Industry-specific

QAD SOLUTIONS

Manufacturing Applications

Technical Reference
QXtend Outbound

Overview and Installation
Implementing and Using QXtend Outbound

78-0623A
MFG/PRO eB, eB2, and eB2.1
June 2005

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.

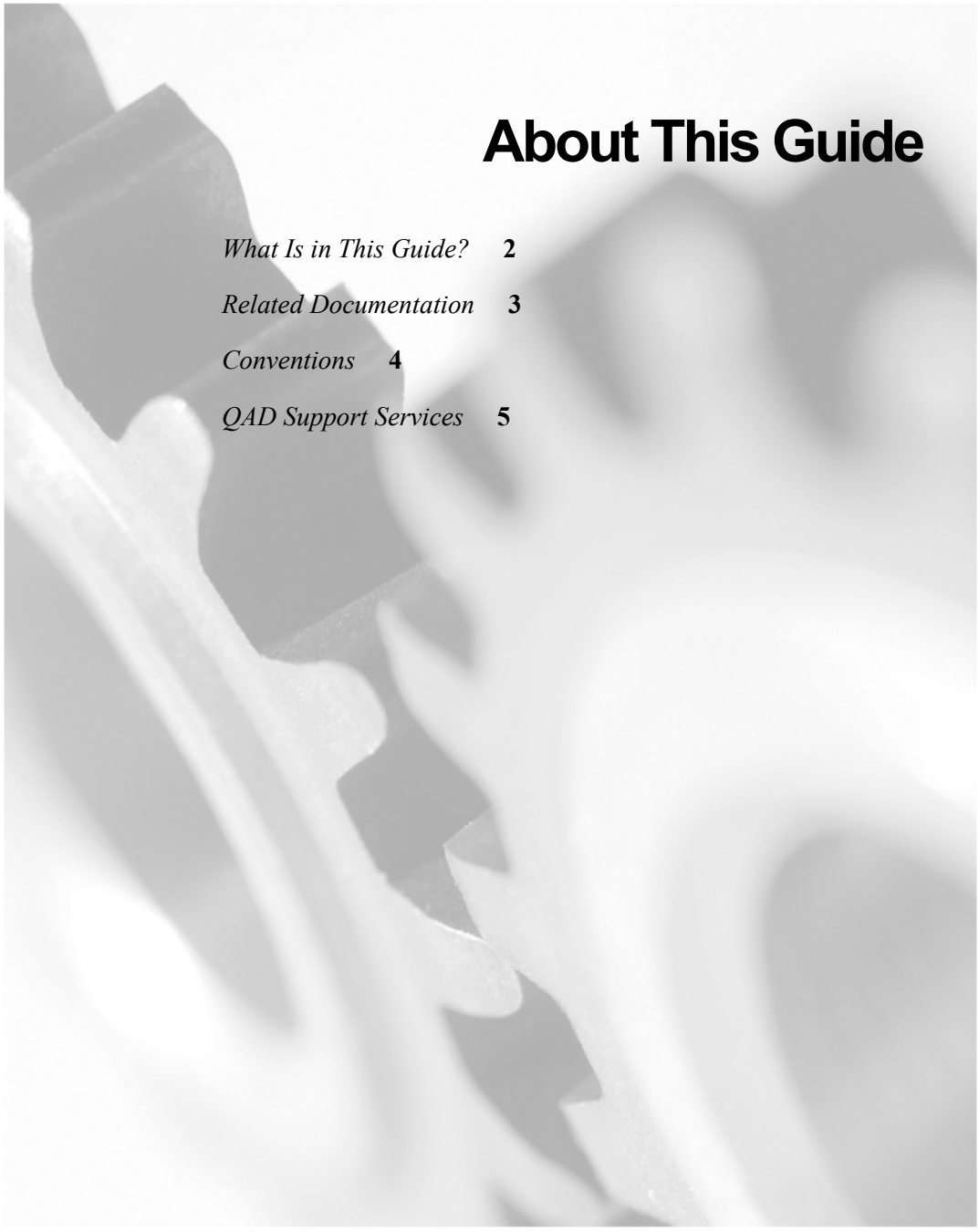
QAD and MFG/PRO are registered trademarks of QAD Inc. The QAD logo is a trademark of QAD Inc.

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 ©2005 by QAD Inc.
78-0623A

QAD Inc.
6450 Via Real
Carpinteria, California 93013
Phone (805) 684-6614
Fax (805) 684-1890
http://www.qad.com

Contents



About This Guide

What Is in This Guide?

Use this guide to install, configure, and use QXtend Outbound. The guide is divided into the following sections:

See page 7.

1 Overview and Installation

This section contains an overview of QXtend Outbound and its various components as well as instructions on how to install QXtend Outbound.

See page 71.

2 Implementation and User Information

This section includes information about implementing QXtend Outbound, managing business objects, and using the product to track QDoc creation and delivery.

Audience

These instructions are intended for a system administrator with experience installing complex production applications, as well as configuring and managing hardware and operating system software. This person also should have a good understanding of networking concepts and administration, as well as an understanding of concepts and technologies discussed in “Application Overview” on page 10.

If you do not have this expertise within your company, contact QAD Support for information on the installation and customization offerings supplied by QAD’s Global Services.

Installation and Product Updates

Prior to starting any QAD installation, check the QAD Support Web site to make sure you have the latest installation errata, installation guides, and installation media.

http://support.qad.com/

Compare the item number or publication date listed on any documents you have with the number or date listed on the QAD Web site. If the document has been updated, download and use the most recent version.

You can also access revised or new QDocs from the QAD Support Web site.

Installation Media

The QXtend Outbound installation media is periodically updated with updates and enhancements. To ensure you have the most up-to-date QXtend Outbound installation media, compare the QXtend Outbound build number on your installation media with the build published on the latest errata sheet posted on the QAD Web site.

Related Documentation

Progress Documentation

- For information on installing and configuring the Progress AppServer, see *Building Distributed Applications Using the Progress AppServer*.

You can find the complete Progress documentation set online at:

http://www.progress.com/products/documentation/index.ssp

http://www.progress.com/products/documentation/index.ssp

MFG/PRO Documentation

QXtend Outbound is designed to interoperate with MFG/PRO.

- For information on installing MFG/PRO or converting to a more recent release, refer to the appropriate installation or conversion guide for your system.
- For information on using MFG/PRO, refer to the *User Guides*.
- For information on QXtend Inbound, see *Technical Reference: QXtend Inbound*.

For QAD customers with a Web account, the complete MFG/PRO documentation is available for review or downloading at:

http://support.qad.com/

You can register for a QAD Web account by accessing the QAD Web site and clicking the Accounts link at the top of the screen. Your customer ID number is required. Access to certain areas is dependent on the type of agreement you have with QAD.

Features of the Web site include an online solution database to help you answer questions about setting up and using the product. Additionally, the QAD Web site has information about training classes and other services that can help you learn about QXtend inbound and Outbound and MFG/PRO.

Conventions

Menu References

This guide applies to MFG/PRO eB, eB2, and eB2.1. A number of menus have been reorganized between these releases. Differences in menu numbers are noted, when necessary, using the following convention:

User Tool Maintenance (36.20.4; 36.20.2 in eB)

The initial menu number identifies the program in the most recent release. The second menu number applies to the release specified and all earlier releases.

UNIX and Windows Conventions

This document supports the installation of QXtend Outbound on both UNIX and Windows platforms. The instructions use graphical screens and Windows file and path conventions. In the few places where the two sets of instructions diverge, the headings and text state explicitly which operating system is the focus of the current set of instructions.

Typographic Conventions

This document uses the conventions listed in the following table.

If you see:	It means:
`monospaced text`	A command, path, or file name.
`italicized monospaced text`	A variable name for a value you enter as part of an operating system command; for example, *`YourCDROMDir`*.
`<italicized monospaced text>`	A variable for a system value such as a drive letter or machine name; for example *`<hostmachine>`*.
`indented command line`	A long command that you enter as one line, although it appears in the text as two lines.
Note	Alerts the reader to exceptions or special conditions.
Important	Alerts the reader to critical information.
Warning	Used in situations where you can overwrite or corrupt data, unless you follow the instructions.

QAD Support Services

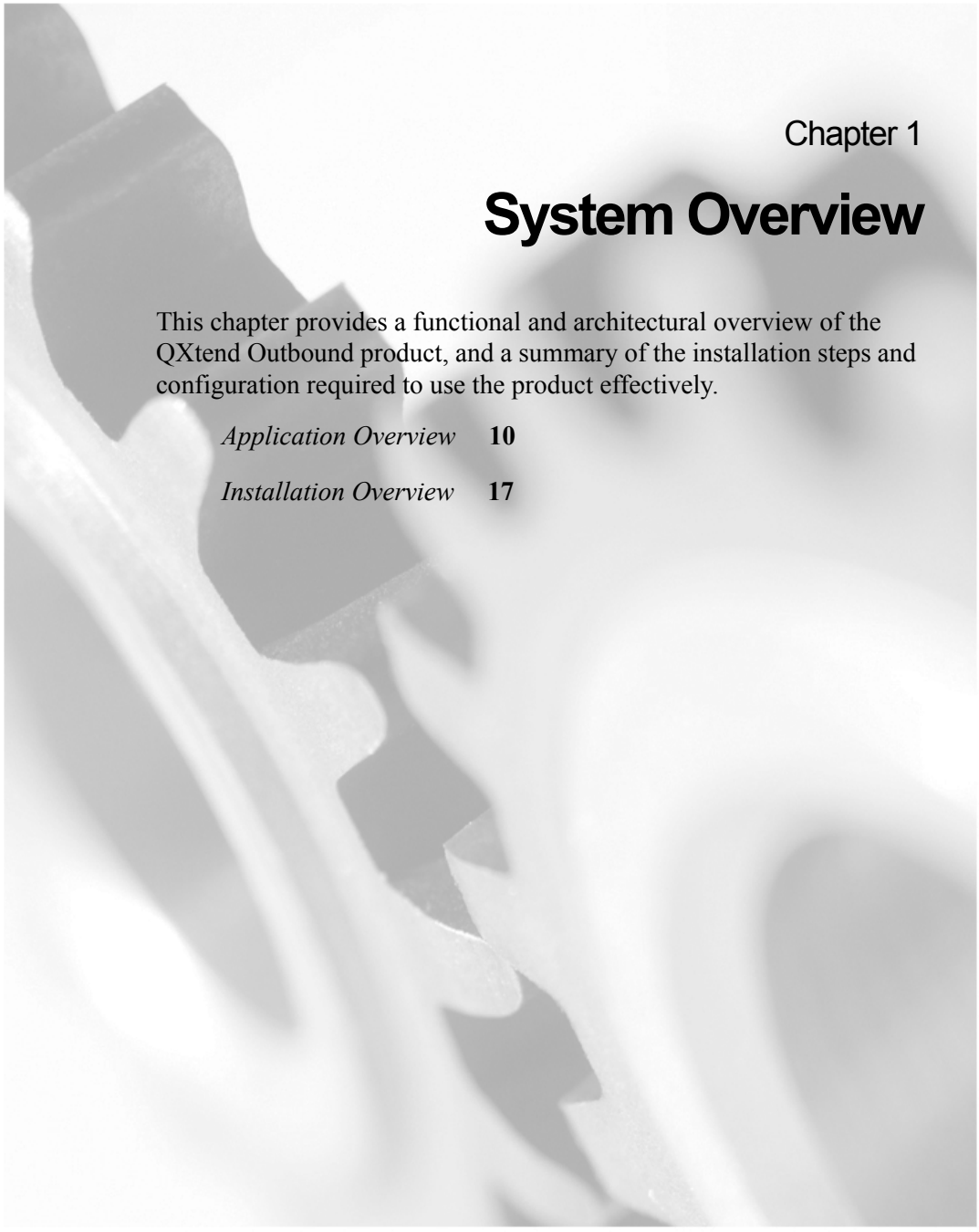
QXtend Outbound has a wide variety of configuration possibilities, is highly scalable, and can be customized easily. While this guide provides basic installation and configuration information, it cannot consider all of the possible computing environments and variations into which QXtend Outbound can be implemented.

To take full advantage of QXtend's flexibility and potential in your specific environment, contact your QAD Support representative for information on the installation and customization offerings supplied by QAD Support Services. These offerings include performance enhancements as well as technical and administration training.

Section 1

Overview and Installation

This section contains an overview of the QXtend Outbound product, and the requirements, preliminary setup, and installation instructions.



Chapter 1

System Overview

This chapter provides a functional and architectural overview of the QXtend Outbound product, and a summary of the installation steps and configuration required to use the product effectively.

Application Overview

QXtend Outbound (QXO) is a component of the QAD QXtend interoperability framework. This framework provides a standardized data interface between QAD products, and between QAD products and external systems. The framework consists of:

See “Glossary” on page 149 for definitions of terms and abbreviations.

- QXtend Inbound, which imports SOAP-compliant XML QDocs from external applications into MFG/PRO and other QAD applications such as QAD JIT Sequencing (JIT/S)
- QXtend Outbound, which exports user-configurable business objects as XML QDocs out of QAD products such as MFG/PRO and JIT/S to external subscribers

This interoperability framework replaces QAD products and solutions including CIM, Q/LinQ, and Data Synchronization. Future releases will incorporate EDI ECommerce.

The QXtend Outbound component exports business objects from MFG/PRO eB, eB2, or eB2.1, as well as from QAD JIT/S. This technical reference describes the standard installation and implementation of QXtend Outbound for MFG/PRO. Details about using, installing, and implementing QXtend Outbound with other QAD applications can be found in the guides for those applications.

Because QXtend Inbound and QXtend Outbound have different data requirements, inbound and outbound QDocs are not always equivalent. Inbound QDocs provide input to MFG/PRO in the same way a user would. The data is input field-by-field through the user interface, but the input is automated. Therefore, inbound QDocs require a rigid data sequence and are limited to a single entry program, such as Sales Order Maintenance.

Outbound QDocs include data from multiple MFG/PRO tables, and possibly from multiple screens, in order to provide external subscribers with a comprehensive business object.

Business Objects and Outbound QDocs

All outbound QDocs are defined by an underlying business object. A business object consists of data from a related set of tables required from a source QAD application. One such set is the sales order business object, which consists of sales order header and line data; ship-to, sold-to, and bill-to customer information; Intrastat data; tax details; and comments.

