ResumeAgents  
QadRFSAdmin StartAgent <agentChannelName>  
QadRFSAdmin StopAgent <agentChannelName>  
QadRFSAdmin KillAgent <agentChannelName>  
QadRFSAdmin PauseAgent <agentChannelName>  
QadRFSAdmin ResumeAgent <agentChannelName>  
QadRFSAdmin CancelReport <agentChannelName>  
QadRFSAdmin Status
```

The `QadRFSAdmin` program connects to the service broker using the same XML service configuration file as the service itself. If the service was configured using a non-default configuration file name or path, then the `-f` option can be used to point the `QadRFSAdmin` program to that file.

Note If the Service was started under the Local System account instead of under a specific user account, then the `QadRFSAdmin` program will not be able to connect to the broker due to an IPC security restriction, giving an error message like this

```
> QadRFSAdmin status  
Broker Status:  
Error: Failed to connect to an IPC Port: Access is denied.
```

To resolve this issue, go to the Windows Services Manager program, select the QAD Reporting Framework service's Properties, go to the Log On tab, and select "This Account" and enter a Windows username and password for the account to run the service under, which must have administrative rights.

The `QadRFSAdmin` program provides a number of administrative commands as shown in the above usage output. The commands that affect a specific agent require an additional parameter specifying the agent's channel name. This channel name is defined as the broker's channel name with the agent name appended at the end after a dot (.). All of these channel names can be clearly seen in the output of the `QadRFSAdmin Status` command. For example:

```

>QadRFSAdmin.exe Status
Broker Status:
  Broker Status : RUNNING
  Process ID   : 1848
  IPC Channel  : qad.reporting.framework.service
  Config File  : C:\Program Files\QAD\QAD Enterprise Applications 2012
EE\plugins\ReportFramework\Build\Debug\plugins\qad.plugin.reportserver\config\QadReporting
FrameworkServiceConfig.xml
  Log File     : C:\Program Files\QAD\QAD Enterprise Applications 2012
EE\plugins\ReportFramework\Build\Debug\plugins\qad.plugin.reportserver\logs\QadReportingFr
ameworkService.log
  Log Level    : DEBUG
  Agents (2):

  Agent Status      : RUNNING
  Agent Process ID  : 2952
  Agent IPC Channel : qad.reporting.framework.service.QADSVC-1
  Agent Start Time  : 10/25/2011 1:06:04 PM
  Agent Memory Info : 48066 K (Mem Usage), 42573 K (VM Size)
  Agent Last Report Action: Started listening at 2011-10-25 13:06:04
  Agent Pending Request : PENDING_REQUEST_NONE
  Agent Command Line : C:\Program Files\QAD\QAD Enterprise Applications 2012
EE\plugins\ReportFramework\Build\Debug\container\QAD.Client.exe -
param.url:file:///C:/Program Files/QAD/QAD Enterprise Applications 2012
EE/plugins/qad.plugin.reportserver/config/QADSVC-1.pf

  Agent Status      : RUNNING
  Agent Process ID  : 5096
  Agent IPC Channel : qad.reporting.framework.service.QADSVC-2
  Agent Start Time  : 10/25/2011 1:06:06 PM
  Agent Memory Info : 50003 K (Mem Usage), 42471 K (VM Size)
  Agent Last Report Action: Started listening at 2011-10-25 13:06:06
  Agent Pending Request : PENDING_REQUEST_NONE
  Agent Command Line : C:\dev-
trunk\plugins\ReportFramework\Build\Debug\container\QAD.Client.exe -
param.url:file:///C:/Program Files/QAD/QAD Enterprise Applications 2012
EE/plugins/qad.plugin.reportserver/config/QADSVC-2.pf

```

In the above example, the channel name for the first agent is:

```
qad.reporting.framework.service.QADSVC-1
```

The QadRFSAdmin commands are described below:

- QadRFSAdmin StartBroker — Starts a Reporting Framework Service broker process, which in turn spawns agent processes. This is essentially the same thing that occurs if the service is started using the Windows Service Manager, except that the Windows service layer is bypassed. It is generally advisable to use the Windows Service Manager instead of this command line function.

Note This command should not be run if the service has already been started using the Windows Service Manager.

Note After this command is run, the DOS command window is unusable until the service is stopped. The DOS window should not be closed until the service has been stopped using the StopBroker command; otherwise the agent processes remains running and orphaned. Should this occur, however, the agent processes (QAD.Client.exe) can be manually killed using Windows Task Manager.

- QadRFSAdmin StopBroker — Stops a Reporting Framework Service broker process, which in turn gracefully terminates its agent processes. This is essentially the same thing that occurs if the service is stopped using the Windows Service Manager, except that the Windows service layer is bypassed. It is generally advisable to use the Windows Service Manager instead of this command line function.

Note This command should not be run if the service was started using the Windows Service Manager.

- `QadRFSAdmin StartAgents` — Starts a new pool of service agents. Has no effect if the agent pool has already been started.
- `QadRFSAdmin StopAgents` — Stops and gracefully terminates each agent in the pool. The broker process is still kept alive.
- `QadRFSAdmin PauseAgents` — Pauses each agent in the pool. Each agent process remains alive but report processing is suspended. Only has effect if the agents are running.
- `QadRFSAdmin ResumeAgents` — Resumes each agent in the pool. Only has effect if the agents are paused.
- `QadRFSAdmin StartAgent <agentChannelName>` — Starts the specified agent process. Only has effect if that agent process is not already running.
- `QadRFSAdmin StopAgent <agentChannelName>` — Stops and gracefully terminates the specified agent process. If the agent is currently running a report, it is not stopped until that report finishes.
- `QadRFSAdmin KillAgent <agentChannelName>` — Similar to `StopAgent`, except that if the graceful termination does not succeed after a few seconds (for instance, if the agent is processing a long-running report or is frozen for some reason), then an ungraceful process kill is initiated to force termination of the agent.

Note If a kill is being attempted to stop an agent that is running a long report query, the `CancelReport` command should be issued first to cancel the server resources being used for the query. In many such cases, the `CancelReport` command will suffice to restore the agent back to proper running order.

- `QadRFSAdmin PauseAgent <agentChannelName>` — Pauses the specified agent in the pool. The agent process remains alive but report processing is suspended. Only has effect if the agent is running.
- `QadRFSAdmin ResumeAgent <agentChannelName>` — Resumes the specified agent in the pool. Only has effect if the agent is paused.
- `QadRFSAdmin CancelReport <agentChannelName>` — Instructs the specified agent to cancel the report it is currently running. Only has effect if the agent is running the report.

Note This command is an essential tool for use in situations where a report run results in a long-running query that must be terminated to restore server performance. The cancellation should free up database and application server query resources after a few seconds.

- `QadRFSAdmin Status` — Provides a detailed description of the current state of the QAD Reporting Framework Service, including the status of the broker and each agent. An example of its output is show above.

Note If the Windows Service Manager is being used to start the QAD Reporting Framework Service, then the `StartBroker` and `StopBroker` commands should not be used. However, all of the other commands are OK to use in this case.

Setting up QAD Reporting Framework Service

Setting up QAD Reporting Framework Service

- Install QAD .NET UI Client on the Report Server
- Register QAD Reporting Framework Service as a Windows service
- Open DOS command window
- Configure the server
- Set the XML service configuration file
- Start the service
- Inspect the log files for proper startup
- Test the service by scheduling a report
- For details, see the [Reporting Framework User Guide Scheduled Batch Mode section](#)



RF-ADMIN-330

Scheduled Batch Mode

You can automate the process of generating routine reports by scheduling them to automatically run at specified times or intervals and have the reports sent to a specified destination, such as a printer or the document service on the report server.

To schedule reports to run at a specified time or interval, on the report server, you create a Windows scheduled task for a batch and group the reports in the batch.

The Windows Task Scheduler should be configured to launch the Report Batch Processor, which is a non-GUI instance of the QAD .NET UI launched from the command line, for a specific batch as scheduled and runs all the scheduled reports grouped in the batch. If already set up, the report outputs are sent to the QAD .NET UI document service and/or server-side printer as configured.

Multiple report servers can be set up for increased throughput and failover. The different servers can be configured to process different batches, or can even jointly process reports in the same batch. The Report Batch Processor coordinates the processing of scheduled reports with different priorities in the correct sequence across multiple report servers.

Scheduled reports have the following additional features compared to reports run in Report Viewer:

- The output file (PDF/Excel) of a scheduled report can be uploaded to the document service so that the user can view it in the QAD .NET UI later.

- E-mail notifications, including SMTP mails and inbox messages embedded in the QAD .NET UI. You should specify e-mail addresses and the inbox user IDs when creating scheduled reports. When a report link is included in the e-mail notification to a scheduled report, the link is a direct link to the report file on the server. This direct link launches the report in a browser. However, for QAD .NET UI inbox messages, the link is a QAD Shell URL (<http://qadsh>) link that launches the report in the QAD .NET UI.
- Server-side printing. You can specify the printer that the report server uses to print the output file.
- Scheduled reports are maintained by the administrator from the maintenance program. The administrator controls the running sequence of scheduled reports by modifying their priorities.

Set Up a Scheduled Batch

- 1 Create a batch in Batch ID Maintenance (36.14.1).
- 2 On the report server, create a scheduled task for the batch through Windows Task Scheduler.
 - a Create a parameter file to contain command line parameters with fixed values. Use the following `params.pf` file as an example:

```
-silent
-config-name:test
-user:mfg
-password:(blank)
-workspace:Domain1.1000
-report-batch:batch1
-enable:qad.plugin.services
-enable:qad.plugin.reports
-enable:qad.plugin.reportserver
-enable:qad.plugin.financials
-report-mode:batch
```

Note You must specify `-enable:qad.plugin.financials` if you are using QAD Enterprise Applications — Enterprise Edition. You must not specify `-enable:qad.plugin.financials` if you are using QAD Enterprise Applications — Standard Edition.

You need to set your own desired values for these parameters:

- `-config-name`: The name of the configuration that the report server should log on to. This is the same as the value chosen by an end user from the drop-down list in the login screen of the QAD .NET UI application when run in GUI mode.
 - `-user`: User ID for logging on to the QAD .NET UI system
 - `-password`: Password for logging on to the QAD .NET UI system
 - `-workspace`: The workspace key of the desired workspace to run scheduled reports in. This is important, since any batch queue is specified by a unique combination of domain and batch ID, and the domain that the report server will use is the domain associated with the specified workspace key.
 - `-report-batch`: The batch ID that will be used in conjunction with the domain associated with the specified workspace key to determine the batch of reports to run.
- b In the launching script of the report server process, use the parameter file in place of the parameters:

```
QAD.Client.exe -param.url:file:///c:/params.pf
```

If the parameter file is referenced by a URL, you can choose to place the file on the local machine or a report server.

```
QAD.Client.exe -param.url:http://localhost/rpt/params.pf
```

Note The language that translations will be done is in the language of the user who scheduled the report, regardless of the server user's language.

Setting Up E-Mail Notifications

To set up e-mail notifications, add the following entries to `client-session.xml` on the home server:

```
<Configuration>
...
<!-- SMTP server host name -->
<Smtp.Host>SMTPHostname</Smtp.Host>
<!-- SMTP port name -->
<Smtp.Port>SMTPPortNumber</Smtp.Port>
<!-- SMTP from email address -->
<Smtp.From>E-Mail</Smtp.From>
<!-- SMTP username -->
<Smtp.Username>SMTPUsername</Smtp.Username>
<!-- SMTP password -->
<Smtp.Password>SMTPPassword</Smtp.Password>
<!-- SMTP use SSL -->
<Smtp.UseSSL>>false</Smtp.UseSSL>
</Configuration>
```

Use the following settings as a reference:

```
<Smtp.Host>smtp.qad.com</Smtp.Host>
<Smtp.Port>25</Smtp.Port>
<Smtp.From>Report Server <johndoe@qad.com></Smtp.From>
<Smtp.Username>admin</Smtp.Username>
<Smtp.Password>123</Smtp.Password>
<Smtp.UseSSL>>false</Smtp.UseSSL>
```

Modifying the Scheduled Report e-mail Template

When a scheduled report is run by the report server, the system sends an e-mail notification if any e-mail addresses were specified when the report was scheduled. The email can optionally contain the report output file as an attachment if this was specified at the time of scheduling.

The system uses an *e-mail template* to determine the subject and body of the report notification e-mails. This template is stored in a special file called `report-email-template.txt` on the QAD .NET UI home server. The file can be modified if administrators want to change the default content.

Note The e-mail template can be overridden for a specific scheduling of a report by setting the `sys_email_template` report value to contain the desired template string. This is possible when scheduling a report programmatically using the Scheduled Report API.

The file is located on the home server as follows:

```
.../webapps/qadhome/configurations/<configuration-name>/storage/reports/
report-email-template.txt
```

The default content of `report-email-template.txt` is as follows:

```
[ SUBJECT ]
```

Scheduled Report Completed: {\$RRO_DESC}
[BODY]

A scheduled report from QAD Enterprise Applications has completed:

Report: {\$RRO_DESC} ({\$RRO_CODE})
Description: {\$SR_DESC}
Link to Report: {\$REPORT_FILE_LINK}

The structure of the e-mail template consists of two parts:

- A [SUBJECT] tag followed by the subject content on the next line, which is used as the subject of the email notification.
- A [BODY] tag followed by any number of lines of text, which is used for the body of the e-mail.

Both the subject and body text can contain both literal text portions as well as dynamic variable references. The dynamic variables get resolved at run-time and are specified using the same syntax as for dynamic variables in scheduled report file routing. In addition to the variables allowed for file routing, there is another variable used only for e-mail templates: \$REPORT_FILE_LINK, which gets resolved at run-time to a URL string that points to the report output file on the home server (if file linking or attaching was specified at the time of scheduling).

Setting Up a Printer

Note Starting with the QAD .NET UI 2.9.2 (Enterprise Applications 2010.1 EE) release, printer setup for scheduled reports has changed. The Printer Setup Maintenance program is no longer used for the Reporting Framework. Instead, use the following steps to set up a printer for scheduled reports. If you are upgrading from an older version, any scheduled reports that refer to the previous printer types will still execute properly. However, the following steps must still be done to allow printers to be defined for new scheduled reports.

- 1 Set up a physical printer on the report server. From the Windows Start menu, select Control Panel|Printers and Faxes|Add a Printer to add a printer.
- 2 In the client session configuration file (`client-session.xml`), located in the `TomcatInstallDir/webapps/qadhome/configurations/SysEnvName/` directory, set up printers available for scheduled reports as follows:

```
<ReportServer>
  <Printer>
    <UNCPath>\\machineA\printerA</UNCPath>
    <Description>Description of printer (optional)</Description>
  </Printer>
  <Printer>
    <UNCPath>\\machineB\printerB</UNCPath>
    <Description></Description>
  </Printer>
</ReportServer>
```

Setting up a Scheduled Report Default Printer

You can choose the default printer for the Schedule Report screen from Tools|Options. The list of printers displayed in the Options window is the same as the list in the Schedule Report screen—the list specified in the `client-session.xml` file. If no default is chosen, the Windows default printer is used as the default if the Windows default printer is one of the printers specified in the `client-session.xml` file.

The format to specify the default printer in `client-session.xml` is to set `default="true"` in the `<Printer>` tag, as shown in the following example:

```
<ReportServer>
  <Printer default="true">
    <UNCPath>\\server_name\printer_name</UNCPath>
    <Description>My Default Printer</Description>
  </Printer>
  ...
</ReportServer>
```

Configuring Output File Naming and Location

Administrators can now configure flexible, dynamic routing rules to control report output file names and path. The rules can be set up with defaults to handle most general types of reports, as well as specific alternatives based on the report type and layout. For example, a certain type of report containing sensitive data could be funneled to a special folder that has Web access disallowed.

In addition to configuring such report routing rules on a system-wide basis, administrators can also override the file name and path rules for any specific scheduling of a report. For example, for a certain report that gets run every month in batch, a special naming convention and folder path can be assigned for that per-month batch run of that report, overriding any general rules that may have been configured for the same report type.

In `client-session.xml`, you specify the file routing in the `<ReportServer>` section.

The `<ReportOutputFileRoute>` element's `reportCode` and `layout` attributes specify the report codes and layouts to which the file routing is applied. If `reportCode` and `layout` are both blank, the specified file routing is applied to all reports and layouts. If additional `<ReportOutputFileRoute>` entries are added that specify report codes and layouts, these entries override the general entry for reports matching the more specific entries.

The rules that govern output file naming and directory path are specified in two child elements of `<ReportOutputFileRoute>`:

- `<OutputFilePath>` specifies the directory path for the file, relative to the reports directory on the home server document storage area (`qadhome/configurations/<config_name>/storage/reports/`).
- `<OutputFileName>` specifies the file name.

Both of the above entries allow dynamic variable substitutions, which get resolved at run-time. The syntax `{<variable_name>}` is used, where `variable_name` includes the following:

- RRO_CODE — specifies the report code (for example, QAD_DaybookSetReport) as defined in Report Resource Maintenance.
- RRO_DESC — the description of the report as defined in Report Resource Maintenance.
- SR_ID — the scheduled report identifier, which gets assigned whenever a report gets scheduled.
- SR_DESC — the scheduled report description.
- DATE_TIME — shorthand for the DATE_TIME_yyyyMMddhhmmssffffff dynamic variable.
- USER_ID — the user ID of the user who scheduled the report.

Any report setting can also be used as a variable name, in which case the run-time value of that setting is dynamically substituted. If no setting is found that matches the variable name, a blank is substituted.

REPORT_FILE_LINK — (only available for the e-mail template)

Additionally, there is a special dynamic variable that can be used with the output file name entry that resolves to a date-time stamp. This is very important to use for file naming, since a file name that matches one that already exists in the same directory from a previous report run results in a file overwrite. The format of this date-time variable is:

DATE_TIME_<format_specifier>

where <format_specifier> is a .NET date-time format string. For example,

DATE_TIME_yyyyMMddhhmmssffffff

will specify the date-time including year, month, day, hour, minutes, seconds, and fractions of a second to six digits. (It is important to include the fractions of a second portion to prevent file name conflicts in cases where multiple report servers are running.)

An example output file name is: 20120209101213083857.pdf.

You can specify DATE_TIME without a format specifier. In this case, it defaults to the yyyyMMddhhmmssffffff format.

Both file path and file name specifiers can mix any combination of literal text and dynamic variables. For example,

```
<OutputFileName>daybook-{$DATE_TIME_yyyy-MM-dd-hhmmss-ffffff}</OutputFileName>
```

will create output files whose names always start with daybook- followed by a dynamic time stamp.

The following example, which includes three <ReportOutputFileRoute> entries, illustrates how the routing rules can be applied:

```
<ReportServer>
...
<ReportOutputFileRoute reportCode="" layout="">
  <OutputFilePath>new-results/{SYS_DOMAIN}/{USER_ID}/{RRO_CODE}</OutputFilePath>
  <OutputFileName>{$DATE_TIME_yyyyMMddhhmmssffffff}</OutputFileName>
</ReportOutputFileRoute>

  <ReportOutputFileRoute reportCode="QAD_DaybookSetReport" layout="">
    <OutputFilePath>new-results/daybook</OutputFilePath>
    <OutputFileName>daybook-{$DATE_TIME_yyyy-MM-dd-hhmmss-ffffff}</OutputFileName>
  </ReportOutputFileRoute>
```