Each business object is defined in the QXtend Outbound administrative interface, which lets you select the tables and fields your target applications—your subscribers—need. However, all subscribers do not require the same elements from any given business object. To tailor business object output for multiple subscribers, you define profiles that filter and qualify business object data for one or more subscribers.

Figure 1.1 shows the basic data flow from MFG/PRO tables to the profiles that provide the template for outbound QDocs for specific subscribers.

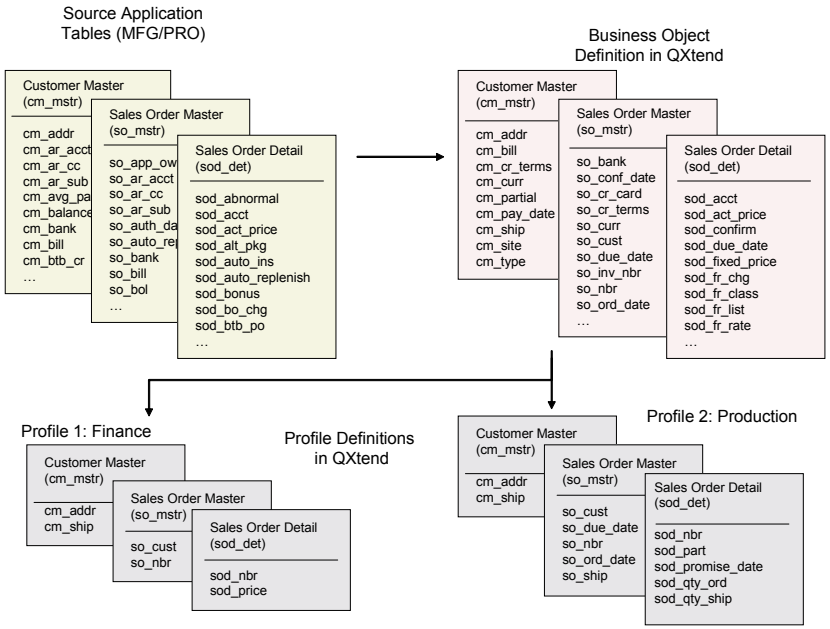


Fig. 1.1 Data Flow from MFG/PRO to Profiles

With the current release of QXtend Outbound, the following 19 business objects are shipped with the product.

Table 1.1 Standard QXtend Outbound Business Objects for MFG/PRO

Carrier	Item	Service Category
Customer	Item Site Cost	Service Support Call
Customer Item	Product Structure	Service Support End User
Customer Schedule	Sales Order	Service Support Engineer
Inventory Transaction	Sales Order Price List	Service Support Work Code
Invoice History	Scheduled Sales Order	Shipper
	Ship-To Customer	

These business objects reference data in the tables listed in Table 1.2. Each of these tables is defined in QXO as an event type. For these tables, schema triggers are loaded during installation and can be activated as needed. The tables vary slightly by MFG/PRO release. In addition to the tables listed in Table 1.2, in-line (programmatic) triggers are used on the following tables:

- abs_mstr, ASN/BOL/Shipper Master
- in_mstr, Inventory Master
- so_mstr, Sales Order Master

Table 1.2 Tables with Schema Triggers

Table Name	Description	eB	eB2	eB2.1
absc_det	Shipment Carrier Detail	✓	✓	✓
absd_det	Shipment Line Item Detail		✓	
absl_det	Shipment Detail Line Charges	✓	✓	
absr_det	Shipment Requirement Detail	✓	✓	✓
ad_mstr	Address Master	✓	✓	✓
ca_mstr	Service/Support Call Master	✓	✓	✓
ccd_mstr	Service/Support Call Fault Code Master	✓	✓	✓
cd_det	Master Comments	✓	✓	✓
cm_mstr	Customer Master	✓	✓	✓
cmt_det	Transaction Comments	✓	✓	✓
cp_mstr	Customer Item Master	✓	✓	✓
egt_mstr	Service/Support Engineer Tracking Master	✓	✓	✓

Table 1.2 — *Tables with Schema Triggers* — (Page 1 of 2)

Table Name	Description	eB	eB2	eB2.1
eng_mstr	Service/Support Engineer Master	✓	✓	✓
eu_mstr	Service/Support End User Master	✓	✓	✓
fsc_mstr	Service Category Master	✓	✓	✓
fwk_mstr	Service/Support Work Code Master	✓	✓	✓
idh_hist	Invoice History Detail	✓	✓	✓
ie_mstr	Import/Export Master	✓	✓	✓
ied_det	Import/Export Detail	✓	✓	✓
ih_hist	Invoice History Master	✓	✓	✓
isb_mstr	Service/Support Installed Base Item Master	✓	✓	✓
itm_det	Service/Support Call Item Detail	✓	✓	✓
lacd_det	Logistics Accounting Charge Detail	✓	✓	
ls_mstr	Address List Detail	✓	✓	✓
pi_mstr	Price List Master	✓	✓	✓
pid_det	Price List Detail	✓	✓	✓
ps_mstr	Product Structure Master	✓	✓	✓
pt_mstr	Item Master	✓	✓	✓
ptp_det	Item Planning Detail	✓	✓	✓
sch_mstr	Release Management Schedule Master	✓	✓	✓
schd_det	Release Management Schedule Detail	✓	✓	✓
sct_det	Cost Simulation Total Detail	✓	✓	✓
scx_ref	Scheduled Order Cross-Reference	✓	✓	✓
sob_det	Sales Order Configuration Bill Detail	✓	✓	✓
sod_det	Sales Order Detail	✓	✓	✓
sodlc_det	Sales Order Detail Line Charges	✓	✓	
spt_det	Cost Simulation Item Detail	✓	✓	✓
tr_hist	Inventory Transaction History	✓	✓	✓
tx2d_det	Tax Detail	✓	✓	✓
wo_mstr	Work Order Master	✓	✓	✓
wod_det	Work Order Detail	✓	✓	✓
wr_route	Work Order Routing	✓	✓	✓

Table 1.2 — *Tables with Schema Triggers* — (Page 2 of 2)

Sequencing Business Dependencies

Because QXtend Outbound supports multiple event services in a distributed environment, it is possible to load data messages in an incorrect sequence. For example, a sales order may be sent through before the new or revised customer record for that order.

See “Add Source Application Business Object Groups” on page 85.

To enforce correct sequencing of data messages, business objects can be assigned to a business object group. Business object groups ensure that designated groups of business objects are published in chronological sequence.

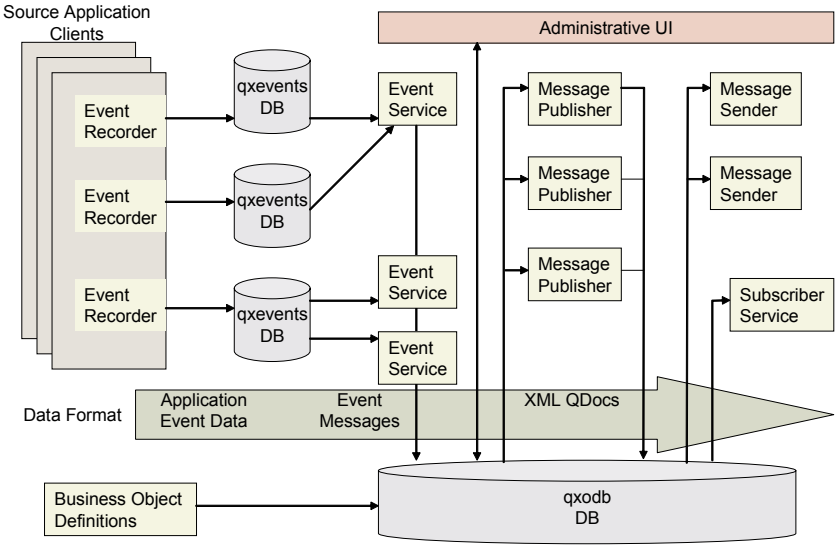
By placing the customer and sales order business objects in a business group, you can guarantee that if an event is generated for both these objects, the customer data will be extracted prior to the sales order data.

Business object groups are set up as part of the source application configuration. Each business object can belong to only one group.

The QXtend Outbound Process

Creating QDocs from extracted MFG/PRO data is accomplished in several steps. Each step is defined in, and can be monitored in, the QXtend Outbound administration interface.

Fig. 1.2 Basic QXtend Outbound Architecture



The steps are:

1. A change occurs to watched data in MFG/PRO.
2. QXtend Outbound queries MFG/PRO for the changed records.
3. The extracted data is published as QDocs.
4. The QDocs are delivered to the assigned subscribers.

Taking each step in turn, the product architecture, and the installation and implementation requirements, become clear. Refer to Figure 1.2 on page 14 to see product architecture and data flow through the product.

Change to Watched Data in MFG/PRO

Data becomes watched when you install and then activate replication triggers on the tables for a given business object. In order to avoid performance overhead, the triggers are simple: they write out a notification of a changed record to a side database, `qxevents`. A change is any add, modify, or delete to these tables.

The triggers required for supported QXtend Outbound business objects are shipped and installed with the product. However, even unused triggers, if active, create a small performance overhead. Therefore, each implementation determines which triggers are activated.

QXtend Outbound Queries MFG/PRO

QXtend Outbound uses an event service to connect to the `qxevents` database and poll for change notifications. When it encounters a change record, the event service queries the MFG/PRO tables that make up the impacted business objects in QXtend Outbound.

One table can be a component of multiple business objects. For example, the customer master, cm_mstr, is a component of the customer, customer item, and sales order business objects, among others. Depending on setup, queries can be for the full business object, or only for the primary index fields and any changed field data.

No query against MFG/PRO is run when:

- No valid or active business object exists for the changed data.
- No active profiles exist for the data.
- No active subscribers request this data.

The event service writes the extracted data to the QXtend Outbound database, `qxodb`.

Publishing the Data

Next, the message publisher generates one or more QDocs using the profiles as templates. It stores the QDocs in `qxodb`. The QDocs generated are XML documents in proprietary QAD format. The message publisher creates the QDocs, then wraps them in a SOAP-compliant container—an envelope, a header, and a footer.

The SOAP envelope allows the QDoc to be sent to Web services that accept or poll for XML documents.

Delivering the Data

The QDocs can be delivered to a directory location where the subscriber polls for and picks up new QDocs. Alternately, the QDocs can be delivered to a Web service listening for new QDocs through a URL (Universal Resource Locator).

The subscriber can be another MFG/PRO instance, another QAD product, or a third-party application or messaging broker set up to receive and interpret QDocs. QDocs moving between QAD products start as outbound QDocs and are picked up by QXtend Inbound.

Installation Overview

Installing QXtend Outbound typically occurs on two servers: an MFG/PRO server and a separate QXtend Outbound server. Table 1.3 provides a breakdown of prerequisites, services entries required, installation steps, and the steps to launch the interoperating application.

Table 1.3 QXtend Outbound Deployment Breakdown

MFG/PRO Server	QXtend Outbound Server	QXtend Outbound Client
Prerequisites		
Progress 9.1d+ MFG/PRO eB to eB2.1	Java 1.4.1+ Tomcat 1.4.31+ Progress OpenEdge 10B.01	Internet Explorer 6.0+
New Services Entries		
qxoevents database	qxodb database NameServer AppServer	
Installation Steps		
Install files to disk. Update services. Add qxevents to database set. Generate new scripts. Run MFG/UTIL workflow: • Load schema to qaddb. • Create qxevents.db. • Load qxevents schema. • Load data to all databases. • Start database servers. • Compile. Modify config.xml Enable QXtend in MFG/PRO	Install files to disk. Update services. Run `prepqxodb`. Compile. Configure AppServer.	
Starting the System		
Start database servers. Start qaddb client.	Start Tomcat. Start AppServer.	Start browser. Access QXtend Outbound.

Note Conversions from version 1.0 to 1.1 are not currently supported.

Implementing QXtend Outbound

1. Configure each MFG/PRO instance as a source application.
2. Activate all required event types for each source application.
3. Configure QXtend Outbound services: event services, message publishers, and message senders.
4. Set up subscribers.
5. Identify the required business objects.
6. Set up or modify the required business objects. For each business object you can:
   - Add or modify the table joins to use.
   - Add or modify the WHERE clause to select what data to retrieve.
   - Set fixed export values for specific data elements.
   - Write and assign calculations for specific data.
   - Add custom fields.
   - Determine if unchanged data is published.
7. Create profiles.
8. Activate the required triggers in MFG/PRO.

Customizing QXtend Outbound

You only need to customize QXtend Outbound for MFG/PRO if you want to modify existing business objects or add new ones. More extensive customization is needed if you want to add business objects for custom MFG/PRO applications or a Progress-based application other than MFG/PRO.

To modify MFG/PRO business objects or add new ones, you must:

1. Define the business object requirements.
2. Identify the MFG/PRO tables required and activate or add these event types.

3 Update `qaddb` schema and compile supplied trigger files for any tables not currently configured for QXtend Outbound.

4 Follow implementation steps 4 through 8 in the previous section.

For details, see “Creating a Standard MFG/PRO Business Object” on page 114.

To use QXtend Outbound with a custom MFG/PRO application or an application other than MFG/PRO, you must:

1 Define the business object requirement by determining what information is needed and identifying the application tables and fields that will supply the detail.

2 For all custom tables added to MFG/PRO or for non-MFG/PRO application tables, define the tables as event types.

3 Update the schema with replication write and delete triggers and create trigger files.

4 Follow implementation steps 4 through 8 in the previous section.

For details, see “Creating a Business Object for Custom Tables” on page 116.

Chapter 2

System Requirements

A QXtend Outbound implementation requires planning for adequate system resources on at least two servers: the QXtend Outbound server and each source application MFG/PRO production server where the MFG/PRO adapter is installed. This chapter provides guidelines you can use to generate estimates of the hardware and software requirements.

QXtend Outbound Server Requirements

The following sections list the prerequisite requirements for the QXtend Outbound server.

Important If you are installing both QXtend Inbound and QXtend Outbound on the same host, QXI must be installed first. The QXO server installation directory (*`QXOsrvInstallDir`*) PROPATH must be placed first in the QXI startup scripts.