```
<ReportOutputFileRoute reportCode="QAD_DaybookSetReport" layout="layout1">
  <OutputFilePath>new-results/daybook-special</OutputFilePath>
  <OutputFileName>daybook-special-{$DATE_TIME}</OutputFileName>
</ReportOutputFileRoute>
...
</ReportServer>
```

In the above example, the first entry's routing rules will be applied to all reports other than QAD_DaybookSetReport. The second entry's routing rules will be used for QAD_DaybookSetReport, except when the layout is layout1, in which case the third entry's rules will be applied.

Note Administrators can override the above file routing rules for a particular scheduled report if desired. When scheduling from the Schedule Report form in the QAD .NET UI, custom file name and file path strings can be entered. These fields are only exposed to users belonging to the rptAdmin role.

Launching Reports from Progress CHUI

Launching Reports from Progress CHUI

- Introduced in 2012 EE (QAD .NET UI 2.9.5)
- Uses Schedule Report API
- Progress program invokes new QAD.NET UI tab
- Tab contains the report viewer, which displays the report output



RF-ADMM-340

Prior to QAD .NET UI 2.9.5 (2012 EE), there were limitations in the ability for a Progress program to be able to automatically launch a QAD Reporting Framework report. The fundamental issue was that the report must be rendered in a QAD .NET UI process on a Windows machine, and Progress programs are not run in this environment. The Scheduled Report API (introduced in QAD 2010.1 EE) first opened the door to this possibility, allowing the Progress program to call the API to allow the report to be scheduled to run on a (Windows) report server in batch. This approach still had major limitations: a time delay between the time at which the report is scheduled and when it later gets run on the server, and also a limited ability for the user of the Progress program to have access to the report output.

Important Because some maintenance programs in the 2012 EE release use this mechanism to invoke reports, it is necessary to install, configure, and run the QAD Reporting Framework Service in order for these programs to run properly.

The service mode provides the basis for a solution of the time delay: by scheduling the report to be run in the virtual immediate batch ID, the report is run typically within a matter of seconds from the time it was scheduled.

Furthermore, a new mechanism was introduced into the QAD .NET UI such that if the Progress program is initiated from a QAD .NET UI session, it can now invoke the launching of a new QAD .NET UI tab containing a report viewer that will display the report output to the user. The viewer runs in a mode where it polls the report server for the output of the scheduled report. Once the report has completed on the server, the viewer then automatically displays it for the user.

Note If the Progress program is not launched from within the QAD .NET UI (for example, on a terminal), then it can still launch the report but will not be able to view the output. However, the report output can still be saved on the Web server or sent to a printer.

If you are interested in doing development to invoke reports from Progress programs, you should order the free QAD product called QAD Reporting Framework Source, which contains API supporting materials and examples and other resources useful for report developers, including launching reports and report viewers from Progress programs.

For more information, see [QAD Reporting Framework User Guide](#). Additionally, if you have internal QAD access to the QAD Developers Network (QDN), see: *How can I invoke QAD Reporting Framework reports from my CHUI Progress program?* (<http://qdn.qad.com/x/VIOoAQ>).

Report Bursting

Report Bursting

- Mass running and distribution of reports
- Introduced in 2012 EE
- A report can be configured to automatically split a report into smaller reports
- A report can be configured to dynamically route each of the split output reports to:
 - E-mail
 - File
 - Printer



RF-ADMIN-350

Starting with the QAD 2012 Enterprise Edition, the Reporting Framework includes an infrastructure to facilitate the mass-running and distribution of reports: report bursting. A report burst is a special way that a report can be run that involves the following aspects:

- A report burst can be configured to automatically split a report into many smaller reports, relieving end users of the burden of manually running numerous reports. Any field in the top-level table of the report's data set can be chosen as the split field (for example, a report could be split to output one report per customer ID, or one report per item number).
- A report burst can be configured to dynamically route each of the split output reports to different e-mail, file, and printer destinations. The logic can be based on data values (for example, the customer ID in each of the output reports could be mapped to an e-mail address for that customer to send the report to).

The report bursting mechanism is a general capability that can be used with any report developed using the QAD Reporting Framework, but not for any other type of report. In addition to dynamic setting of output destination, the infrastructure allows for most of the report settings to be set dynamically based on the data; this includes such settings as language for translated labels, date and number formats for internationalization, output file type (PDF, Excel, RTF, and so on) and layout type (for example, different form layouts of the same data for different countries).