Operating Systems

The QXtend Outbound server supports the following platforms:

- Linux
- HP-UX
- Sun Solaris (SPARC)
- Compaq UNIX (Tru64)
- IBM AIX
- Windows 2000 and above

Software

The following software must be present on the QXtend Outbound server at the time of the QXtend Outbound installation:

- A complete Progress OpenEdge 10 (minimum version 10.0B01) installation including the following components:
  - Enterprise RDBMS
  - OpenEdge Application SVR Enterprise
  - OpenEdge Studio
- Java Software Development Kit, J2SDK, version 1.4
- A functioning application Web server of either:
  - Tomcat, version 4.1.31+
  - WebSphere, version 5.1+

MFG/PRO Server Requirements

The following sections list QXO requirements for the MFG/PRO server.

Warning If you implement DataSync and QXtend Outbound together, extra events are generated and processed. There is a minimal, unavoidable performance impact.

Software

- A complete Progress version 9.1D+ installation
- MFG/PRO eB through eB2.1, or other external applications

  Any application interfacing with QXtend Outbound must use Progress databases with Progress V9 or greater or Oracle databases with a Progress V9 or greater dataserver.

For more information, see the MFG/PRO installation guide for your system.

Hardware

In addition to the resources already allocated on the MFG/PRO server, the adapter requires:

- 50 MB disk space

Client Requirements

QXtend Outbound client systems are browser based and require only:

- Internet Explorer, version 6.0+

Supporting Technologies

QXtend Outbound incorporates various Web-based technologies to support various features. These technologies are included with the product and are transparent to the user. They are listed here to give credit to the open-source projects that created them.

- Struts is an open source framework for building Web applications, part of the Jakarta Project, sponsored by the Apache Software Foundation.

  http://jakarta.apache.org/struts/index.html

- Apache AXIS is an implementation of the SOAP (Simple Object Access Protocol) submission to W3C.

  http://ws.apache.org/axis

- All messages and QDocs are logged using Log4j from Apache, a reliable, fast and flexible logging framework for Java.

  http://jakarta.apache.org/log4j

Chapter 3

Preparing the QXtend Outbound Environment

Use this chapter to set up your environment before installing QXtend Outbound.

Setting Environment Variables

In order for the QXtend Outbound installation scripts to run successfully, several environment variables need to be set correctly.

Set Windows Environment Variables

1. Open the System icon in the Control Panel.
2. On the Advanced tab, click Environment Variables.
3. Click New to add a new variable name and enter the variable value:

Variable Name	Variable Value
DLC	Progress OpenEdge 10 install directory
PATH	Click Edit to add `DLC\bin` to the beginning of the path, and append `;c:\tomcat\bin ;c:\j2sdk1.4.0\bin` to the end.
TOMCAT_HOME	`c:\tomcat`
JAVA_HOME	`c:\j2sdk1.4.0`
CATALINA_HOME	`c:\tomcat`

4. Click OK to exit the screen.

Set UNIX Environment Variables

In UNIX environments, add the following lines to the Tomcat user `.profile` to ensure the correct path with each log-in. The Tomcat user may be the `mfg` user or `root`:

```
export DLC=/pathtoProgress10
export PATH=$DLC/bin:oldPath:/tomcat/bin:/j2sdk1.4.0/bin
export TOMCAT_HOME=/pathtoTomcat
export JAVA_HOME=/pathtoJ2SDK
export CATALINA_HOME=/pathtoTomcat
```

Creating Services Entries

QXtend Outbound requires service entries for the two QXtend Outbound databases and for a QXtend Outbound AppServer. All service names, host names, and port numbers must match in the `services` files of each client and each database server on the network. The location of the UNIX `services` file on the server is typically the `/etc` directory. On Windows it is located in:

```
c:\winnt\system32\drivers\etc
```

Add the names of your database services to your `services` file. Add one for each database. Limit the service names to 16 characters. You can use any unused port numbers.

Add QXtend Outbound Server Services

Table 3.1 shows an example service name and port number for the QXtend Outbound server.

Table 3.1 QXtend Outbound Server Service File

Service Name	Port #/Protocol	Comment
qxodb-service	6600/tcp	# QXO Database Service

Add MFG/PRO Server Services

Table 3.2 shows example service names and port numbers for the MFG/PRO server. The production database server services may have been added during the MFG/PRO install.

Table 3.2 MFG/PRO Server Service File

Service Name	Port #/Protocol	Comment
qxevents-service	6700/tcp	# QXO Events Database
mfgprod-service	6710/tcp	# MFG/PRO qaddb Database
hlpprod-service	6720/tcp	# MFG/PRO help Database
admprod-service	6730/tcp	# MFG/PRO admin Database

Chapter 4

Installing QXtend Outbound

This chapter provides instructions for installing the QXtend Outbound server files and database and for building the QXtend Outbound events database and environment on your MFG/PRO application server.

Overview

The QXtend Outbound installation requires the following major tasks:

- Install and configure the QXtend Outbound server.
- Install and configure the QXtend Outbound MFG/PRO adapter.
- Install the QXtend `.war` file.
- Install MFG/PRO GUI client updates (optional).
- Test the installation.

The following instructions apply to Windows and UNIX environments.

Installing the QXtend Outbound Server

These steps transfer the QXO application from the delivery media to your server, and configure the application for operation with production MFG/PRO databases.

Prerequisites

This installation assumes that you have completed the following:

- Installed Java.
- Installed Tomcat.
- Set the JAVA_HOME and TOMCAT_HOME variables.
- Ensure that Tomcat is *not* running.
- Ensured that you have full permissions in the target installation directory.

Mount the CD-ROM (UNIX only)

1. Log on as `mfg`.
2. Mount the CD-ROM. Example commands are listed in Table 4.1.

Table 4.1 UNIX CD Drive Mount Commands

Hardware	Mount Command
Sun	`volcheck cdrom`
HP	`/etc/mount -F cdfs /dev/dsk/`*`YourCDDevice`*` /cdrom`

Hardware	Mount Command
Digital	`mount -r -o noversion -t cdfs /dev/`*`YourCDDevice`* `/cdrom`
AIX	`smitty mountfs` Then select file system, directory, and file system type (`cdrfs`).
Linux	`mount /dev/hdb /mnt/cdrom` Where `/hdb` could be `hdc` or `hdd` among other possibilities.
All others	Refer to your operating system documentation or vendor for requirements to mount a CD-ROM. You may be able to type `man mount` to determine the correct command.

Install the Outbound Server

1. In UNIX, log on as user `mfg` under the group `qad`. For Windows, log in as a user with Administrator privileges.
2. On the CD, change to the directory containing the database server media. This is the temporary tape directory for tape installs.
3. Change to the `install` directory:

   ```
   cd install
   ```

4. Launch the database server installation script in that directory:

   ```
   ./install.ksh
   ```

   For Windows, run `install.exe` from Run on the Start menu.

   A welcome screen displays. Press Enter.

   ```
   Welcome to QAD's QXtend Outbound 1.1 installation.
   We are installing QXtend Outbound 1.1 for Linux.
   Press <Enter> to view license agreement.
   ```

5. Accept the QXtend Outbound software license agreement. Press Ctrl+C to jump to the end of the agreement.

   ```
   Do you accept all the terms of the preceding License
   Agreement?
   If you choose no, the install will stop.

   To install QXtend Outbound 1.1, you must accept this
   agreement. (y/n)?
   ```

6 Accept the default or enter the installation log file location. Other QAD installs (MFG/PRO, JIT/S, service packs, and others) look in this location on this machine for installation information. If you enter a different log file location, make note of it for later installations.

```
Please enter the location where the log file should be
written.
Default is /users/mfg/instlog
->/users/mfg/instlog
```

For Windows the default is `c:\instlog`.

If this is your first installation, you are prompted:

```
Unable to locate instqxo.ini
```

If this is your initial QXtend Outbound 1.1 install, accept Yes here to create this file. If this is an existing install, consider answering No here and locating the file. It contains useful previous installation information.

7 An important note displays a warning that you must be on a supported MFG/PRO version and service pack level. In addition, if your MFG/PRO version is displayed in the install options, but your service pack level is not, you will not need to compile for MFG/PRO.

Selecting Installation Components

In the next set of steps, you are prompted for which components of QXtend Outbound you are installing. The set of components shipped on the product CD are:

- QXtend Outbound server
- Updates for MFG/PRO
- Web server `.war` file
- Updates to the MFG/PRO GUI client

You can answer Yes to one or all of these prompts depending on what components you are installing to the current host. This document covers each install separately in order to keep install and configuration steps together. In general, you should install and implement the components in the order presented in this guide.

1. You are prompted if you want to install QXtend Outbound Server from this machine. Enter Yes.
2. Enter a new, or accept the default, install directory. By default this is `/home/qxo` in Unix environments and `c:\qxo` under Windows. In this document, this is the *`QXOsrvInstallDir`*.
3. You are now prompted whether you want to install the following components:
   - Updates for MFG/PRO
   - Web server `.war` file
   - Updates to the MFG/PRO GUI client

   Respond Yes or No to each depending on whether the component is to be installed on the current host.
4. A summary of your install choices displays. Confirm your response and the components are installed to your chosen destinations.
5. The installation log, `qxo.log`, is written to the log directory you specified.

Prepare the Progress Environment

This section is divided into Windows and UNIX steps. In this set of steps you run a script that creates two Progress databases: `qxodb.db` and `qxevents.db`. You then configure your Progress session scripts.

Before you begin, make sure your:

- DLC variable is set to Progress OE10B.
- Path variable contains `DLC\bin`.

If you plan to use a naming convention for your databases other than the defaults, you can rename them after the installation completes. Regardless of the database names you use, you must continue to use `qxodb` and `qxevents` as the logical database names.

Progress Preparation for Windows Installs

1. If you do not have a Progress developer's license, you must modify one of the installation scripts. In a text editor, open *QXOsvrInstallDir*\runinstallp.bat. Add -rx following the pro command in the script:

   ```
   e:\oe10b\bin\pro -rx -p install.p -param qxodb
   ```

   Save and exit the file.

2. Navigate to *QXOsrvInstallDir* in the Windows Explorer and double-click prepqxodb.bat. This file creates the two databases in *QXOsrvInstallDir*\db. The qxodb database is the main QXtend Outbound database. The qxevents database is required on this server for compiles only.

3. If you used different service names than recommended in "Creating Services Entries" on page 27, open start-sess.pf in a text editor and modify each instance of qxodb-service to match the correct service name.

   Ensure the -H parameter specifies the name of the machine where qxodb is located.

4. Copy the following:

   ProgressInstallDir\bin\progress.ini

   To:

   QXOsrvInstallDir\wrk

   Rename it to:

   qxo-service.ini

5. Open qxo-service.ini in a text editor and modify the PROPATH in the [Startup] and SysCheckmark sections to include, in the following order:

   QXOsrvInstallDir\wrk\progress\qxtend\src

   QXOsrvInstallDir\wrk\progress\qxtend\src\proxy

   QXOsrvInstallDir\wrk\progress\qxtend\src\qxo

Progress Preparation for UNIX Installs

To run the following script, ensure that you have execute permission. If not, execute the command:

```
chmod 777 prepqxodb.sh
```

1 If you do not have a Progress developer's license, you must modify one of the installation scripts. In a text editor, open *QXOsvrInstallDir*/prepqxodb.sh. Add -rx following the pro command in the script:

```
$DLC/bin/pro -rx -p install.p -param qxodb
```

Save and exit the file.

2 Run *QXOsrvInstallDir*/prepqxodb.sh. This script creates both databases and compiles the code on the QXtend Outbound server. The script prompts for confirmation of:

   a The QXO work directory

   b The $DLC to Progress OpenEdge 10B

   c The $PATH variable set for the current session

3 Open start-sess.sh from *QXOsrvInstallDir*/wrk in a text editor and make the following changes:

   - Replace all instances of <OpenEdge installdir> with the Progress OpenEdge installation location.
   - Replace the export PATH= statement with your current path. Also ensure that $DLC/bin is included in the PATH statement.
   - Modify the export INSDIR= statement to point to *QXOsrvInstallDir*.
   - If you used different service names than those recommended in “Creating Services Entries” on page 27, modify each instance of qxodb-service to match the correct service name. Ensure the -H parameter specifies the name of the machine where qxodb is located.

4 Set permissions for the modified file by entering the command:

```
chmod 777 start-sess.sh
```

Modify catalina.bat or catalina.sh

In order to display graphs in QXtend, properties must be set in *TOMCAT_HOME*/bin/catalina.bat for Windows and *TOMCAT_HOME*\bin\catalina.sh for Unix. Displaying graphs requires that the Tomcat headless option be set to true:

```
JAVA_OPTS="-Djava.awt.headless=true"
```

Insert this line at the start of the file where other options are set.

Compile Outbound Code (Windows)

This section describes the required compile steps using the Progress compiler. Complete these steps for Windows installs. (The UNIX compile is completed within the database creation script.)

1. In the Windows Explorer, navigate to *QXOsrvInstallDir*.
2. Double-click compile.bat to launch the compile.
3. Once the compile is complete, open compile.log in *QXOsvrInstallDir* and check for errors.

Configure AppServer on Windows

A new Progress AppServer, QXOSession, is required for QXtend Outbound. The following steps are for Windows systems. See “Configure AppServer on UNIX” on page 41 for the steps on UNIX systems.

Adding QXtend Server to Progress Explorer

1. Launch the Progress Explorer Tool from the Start menu.
2. Choose Add Progress Server from the Action menu.
3. Add the QXtend Outbound server, typically localhost, using a valid user name and password as shown.

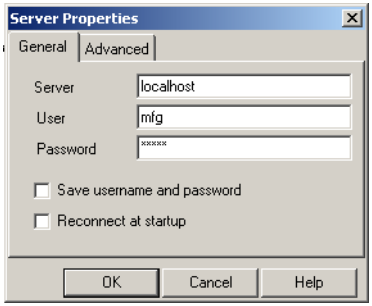


Fig. 4.1
Server Log-In in Progress Explorer

4 Right-click the localhost node and select Connect to connect to the server.

Starting the AdminServer

Prior to creating the new AppServer, start the AdminServer using the following steps.

1 Open Administration Tools|Services from the Windows Control Panel.

2 Choose AdminService for Progress OpenEdge.

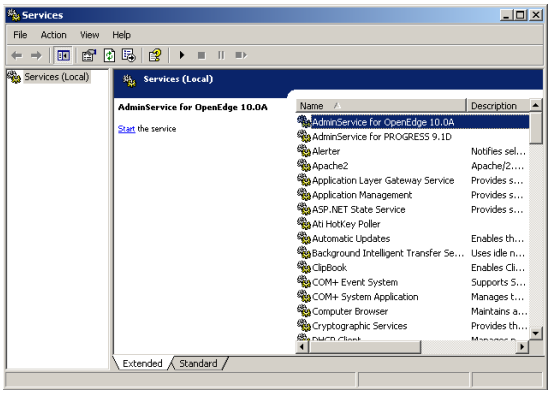


Fig. 4.2
Starting the Progress OpenEdge AdminService in the Windows Services Program

3 Select Start to start the service.

Creating the AppServer

1. Launch the Progress Explorer Tool from the Start menu.
2. Create a new AppServer named QXOSession.
3. Edit the properties for the AppServer as follows.
   a. In Broker Properties, open the General tab and set:

Operating Mode	Stateless
Port Number	Any available port
Working Directory	*QXOsrvInstallDir*\wrk

   b. Under the Logging Settings tab, set:

Log Filename	*QXOsrvInstallDir*\wrk\logs\QXOS.broker.log

   c. In the Server Properties node, open the General tab and set:

Startup Parameters	-db qxodb -H *hostname* -S *service-name*
Propath	*QXOsrvInstallDir*\wrk\progress\qxtend\src\proxy; *QXOsrvInstallDir*\wrk\progress\qxtend\src\; *QXOsrvInstallDir*\wrk

   d. Under the Logging Settings tab, set:

Log Filename	*QXOsrvInstallDir*\wrk\logs\QXOS.server.log

4. Add the qxodb database to the Explorer.
   a. Create new database.
   b. Enter the path and file name, *QXOsrvInstallDir*\db\qxodb.
   c. Edit the database defaultConfiguration adding the server arguments -H localhost -S qxodb-service using values specified in the services file updated in “Creating Services Entries” on page 27.
   d. Increase the # Lock table entries to 200,000.
5. Accept all other defaults and click OK.

Managing Lock Table Sizes

Setting up the `qxodb` database requires setting the size of the lock table for the database server. The default value recommended is 200,000. This is based on use of the QAD-defined business objects.

The following factors affect how this setting should be adjusted:

- The number of tables in a business object
- The number of fields in each business object table
- The number of records extracted for a table in a single business object message
- The number of event service processes running in parallel

An increase in any of these results in an increase in the number of locks required. Should the number of locks required exceed the number available, the Progress session trying to get those locks is automatically restarted (STOP condition) and an error is logged in the session log file and the database log file.

If this happens, do the following:

1. Shut down the QXtend service processes.
2. Shut down the `qxodb` database server.
3. Double the value of the `-L` parameter in the database startup script.
4. Restart the database server.
5. Restart the QXtend service processes.

The messages being processed when the process went down are backed out. They are picked up and processed correctly when the processes come back up.

You may need to carry out these steps several times to find a value that works with your business objects.

Unable to Invoke Proxy Call

In the event of “Unable to invoke proxy call” error messages during reconnection, increase the maximum servers in the AppServer pool. The default is 1; try a setting of 5.

On Windows

1 Open the Progress Explorer.

2 Connect to the AppServer QXOSession host machine.

3 View the AppServer properties.

4 Navigate to Agent|Pool Range.

5 Set Maximum servers to 5.

On UNIX

1 Open *ProgressInstallDir*`\properties\ubroker.properties` in a text editor.

2 Modify the QXOSession entry by adding:

```
minSrvrInstance=1
maxSrvrInstance=5
```

Starting the Database Server and AppServer

1 In the Progress Explorer, open the node for your host machine and select Databases.

2 Select `qxodb` under Databases and click the Start icon, a green arrow in the toolbar, as shown. The database server starts.

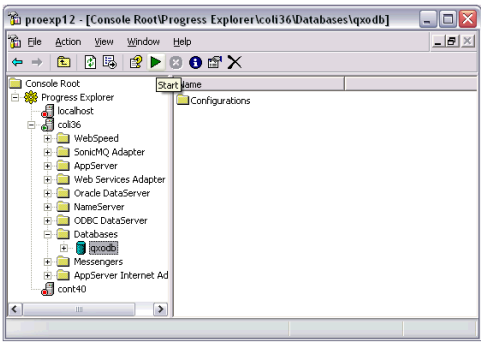


Fig. 4.3
Starting the Database Server

3 Select the AppServer node for the host.

4 Select the QXOSession AppServer and click the Start icon to start the AppServer.

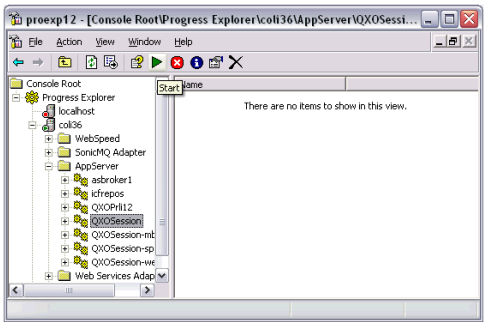


Fig. 4.4
Starting the Progress AppServer

Configure AppServer on UNIX

Create a new AppServer by adding an entry in `ubroker.properties` for the QXtend Outbound AppServer QXOSession.

1 Run the Progress `genuuid` utility to generate a unique identifier value for your AppServer. Save the value to assign to the uuid field later in these steps. In a terminal session run the following command:

```
$DLC/bin/genuuid
```

2 Open the Progress `ubroker.properties`, located in *`ProgressInstallDir`*`/properties`, in a text editor and add the following entry in the AppServer section:

```
[UBroker.AS.QXOSession]
    appserviceNameList=QXOSession
    brkrLogAppend=0
    brkrLoggingLevel=1
    brokerLogFile=<installdir>/wrk/logs/QXOSession.broker.log
    controllingNameServer=NS1
    defaultService=1
    operatingMode=Stateless
    portNumber=<Available Port Number>
    PROPATH=.,QXOsrvInstallDir/wrk/progress/qxtend/src/
    proxy,QXOsrvInstallDir/wrk/progress/qxtend/src/,
    QXOsrvInstallDir/wrk
    srvrLogAppend=0
    srvrLogFile=<installdir>/wrk/logs/QXOSession.server.log
    srvrLoggingLevel=1
    srvrStartupParam= -db qxodb -H <hostname> -S <service-name>
    uuid=<new-UUID>
    workDir=<installdir>/wrk
```

3 Insert the uuid generated in step 1 as the <new-UUID> value.

4 Replace all instances of <hostname> with your database host name and instances of <service-name> with the value entered in the services file for your QXO AppServer in “Creating Services Entries” on page 27.

5 Replace the <installdir> and <Available Port Number> variables.

Starting the qxodb Database Server

You can now start the database server for qxodb.

1 In a text editor, create start and stop scripts for the database server. Use the sample scripts that follow as models.

startQXODB.sh

```
export DLC=/dr01/progress/dlc10a

export PATH=/dr01/progress/dlc10a/bin:/bin/posix:/bin:/usr/bin:/
usr/local/bin:/usr/contrib/bin:/usr/bin/X11:/qad/local/bin:/
qad/local/scripts:/qad/continuus/ccm45/bin:/dr01/progress/
dlc10a_edoc/psviewer

proserve qxodb -S <service-name> -L 200000
```

stopQXODB.sh

```
export DLC=/dr01/progress/dlc10a

export PATH=/dr01/progress/dlc10a/bin:/bin/posix:/bin:/usr/bin:/
usr/local/bin:/usr/contrib/bin:/usr/bin/X11:/qad/local/bin:/
qad/local/scripts:/qad/continuus/ccm45/bin:/dr01/progress/
dlc10a_edoc/psviewer

proshut -by qxodb
```

2 Run startQXODB.sh to launch the database server.

Starting the AdminService and AppServer

Start the AdminServer using the following steps.

1 Ensure the $DLC environment variable is set to the *ProgressInstallDir*.

2 Verify that the $PATH variable contains $DLC\bin.

3 At a command prompt enter the command:

```
proadsv -start
```

Check the status of the AdminService by entering:

```
proadsv -q
```

4 Start the newly configured AppServer by entering:

```
asbman -name QXOSession -start
```

Query the AppServer to ensure it has started correctly:

```
asbman -name QXOSession -q
```

This completes the installation of the QXtend Outbound server.

Installing the MFG/PRO Outbound Adapter

The following steps transfer the application files to the MFG/PRO server, create the new `qxevents` database, add the database to your production database set and startup scripts, and compile the source code.

1 Mount the QXtend Outbound CD-ROM as described in “Mount the CD-ROM (UNIX only)” on page 30.

2 Repeat the steps under “Install the Outbound Server” on page 31, answering Yes when prompted to install the MFG/PRO updates for QXtend Outbound to this machine.

3 You are prompted for the location of your MFG/PRO installation. Do not enter the full path to the `/db` directory. Enter the *`MFGPROInstallDir`* and press Enter.

4 You are then prompted to identify the MFG/PRO version (eB, eB2, or eB2.1). Enter the version and press Enter.

5 Identify the MFG/PRO service pack level. If you are uncertain, the service pack level is shown in your MFG/PRO log-in screen or is available in the `version.mfg` file in *`MFGPROInstallDir`*. Select the service pack level and press Enter.

6 A summary of your install choices displays. Confirm your response and the MFG/PRO updates for QXtend Outbound are installed to your chosen destination.

Configure MFG/PRO for QXtend Outbound

Before launching MFG/UTIL, two steps are required:

1 Shut down your MFG/PRO database servers by running:

`stop.`*`dbsetName`*

For example, `stop.Production`.

2 Open *`MFGPROInstallDir`*`\wk0700.ini` in a text editor and update all instances of `DBName=.\db\mfgprod.db` with the correct database name and path for your environment as needed.

Add qxevents to a Database Set

Using the MFG/UTIL menus, you add `qxevents` to your MFG/PRO production database sets.

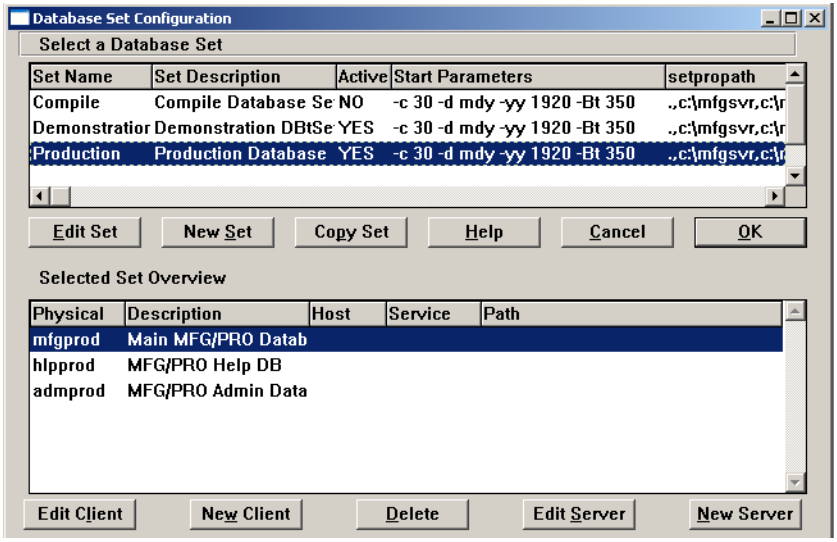
1 Launch MFG/UTIL from the *`MFGPROInstallDir`* using the following command:

`./mfgutil`

For Windows, launch MFG/UTIL from the icon on the Start menu.

2 Choose Database Set Maintenance from the Configure menu. The database set editor displays.

Fig. 4.5
Database Set Maintenance



3 In the upper frame, select the Production database set and choose Copy Set.

4 You are prompted for a new name. This is up to you. The name is used in startup scripts. In this guide, the set name used is QXOProd. Choose OK.

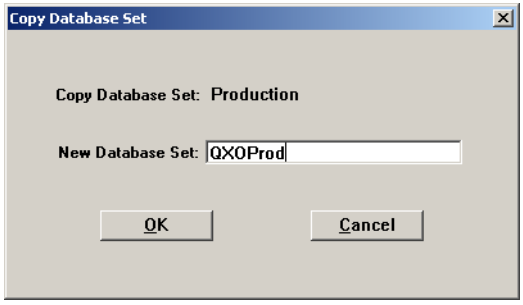


Fig. 4.6
Copy Database Set

5 In the Database Set Maintenance screen, select the new database set and choose Edit Set. In the Database Set Parameters screen, update the Set Description and place the following entry before the existing PROPATH:

- *QXOMFGsrvInstallDir*\qxo\

The new PROPATH would look like this:

QXOMFGsrvInstallDir\qxo,*Old_Propath*

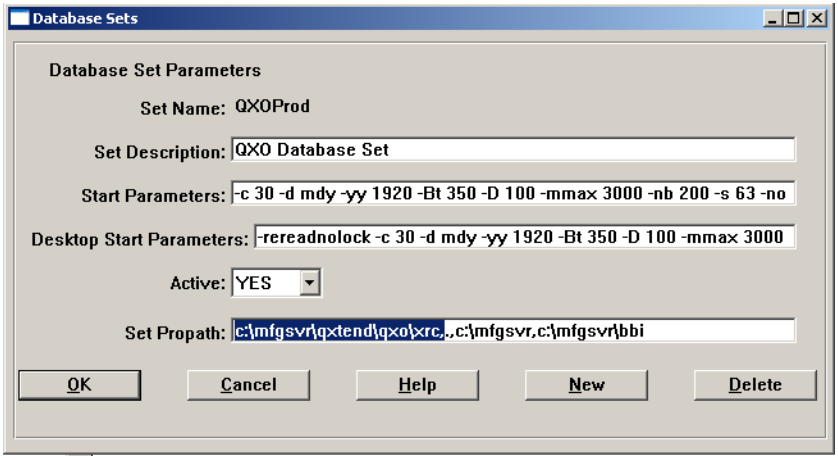


Fig. 4.7
Editing the QXO Set Parameters

6 Choose OK.

7 In the lower frame of Database Set Maintenance, select the MFG/PRO production database, `mfgprod`, and choose Edit Client.

See page 27.

8 In the Client Database Parameters screen, set Connection Type to ClientServer, and enter valid host and service names. Choose OK to save the changes.