Report bursts internally schedule each of the split output reports to be asynchronously (but immediately) run by a report server using the QAD Reporting Framework Service. This leverages the many benefits of the report server architecture such as robustness, scalability, and failover.

Although the new bursting capability is included in the QAD Enterprise Applications 2012 – Enterprise Edition release, this release does not contain any QAD application programs that take advantage of it. You can, however, set up your own desired scenarios for bursting using the following mechanisms:

- The Scheduled Report API can be used to write programs to schedule report bursts to be run on the server at desired times. Special burst settings are used to configure the burst run. These programs can be written in the Progress or .NET languages. They could be either simple script-like utility programs for administrators to run, or could be fronted with user-interface logic to expose the functionality to end users if desired.
- The .NET Run-Report API can be used to write programs to initiate report bursts immediately within a QAD .NET UI process. The programs could be fronted with user-interface logic to expose the functionality to end users if desired.
- Administrators can use a new Burst Settings tool button from the report viewer program to run an immediate ad-hoc burst of that report, and also to choose settings and then schedule the report to be run as a burst in batch on a report server.

The logic that controls the dynamic routing settings is completely configurable. However, the QAD Enterprise Applications 2012 – Enterprise Edition release contains no default routing logic. If dynamic routing is desired, it is necessary to first codify the needed rules using the Progress programming language in a special dynamic-routing program that can be invoked during report bursts.

If you want help with creating report burst programs or setting up the desired routing rules, please consult QAD Services.

Understanding Report Bursting Technology

Prior to the introduction of QAD report bursting technology, third-party solutions were required. Those products typically process a plain text report output file, parse it, split it into multiple files, transform it to alternate output formats, and route the output files to destinations such as document storage and e-mail. They typically allow dynamic routing logic to be configured based on the data in the document.

The introduction of the QAD Reporting Framework posed new challenges as well as new possibilities for such report bursting approaches. Its ability to output rich, graphical reports in binary formats such as PDF eliminate the need for third-party post-rendering, and also create a more challenging file format to feed into those products that cannot be parsed as easily as plain text.

The QAD report bursting capability now provided with the QAD Reporting Framework solves these challenges in an out-of-the-box manner that eliminates the need for third-party technology. It uses a fundamentally different approach that does not involve post-processing of report output files. Instead, report bursting is implemented internally as the report is run. It is important to understand the basic process used in order to be able to configure and administer report bursts.

When a report is run in burst mode, it operates as a *driver report*, which, during its processing, spawns one or more *target reports*, each of which performs the rendering of the final output documents. The driver report first queries the dataset for the overall burst and splits this into a smaller dataset for each target report. The target reports are not directly run in the driver report process but instead are scheduled to be run in the QADSVCS batch that is processed continuously

by the QAD Reporting Framework service running on one or more report server machines. This provides benefits including robustness, scalability, and increased throughput. It also records a permanent record of the report burst results, which can be viewed using Scheduled Report History Browse.

For more information on report bursting with Dynamic Output Routing, see *QAD Reporting Framework User Guide*.

Summary

Summary

In this session you have learned how to:

- Configure the QAD Reporting Framework
- Set up user authorization for report development and admin programs
- Manage reports on the system menu
- Administer scheduled report servers
- Recent enhancements



RF-ADMIN-360

Product Information Resources

QAD offers a number of online resources to help you get more information about using QAD products.

[QAD Forums \(community.qad.com\)](http://community.qad.com)

Ask questions and share information with other members of the user community, including QAD experts.

[QAD Knowledgebase \(knowledgebase.qad.com\)*](http://knowledgebase.qad.com)

Search for answers, tips, or solutions related to any QAD product or topic.

[QAD Document Library \(www.qad.com/documentlibrary\)](http://www.qad.com/documentlibrary)

Get browser-based access to user guides, release notes, training guides, and so on; use powerful search features to find the document you want, then read online, or download and print PDF.

[QAD Learning Center \(learning.qad.com\)*](http://learning.qad.com)

Visit QAD's one-stop destination for all courses and training materials.

*Log-in required