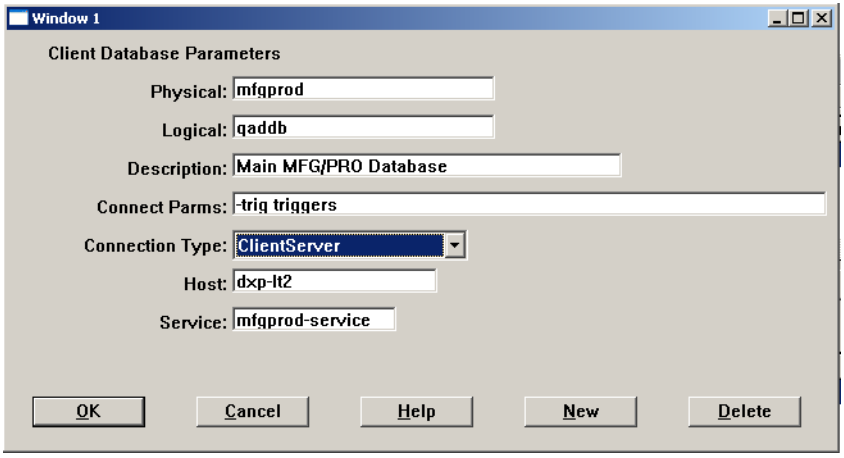


Fig. 4.8 Editing Client Parameters for mfgprod

9 Repeat these changes for the other two production databases, `admprod` and `hlpprod`.

10 In the lower frame, choose New Client to add the `qxevents` database to the set.

11 In the Client Database Parameters screen, set the values as shown in the figure. Enter the logical database name as `alias_qxevents`. Set Connection Type to ClientServer, and enter valid host and service names. Choose OK to save the changes.

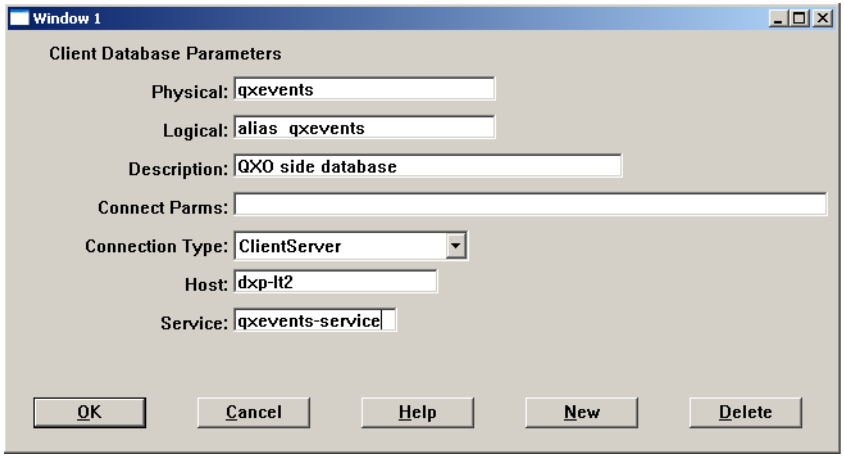


Fig. 4.9
Entering the qxevents Client Parameters

12 In the lower frame of Database Set Maintenance, select the `qxevents` database and choose Edit Server.

13 In the Server Database Parameters screen, set the path to the `qxevents` database as shown in the figure. Choose OK.

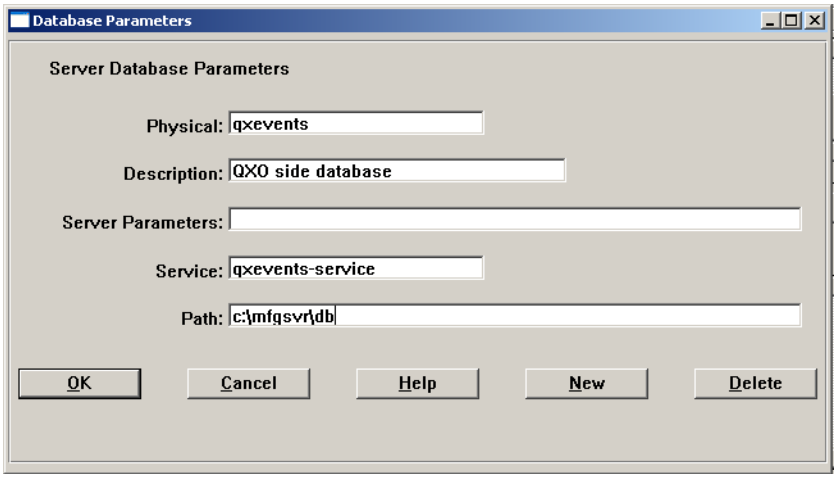


Fig. 4.10
Entering qxevents Server Parameters

14 The completed database set should look like the one in Figure 4.11.

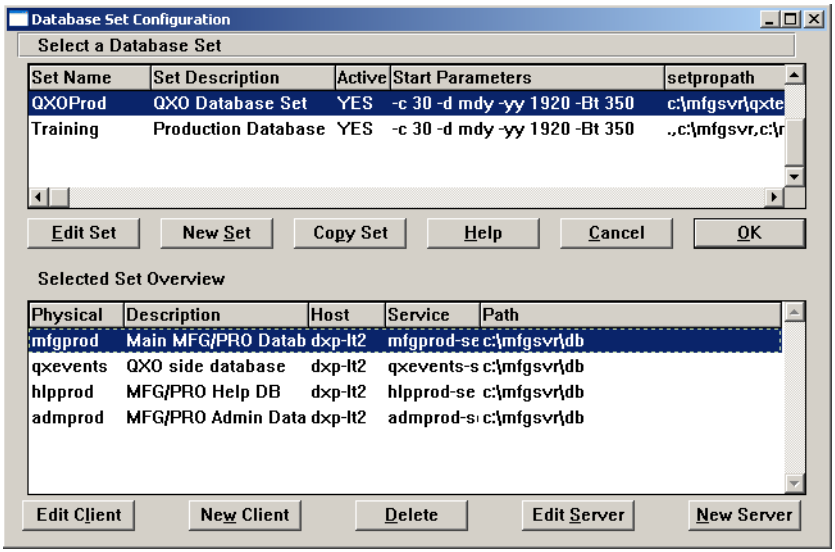


Fig. 4.11
Completed QXtend Outbound Database Set

15 Tab to the upper frame of Database Set Maintenance and choose OK to exit the program.

Generate Scripts

You can now generate server and client start and stop scripts that manage all the databases in the new database set at once.

1 In MFG/UTIL, choose Generate Scripts from the Scripts menu.

2 In the database sets menu, choose QXOProd and choose OK.

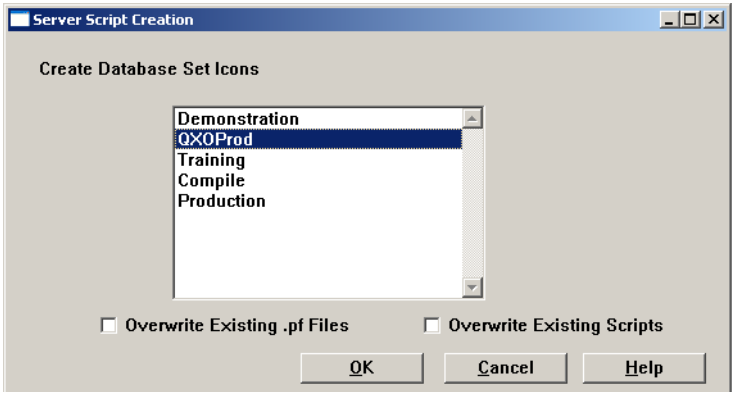


Fig. 4.12
Database Set Selection Menu

3 For Windows systems, you are prompted to create database set icons. These are program launch icons in the Start menu. Choose Yes if you want these available.

4 If you chose Yes in step 3, you are prompted for a folder to place the icons in. Choose OK to generate the scripts.

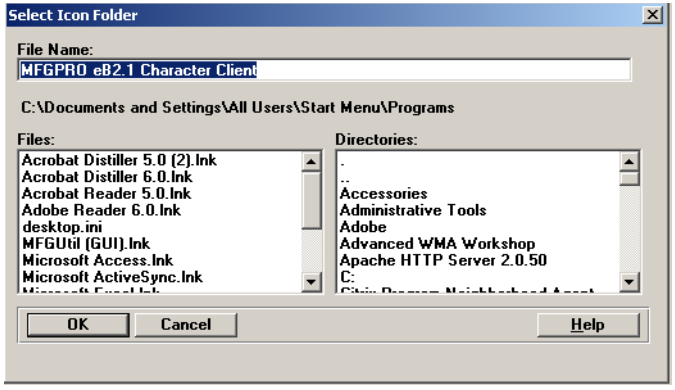


Fig. 4.13
Scripts Folder

5 The Log Window displays. Choose Close to continue. Several new scripts are created in *MFGPROInstallDir*.

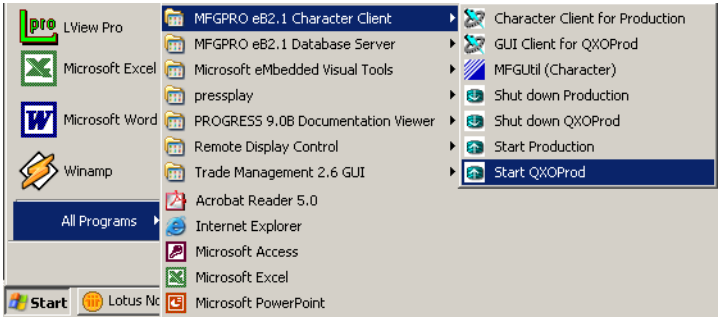


Fig. 4.14 Start and Stop Scripts in the Windows Start Menu

6 The Generate Scripts program does not update the `progress.chr` file; this must be done manually:

a Open *MFGPROInstallDir*`\progress.chr` in a text editor.

b Locate the first instance of the Propath under [Startup] and add the following entries:

```
QXOMFGsrvInstallDir\qxo\
QXOMFGsrvInstallDir\qxo\xrc
```

c Do the same under the [WinChar Startup] section.

d Save your changes and exit the text editor.

Create the qxevents Database

The `qxevents` database is created using an MFG/UTIL Guided Setup. The Guided Setup uses a text file, `wk0700.ini`, to partially automate the steps required to update your MFG/PRO environment. The workflow creates the `qxevents` database and loads the necessary schema and data files. This workflow also updates your production database with the QXtend Outbound triggers.

1 In MFG/UTIL, choose MFG/PRO Guided Setup from the Configure menu.

2 Select QXtend Outbound Installation Workflow in the Operation Set drop-down list box.

Review the following figures to become familiar with Guided Setup.

Note The number of operations is determined by the number of sections in `wk0700.ini`.

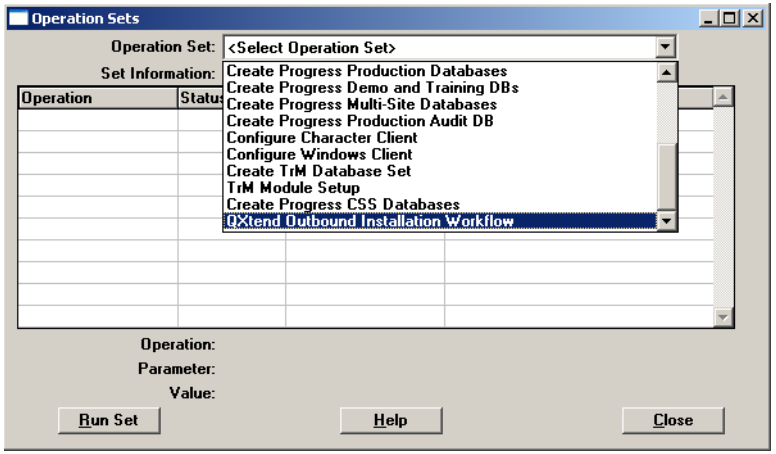
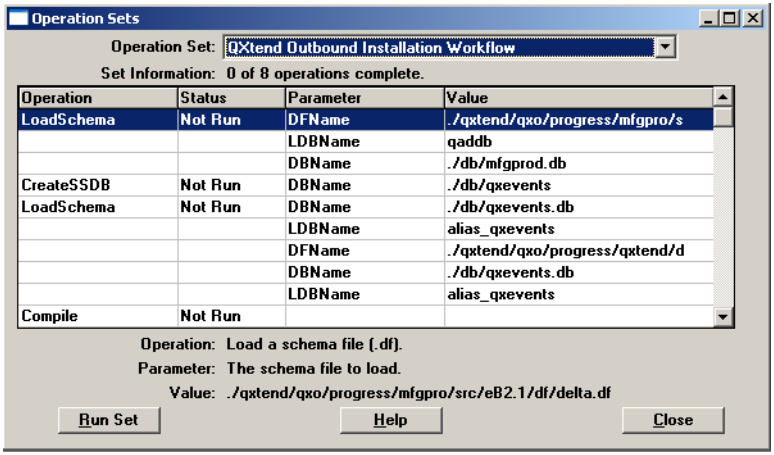


Fig. 4.15 Selecting the QXtend Outbound Workflow

Operation sets are groups of installation activities. The operations in a set display in the Operation frame. On completion, the status changes to Done. If errors occur or if you cancel processing prior to

completing a step, the status is Error. Below the Operation frame, the operation, the key variable required, and default value for that variable display.

Fig. 4.16
QXtend Outbound Workflow Tasks



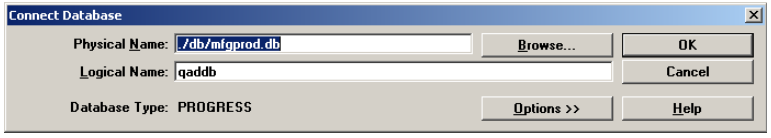
If you stop the workflow and an Error status is written to a step, this is the first step run when you restart the operation set.

You can exit the workflow and reset all the steps by choosing Reset MFG/PRO Guided Setup Files from the Configure menu.

3 Choose Run Set and press Enter.

4 The Connect Database screen displays with the MFG/PRO production database you identified in the preliminary steps. Choose OK to connect.

Fig. 4.17
Connecting to mfgprod



5 The Load Data Definitions screen displays with the correct path and data definition file name shown. Choose OK to load the schema into MFG/PRO.

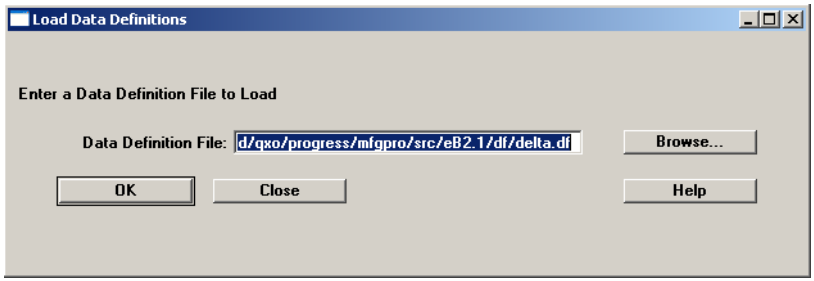


Fig. 4.18 Loading Schema into mfgprod

6 The QAD Log Window shows progress. On completion, choose Close to disconnect from the production database and continue the workflow.

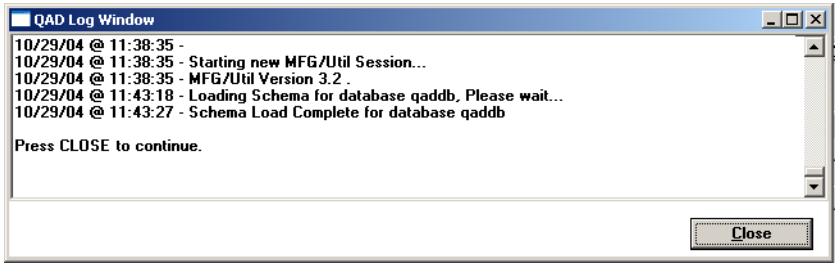


Fig. 4.19 QAD Log Window After Schema Load

7 The Create Database screen displays for `./db/qxevents`. Accept the defaults and choose OK to create the database.

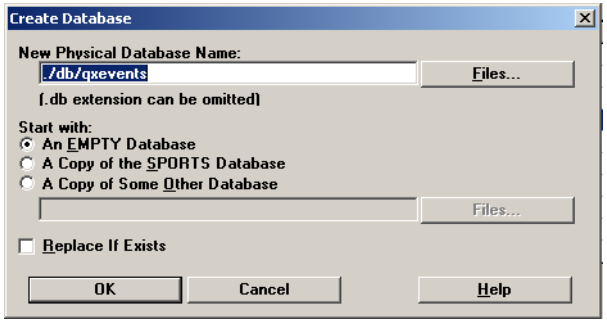
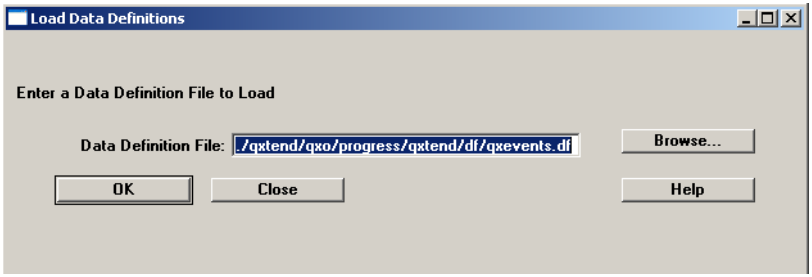


Fig. 4.20 Create Database for qxevents

8 The Connect Database screen displays for the new `qxevents` database. Enter a logical name of `qxevents`. Choose OK to connect.

9 The Load Data Definitions screen displays with the correct path and data definition file name shown for `qxevents.df`. Choose OK to load the schema into `qxevents`.

Fig. 4.21 Connecting to qxevents



10 The QAD Log Window shows progress. On completion, choose Close to continue the workflow.

Load Data

Depending on your MFG/PRO version and service pack level you will complete either one load or three. You must always load into `qxevents`; `qaddb` and `qadadm` are service-pack dependent.

1 The Connect Database screen displays for the `qxevents` database. Enter a logical name of `qxevents`. Choose OK to connect.

2 The Load Data screen displays with the correct path to the administrative data required in `qxevents`. Choose OK to load.

3 The Table Selection for Load screen displays with a single table, `erm_event_type`. The table is selected automatically. Choose OK.

4 Repeat steps 1 through 3 for the databases and directories listed in Table 4.2.

Table 4.2 Additional Data Loads

Database	Logical Name	Data Directory
mfgprod	qaddb	*MFGPROInstallDir*/qxtend/qxo/data/qaddb
admprod	qadadm	*MFGPROInstallDir*/qxtend/qxo/data/qadadm

5 The Compiler Options screen displays.

Important Without quitting MFG/UTIL or the workflow, complete the next step prior to running the compiles.

6 In a separate command window, run `start.QXOProd` to start the database servers before continuing with the next set of steps.

Compile Adapter Code

You now use MFG/UTIL to compile the QXtend code that resides on the MFG/PRO server. There are two separate compiles required. Use the following description and table to ensure you run each compile correctly.

1 Select Compile Procedures from the Programs menu.

2 Refer to Figure 4.22 and Table 4.3 to enter values into the Compiler Options screen.

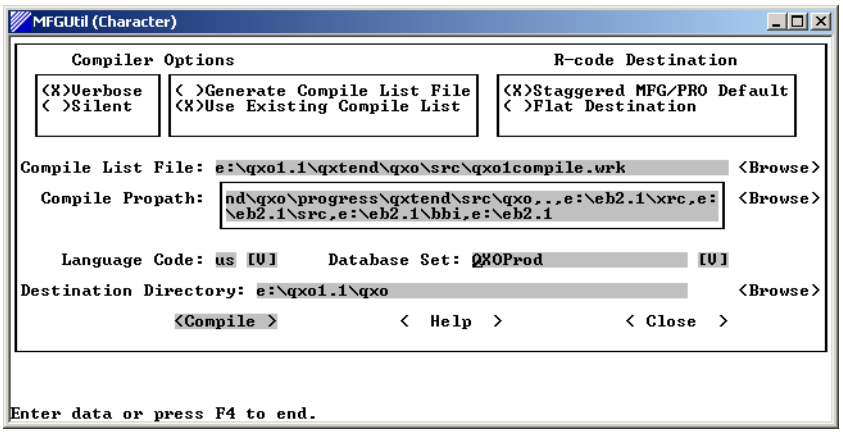


Fig. 4.22 Compiler Options Screen With First Compile Set Up

Table 4.3
Compiler Options Settings for QXtend Outbound

Field	Value
Compile 1	
R-code Destination	Staggered MFG/PRO Default
Compile List File	*QXOMFGsrvInstallDir*\qxtend\qxo\src\qxo1compile.wrk
Compile Propath (eB)	*QXOMFGsrvInstallDir*\qxo\src, *QXOMFGsrvInstallDir*\qxo\qxtend\src, ., *MFGPROInstallDir*\us\xrc, *MFGPROInstallDir*
Compile Propath (eB2 and eB2.1)	*QXOMFGsrvInstallDir*\qxo\src, *QXOMFGsrvInstallDir*\qxo\qxtend\src, ., *MFGPROInstallDir*/xrc, *MFGPROInstallDir*
Database Set	QXOProd
Destination Directory	*QXOMFGsrvInstallDir*\qxo
Compile 2	
R-code Destination	Flat Destination
Compile List File	*QXOMFGsrvInstallDir*\qxo\src\qxo2compile.wrk
Compile Propath (eB)	*QXOMFGsrvInstallDir*\qxo\src, *QXOMFGsrvInstallDir*\qxo\progress\qxtend\src, ., *MFGPROInstallDir*\us\xrc, *MFGPROInstallDir*
Compile Propath (eB2 and eB2.1)	*QXOMFGsrvInstallDir*\qxo\src, *QXOMFGsrvInstallDir*\qxo\progress\qxtend\src, ., *MFGPROInstallDir*\xrc, *MFGPROInstallDir*
Database Set	QXOProd
Destination Directory	*QXOMFGsrvInstallDir*\qxo

3 After verifying the settings, choose Compile. If the target directories do not exist, you are prompted to confirm their creation. A confirmation screen then displays.

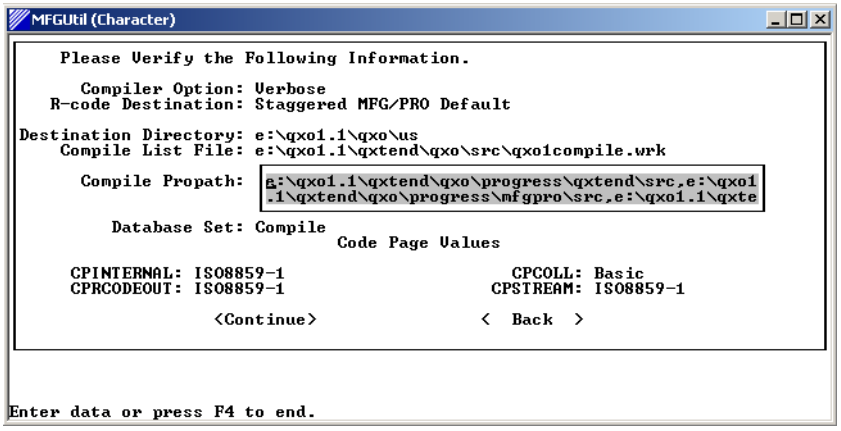


Fig. 4.23
Compile Summary

4 Validate the settings and choose Continue. The log window displays the compile progress. When the compile is complete, choose Close in the log window to continue.

Important Due to an internal error that could not be corrected prior to release, an additional step may be required after each compile. After closing the log window for each compile, even when no errors have occurred in the compile, an error message displays:

```
Error status from MFGPROInstallDir\xmfgusrc\compui.w.
Quitting operation set.
```

(Whether you get this error or not will depend on your version of MFG/UTIL.) In order to continue the operation set, you must exit the operation set workflow, open `wk0700.ini` in a text editor, and change the Status for the compile from `Error` to `Done`. You can then restart the operation set and it will automatically jump to the next step.

5 These steps repeat for the second compile.

6 On completion of the second compile, the Operation Sets screen displays again. Choose Close to exit.

7 Choose Exit from the File menu to close MFG/UTIL.

Define Configuration Settings

During installation on the server, a configuration file for the source application is copied to:

QXOMFGsrvInstallDir\qxo\config\config.xml

You must tailor the settings in this file based on your business requirements. The following code illustrates the file content. A description of each setting follows the code.

```
<serverconfig version="1.0" value="false">
    <general-config>
        <!-- Determines whether or not to abort a session transaction
        if the connection to the QXEvents database is lost -->
        <haltOnDisconnect label="Halt application when QXEvents not
        connected" value="false"/>
        <!-- Determines the length of time in seconds to wait before
        trying to re-establish a connection to the QXEvents database
        -->
        <connectRetryTime label="seconds before retry connecting"
        value="300"/>
        <!-- Determines the maximum amount of time spent attempting
        to re-establish a connection to the QXEvents database -->
        <sessionAbortTime label="seconds before abort" value="1800"/>
        <!-- Enter the QXEvents database name together with the full
        path.-->
        <databaseName label="QXEvents Database name." value="mfgsrv/
        db/qxevents"/>
        <!-- Connection Parameters for the QXEvents database. -->
        <connectionParameters label="connection parameters for
        QXEvents" value="-H localhost -S qxevents-service -ld
        alias_qxevents"/>
    </general-config>
</serverconfig>
```

haltOnDisconnect. This setting determines what happens to a client session when it cannot connect to the `qxevents` database.

True: The client session is terminated when it is not connected to a `qxevents` database.

False: The client session can continue creating MFG/PRO event records and storing them in the qad_wkfl table. These events are restored to the `qxevents` database when reconnected.

Set this to true when you need to capture every change in the source application, creating a complete audit trail. Setting this to true prevents a second change to a record before QXO has extracted the record details for the first change. However, a true setting can impact system performance since more detail is required when events are captured.

Set this to false when a complete audit trail is not necessary and it is more important to keep MFG/PRO running than to capture all changes. With this setting, it is possible that when multiple changes to the same record occur close together, the changes are merged into a single event.

Note The possibility of the `qxevents` database being unavailable when the MFG/PRO databases are available is very small.

connectRetryTime. If the client session is disconnected from the `qxevents` database, it attempts to reconnect using the values provided by the databaseName and connectionParameters tags. The connectionRetryTime defines the interval between reconnection attempts.

sessionAbortTime. If haltOnDisconnect is true and the `qxevents` database is not connected, the client continues to attempt to reconnect based on the time specified for connectRetryTime. The sessionAbortTime determines how long to continue these attempts. Once this threshold is reached, the system displays a message directing the user to contact their system administrator.

databaseName. This field specifies the physical location and database name for the `qxevents` database. This value is used by the client session when attempting to reconnect.

connectionParameters. This field specifies any connection parameters that the client should use when attempting to reconnect to the `qxevents` database. Specify the same values entered when creating a new database set and adding the `qxevents` database to it.

See “Add qxevents to a Database Set” on page 44.

Enable QXtend in MFG/PRO

Prior to running QXtend, you must enable the application within MFG/PRO.

1. Start an MFG/PRO session.
2. Navigate to QXtend Outbound Control (36.16.19).
3. Set Enable QXtend Outbound to Yes.
4. Press Go to save and exit.

This completes the QXtend Outbound installation on the MFG/PRO server.

Installing the QXtend WAR File

You can install the QXO client for either a Tomcat or WebSphere application server. For WebSphere, install the QXO UI Webapp using a standard WebSphere install. Additional steps are required for WebSphere environments as described in “Configure WebSphere” on page 61.

The following steps transfer the QXtend Outbound `.war` file to the Tomcat or WebSphere environment:

1. Shut down the Tomcat or WebSphere application server.
2. Mount the QXtend Outbound CD-ROM as described in “Mount the CD-ROM (UNIX only)” on page 30.
3. Repeat the steps under “Install the Outbound Server” on page 31, answering Yes when asked if you want to install `qxo-ui.war` to this machine.
4. You are prompted for the location of your Web application home directory. For Tomcat, this is the *`TomcatInstall`*`\webapps` directory. For WebSphere, this can be any location; however, you must then use WebSphere functions to specify where the `.war` file is. This is covered by the standard WebSphere installation process. Enter the directory and press Enter.
5. You are asked if you want to install the MFG/PRO GUI Client. Typically, you would answer No.
6. A summary of your install choices displays. Confirm your response and the `qxo-ui.war` is installed to your chosen destination.
7. Restart the application server. This automatically unpacks the `.war` file in your environment and starts the qxo-ui Web application.

Set Tomcat Security

1. Shut down Tomcat.
2. Open *`TOMCAT_HOME`*`\conf\tomcat-users.xml` in a text editor.

3 Update or add roles, including user names and passwords:

```
<?xml version="1.0" encoding="utf-8"?>
<tomcat-users>
…..
    <role rolename="qadadmin"/>
    …
    <user username="admin" password="mfgpro" roles=
    "admin,manager,qadadmin"/>
…..
</tomcat-users>
```

4 Save the file and restart Tomcat to load the new configuration.

Configure WebSphere

For WebSphere implementations, after the `qxo-UI.war` install, open the WebSphere Admin page.

1 Open Applications|Enterprise Applications|qxo-ui. Change Classloader Mode to Parent_last and choose Apply.

2 Scroll down to Web Modules and select `qxo-ui.war`. Once again, change Classloader Mode to Parent_last and choose Apply.

3 Choose Save and exit the Admin page.

Creating the logs Directory

On WebSphere, the `logs` directory is not created when `qxo-ui.war` is expanded.

1 Navigate to the WEB-INF directory. On WebSphere this would be:

WebSphereHome`/qxo-ui_war.ear/qxo-ui.war/WEB-INF`

2 Create a directory called `logs`.

Configure the QXtend Outbound Client

Prior to launching the QXtend Outbound client, make the following changes to the `environmentmanager.xml` file. The following instructions assume a Tomcat installation; use WebSphere paths for WebSphere installs.

1. Open *`TOMCAT_HOME`*`\webapps\qxo-ui\WEB-INF\conf\environmentmanager.xml` in a text editor.
2. Set the value of `qxohost` to point to the machine the QXOSession AppServer is running on.
3. Set the value of `qxoport` to the port that the Progress NameServer is running on.
4. Ensure the value of `qxoappserver` is the same as the name of the AppServer created earlier.

See "Configure AppServer on Windows" on page 36.

Installing MFG/PRO GUI Client Updates

If you run MFG/PRO over GUI clients, updates to the client code are required prior to running QXtend Outbound on the client.

1. Mount the QXtend Outbound CD-ROM as described in "Mount the CD-ROM (UNIX only)" on page 30.
2. Repeat the steps under "Install the Outbound Server" on page 31, answering Yes when asked if you want to install updates to the MFG/PRO GUI client to this machine.
3. You are prompted for the location of your MFG/PRO GUI client files. By default this is `c:\mfgsvr\guicli` on Windows and `/home/mfg/guicli` on UNIX. Enter the directory and press Enter.
4. You are then asked to identify the MFG/PRO version (eB, eB2, or eB2.1). Select the version and press Enter.
5. Identify the MFG/PRO service pack level. If you are uncertain, the service pack level is shown in your MFG/PRO log-in screen or is available in the `version.mfg` file in *`MFGPROInstallDir`*. Select the service pack level and press Enter.

6 A summary of your install choices displays. Confirm your response and the MFG/PRO GUI client files are installed to your chosen destination.

Compile GUI Client Code

Prior to running your MFG/PRO GUI client again, you must recompile the code.

1 Run Generate Compile List File from the Programs menu.

   a Enter the Source Directory. This is the location of your MFG/PRO GUI client files entered in step 3 during the install of the client.

   b Enter a Compile List File by name and path; for example, *MFGPROInstallDir*\guicomp.wrk.

   c Choose Generate to create the file.

2 Select Compile Procedures from the Programs menu. Leave everything set to the defaults except:

   a Enter or browse for the Compile List File created in the last set of steps.

   b Set the Compile Propath (where *MFGPROInstallDir*\guicli is the GUI client installation directory).

   For eB:

   MFGPROInstallDir\guicli\qxtend\qxo\src,
   MFGPROInstallDir\us\xrc

   For eB2 and eB2.1:

   MFGPROInstallDir\guicli\qxtend\qxo\src,
   MFGPROInstallDir\xrc

   c Set the destination directory to:

   MFGPROInstallDir\guicli\qxtend\qxo

   d Choose Compile to review the compile summary.

   e Verify the compile setup. Choose Continue to start the compile. A log window displays to show progress.

3 When the compile is complete, choose Close in the log window.

4 Open the `progress.svg` file in *`MFGPROInstallDir`* in a text editor.

   a Locate the PROPATH entry in the [Startup] portion of the file.

   b Enter the GUI client install directory at the beginning of the existing PROPATH.

   c Save the file and exit the editor.

Verifying Application Communication

After completing the installation and configuration of QXtend Outbound, you can use these steps to verify that it is installed and configured correctly.

Note These steps comprise a basic outline that verifies a complete business cycle test. Detailed instructions on executing these steps are included in the chapters that are part of Section 2, "Implementing and Using QXtend Outbound," beginning on page 71.

Verify QXO Server Installation

This test verifies that the QXOServer has been correctly installed, that the required AppServer has been correctly configured, and that the QXO user interface has been correctly installed and configured.

1 Make sure the application server (Tomcat or WebSphere) is running.

2 Make sure the Progress AppServer is running.

3 Make sure the MFG/PRO production and `qxevents` database servers are running.

4 Start a browser session on the client, and set the URL to:

   http://localhost:8080/qxo-ui

5 The QXtend Outbound dashboard should display as in Figure 4.24.

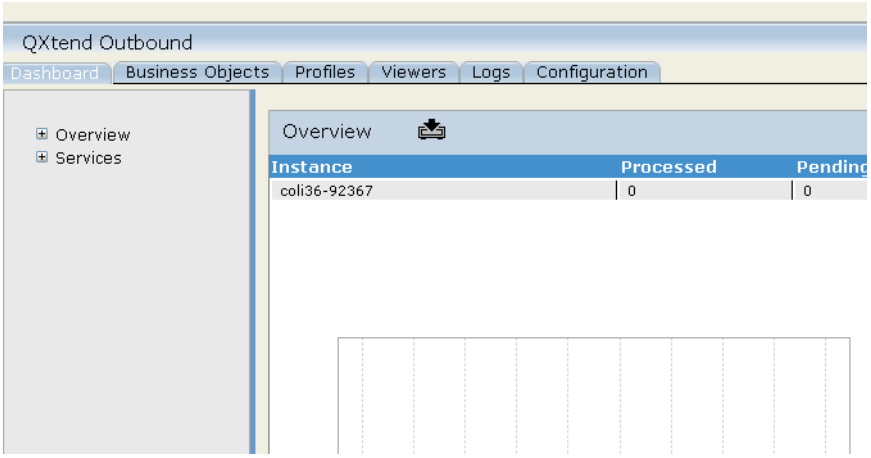


Fig. 4.24 QXtend Outbound Dashboard

See Chapter 5, “Implementing QXtend Outbound,” on page 73 for details about the administrative functions.

If your system is not configured properly, you may see the following message when you try to display the QXO Console:

```
Unable to start Appserver AppServer://servername:9999/
QXOSession appserver com.qad.qxo.proxy.QXOProxy
```

This message indicates that one of the following has occurred:

1 The AppServer has not been started on the server *servername*, possibly because of configuration problems.

2 The configuration file for the QXO-UI (`environmentmanager.xml`) has not been correctly modified to point to the AppServer. Check that:

See “Configure the QXtend Outbound Client” on page 62.

   a The name of the server where the AppServer is running is correct.

   b The correct port is specified for the NameServer with which the AppServer is registered.

   c The name of the AppServer is correct. The default name is QXOSession, but this can be modified if multiple AppServers are running on the same machine.

Verify QXO Server and Source Application Connection

This test ensures that the QXO server connects correctly to the `qxevents` database. It assumes that a suitable QXO-adapted MFG/PRO is available as a source application.

1 Start a browser session on the client, and set the URL to:

http://localhost:8080/qxo-ui

See "Defining Source Applications" on page 80.

2 In the QXO Console, click the Configuration tab and then the Source Applications node in the left-hand tree view. Create a new source application and define its details.

3 Click Event Types under your Source Application and ensure that the event types appear.

If your system is not configured properly, you may see the following message when you try to display the event types:

```
Database not available
```

This message indicates that one of the following has occurred:

1 The database name is configured incorrectly; correct the database name using the Configuration tab.

2 The `qxevents` database server is not running. Verify that the servers are started for the MFG/PRO databases including `qxevents` using your `start.`*`DBName`* script.

If a blank screen displays, this indicates that the database connection parameters for the `qxevents` database are incorrect. Correct the database connection parameters using the Configuration tab.

Verify PROPATH and Event Recording

This test ensures that the MFG/PRO database set has been configured correctly with the correct PROPATH and database connections. It verifies that an event is recorded when a change in the source application occurs.

It assumes that a suitable QXO-adapted MFG/PRO is available as a source application.

1. Start a browser session on the client and set the URL to:

   http://localhost:8080/qxo-ui

2. In the QXO Console, click the Configuration tab and then click Event Types under your source application.

See “Edit Source Application Event Types” on page 87.

3. Edit the source application event types and select the Active check box for the cm_mstr event type.

4. Launch an MFG/PRO session. Use Customer Maintenance (2.1.1) to create a new customer.

   If the following error message appears in Customer Maintenance, the PROPATH is incorrect:

   ```
   ** "filename" was not found. (293)
   ```

   Correct the PROPATH for the QXO database set and create new startup scripts.

See “Add qxevents to a Database Set” on page 44.

5. From your MFG/PRO session, access the Progress Editor. From Tools|Data Dictionary|Database, choose Select Working Database and select the `qxevents` database.

6. Choose Admin|Dump Data and Definitions|Table Contents (.d) file. Choose the ert_app_event table, press Go, and specify an output file name.

7. Open the output file in a text editor and verify that there is one record similar to the following:

   ```
   1 "0x0057d684" "TMP1497" "mfg" "" "cm_mstr"
   2453319.175590277 "mfgprod" 1 3
   ```

If no records display in the file, one of the following may have occurred:

1. The cm_mstr event type is not active. Restart this test and ensure you select the Active check box for cm_mstr in the QXO Console.

2. The source application database schema was not updated correctly during installation. Verify that replication write and replication delete triggers exist for the cm_mstr table. If these triggers do not exist, you must reinstall the adapter to MFG/PRO.

See “Create the qxevents Database” on page 50.

To verify the existence of the triggers:

1. From your MFG/PRO session, access the Progress Editor. From Tools|Data Dictionary|Database, choose Select Working Database and select the `qaddb` database.
2. Choose Database|Reports|Trigger.
3. In the Trigger Report, ensure that the following displays:

```
Table/Field Name  Event     Check CRC  Flags   Procedure
cm_mstr           RP-DEL    no                 cmrd.t
                  RP-WRI    no                 cmrw.t
```

Verify QXtend Outbound Messages

This test ensures that a subscriber to a particular profile receives an outbound message after a change that affects that profile takes place in the source application. It assumes that the MFG/PRO source application has been added and the databases are correctly set up.

See page 99.

1. Load the Customer business object (`Customer.xml`) using the XML Import screen under the Configuration tab in the QXO Console.

See page 95.

2. In the Configuration tab, click Subscribers in the navigation tree and create a new subscriber. Register the default profile for the Customer BO with the subscriber.

See page 90.

3. In the Configuration tab, click Event Services in the navigation tree and create a new event service. Register the MFG/PRO source application with the event service.

See page 93.

4. In the Configuration tab, click Message Publishers in the navigation tree and create a new message publisher session profile. Register the Customer BO with the message publisher.

See page 94.

5. In the Configuration tab, click Message Senders in the navigation tree and create a new message sender session profile. Register the new subscriber with the message sender.

6 In the QXO Dashboard, start the event service, message publisher, and message sender sessions. The event created in the previous test when you created a new customer will be processed by QXO.

See page 133.

7 Verify that a message is sent to the subscriber you created by clicking the Log tab in the QXO Console.

Section 2

Implementing and Using QXtend Outbound

This section covers the implementation and use of QXtend Outbound.

Chapter 5

Implementing QXtend Outbound

The following material covers the steps required to configure QXtend Outbound.

Introduction

QXtend Outbound relies on a sequential processing flow to:

- Gain notification of a change in a source application.
- Extract data from the source application.
- Publish the data as XML QDocs.
- Send the QDocs to subscribers.

To implement this sequence, you must:

- Define your source applications.
- Define an event service to check for and then extract the data.
- Define a message publisher to publish the QDocs.
- Define a message sender to send the Qdocs.
- Define subscribers to receive the QDocs.

Prior to implementing the QXtend Outbound processing sequence, users unfamiliar with Web or HTML applications may want to review the following material. If you are comfortable using a Web-based application, skip to “Defining Source Applications” on page 80.

QXtend Outbound Console

QXtend Outbound displays as a multi-function console that includes graphing capabilities and lets you view, monitor, and configure QXO components through a set of tabs. Figure 5.1 illustrates the initial view displaying the Dashboard tab.

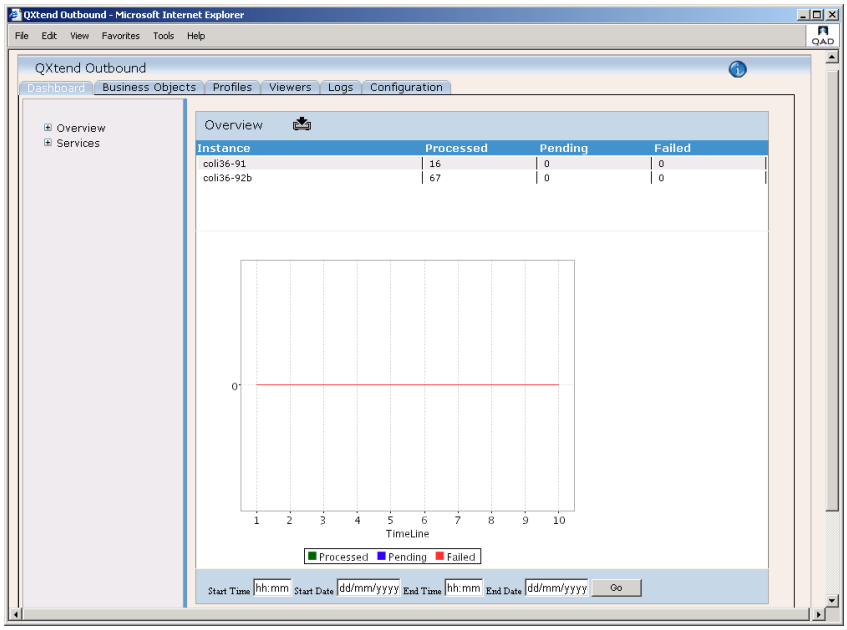


Fig. 5.1
QXtend Outbound Administrative Console

The various tabs provide quick access to the following functions:

Dashboard. Display the current status and statistics about QXO processes and drill down to more detailed views.

Business Objects. Create or modify business objects in a tree view and display data related to each node in the object.

Profiles. Create or modify business object profiles in a tree view and review and modify the fields associated with each node.

Viewers. View messages generated by QXO at various points in the transformation process.

Logs. View logs related to events, messages, and QDocs.

Configuration. Create and update profiles for components of QXO, including source applications, event services, message publisher services, message sender services, and subscribers.

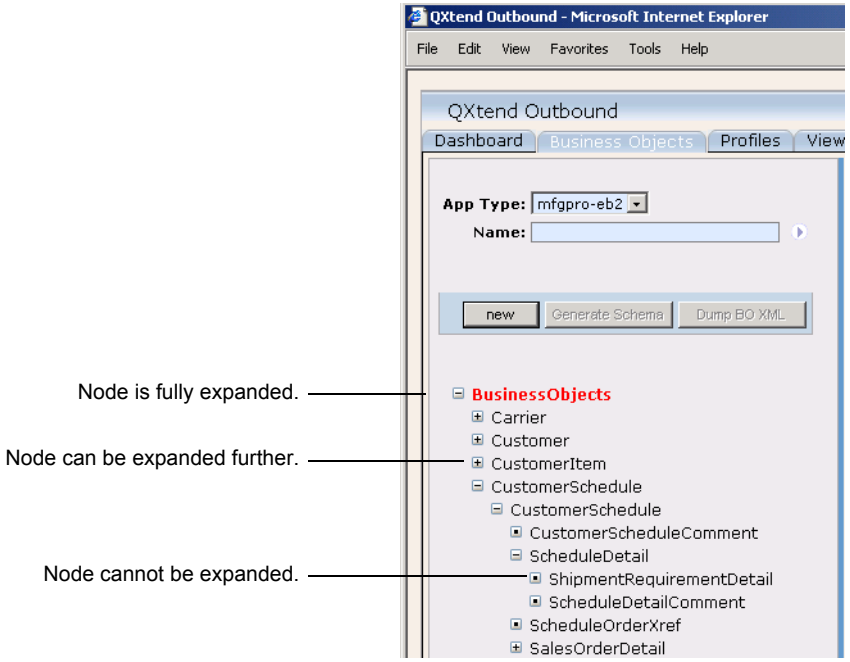
General UI Functions

The QXO administrative user interface uses tab-based HTML forms. This section discusses common aspects of the UI in the various functions, and describes how to use them.

Navigating Tree Views

Many of the QXO administrative screens display items in a tree view. Navigate the tree by clicking nodes to expand or collapse them. Icons indicate if a node is expandable, fully expanded, or cannot be expanded. Figure 5.2 illustrates a navigation tree.

Fig. 5.2
Navigation Tree



Entering Data in HTML Forms

The data entry forms as illustrated in Figure 5.3 show field names and values. Access the fields by clicking or by tabbing through the fields. Fields displaying a red asterisk (*) are required.

Fig. 5.3
HTML Data Entry Form

Filtering Names

On many QXO administrative screens, you can enter a name to filter the items displayed for update or viewing. Figure 5.4 illustrates the Name filter field for business objects. The rules for entering values in other Name fields are the same as for this one.

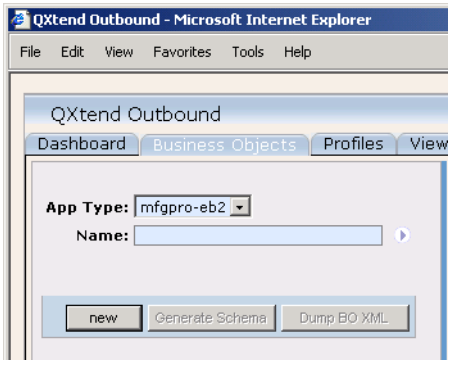


Fig. 5.4
Name Filter Field

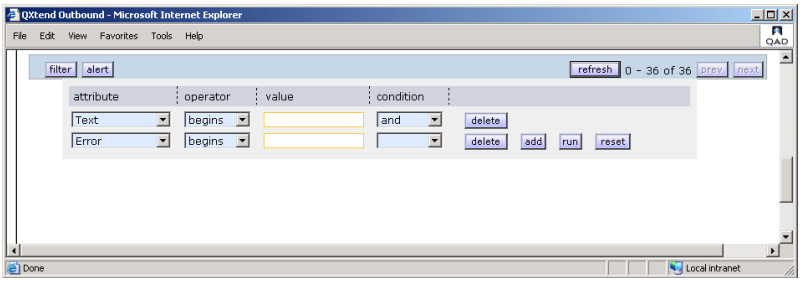
Enter a value in the Name field to determine what displays in the left pane for selecting items to view or update.

- Enter a specific search value to find an exact match. For example, enter Carrier to update the carrier record.
- Enter a search value followed by the wild card (*) to find all items with names that start with that value. For example, enter sales* to see business objects with names that start with sales.
- Leave blank to display all items.

Using Table Filters

On QXO screens where you review data, such as the logs and viewers, you can filter data by one or more sets of criteria. To add or modify filter criteria, click Filter. A screen like the one in Figure 5.5 displays.

Fig. 5.5
Filtering



Use the Add button to add a row for specifying an additional filter. Use the Delete button to remove a filter line. Click Run to apply the filter to the displayed information. Click Reset to close the filter view.

Choose an attribute from the drop-down list. The attributes match the columns of data being displayed. For example, to filter by message text, choose text as the attribute for the filter.

You can use the following operators with filters:

Operator	Description
Text Operators	
begins	The system returns records with values that start with the value you specify.
>	The system returns records with values that start with a value greater than the one you specify.
matches	The system returns records with values that exactly match the value you specify.
Numeric Operators	
>=	The system returns records with values that start with the value you specify or any value higher than it.
<	The system returns records with values that start with the a value less than the one you specify.
=	The system returns records with values that exactly match the value you specify.

<>	The system returns records with values that do not exactly match the value you specify.
<=	The system returns records with values that start with the value you specify or any value lower than it.

Use the Condition setting when you have more than one filter criterion to indicate whether you want values that meet all conditions to be returned (AND condition) or values that meet any one of the conditions (OR condition).

Using Lookups

In many QXO screens, you can associate one type of record with another. For example, you can associate source applications with event services. To make this kind of association, you choose values from a lookup that displays all available records in one list and the records currently associated in another.

These lookups all operate in the same fashion. The screen shown in Figure 5.6 illustrates the lookup for source applications.

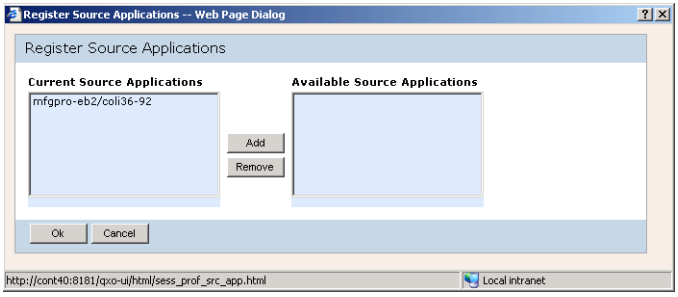
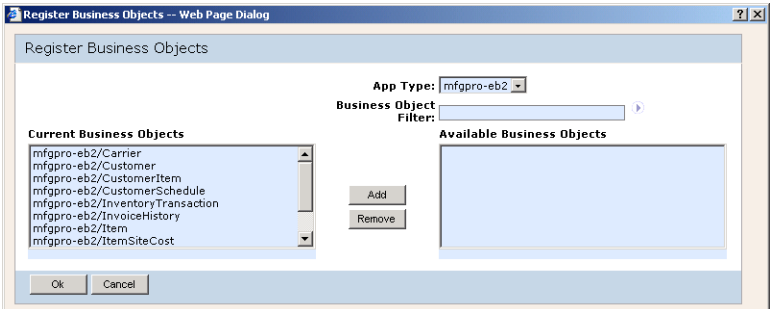


Fig. 5.6 Register Source Applications for a Session Profile

Use the Add and Remove buttons to move selected objects from one list to the other. Click OK when you are satisfied with your changes; click Cancel to discard changes without saving them.

In some lookups, you can filter the values that display in the available records list. For example, when you register business objects, additional fields display that let you filter the list, as shown in Figure 5.7.

Fig. 5.7
Register Business Objects for a Message Publisher



Enter a source application value in the App Type field. Then specify a filter value in the Business Object Filter field—or leave the field blank to display all. Click the arrow to display matching business objects in the Available Business Object list. Then use the Add and Remove buttons to move values from one list to the other as needed.

Defining Source Applications

Source applications are the specific databases from which the QXO event service extracts data. Each source application database that shares a unique schema—for example, all MFG/PRO eB2, SP3 databases—can be managed under a single source application type. For eB2.1 databases, this also extends to multiple domains that share a schema definition.

A source application type also lets you identify the specific event types you want to extract from the databases within that type. Event types are essentially the tables within the source databases.

See “Sequencing Business Dependencies” on page 14.

You can also define business object groups for any source application. Business object groups enforce the correct chronological sequencing of data messages from dependent business objects. Each business object can belong to only one group.

In this first step, you define all the source applications from which you require data. You must also define a source application before you can update business objects or profiles.

1 In the QXO Console, click the Configuration tab. This tab provides access to all the processing implementation services.

2 Click the Source Applications node in the left-hand tree view. If it was not already displayed, the Source Application Types screen displays.

3 Click New in the Source Application Types screen. An entry line is added to the screen.

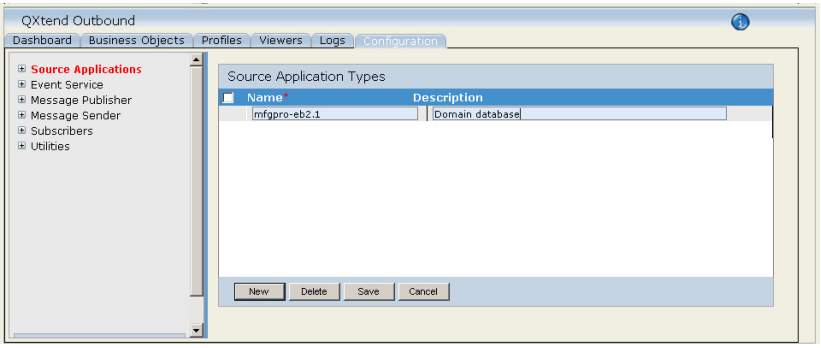


Fig. 5.8 Creating a New Source Application Type

4 Enter the name and a description of the new source application type and click Save.

Important In order to load the QAD standard business objects and profiles, the source application types must be named correctly. The only permissible names are:

- mfgpro-eb
- mfgpro-eb2
- mfgpro-eb2.1

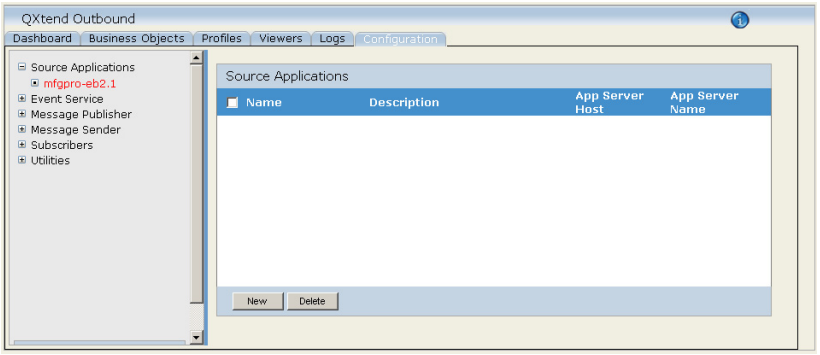
The names must be all lowercase.

If you choose to change the naming convention, the folders containing the QAD-standard business object XML to be loaded must be renamed to the defined source application type. The XML resides in *QXOsrvInstallDir*\wrk\progress\qxtend\boXML.

See “Initial Load of Business Objects and Profiles” on page 99.

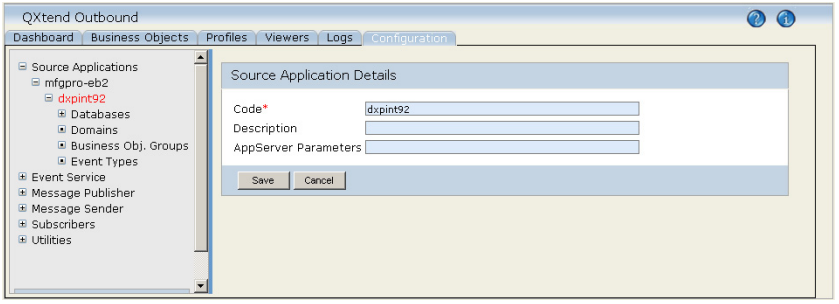
5 Click Source Applications in the navigation tree to open the node. The new source application type appears. Click it to open the Source Applications screen.

Fig. 5.9 Source Applications



6 Click New to display the Source Application Details screen.

Fig. 5.10 Source Application Details



Code. Enter a code identifying a source application from which QXtend Outbound messages originate. You use this code to identify the source application in the event services definition.

Description. Enter a brief description as needed.

AppServer Parameters. In general, enter the NameServer application service (`-AppService`) and the AppServer host (`-H`). For example:

```
-AppService qxoappservice -H co9906
```

In the example, the application service is `qxoappservice`; the host is `co9906`.

The AppServer Parameters field supports the use of calculated field programs documented in “Implementing Calculated Fields” on page 101.

Additional values can be entered in this field. For more information on AppServer parameters, see the Progress documentation.

7 Click Save to save the new source application.

8 Repeat these steps for each source application in the source application type.

Add Source Application Databases

Each source application can contain multiple databases, as long as all the databases in the source application share the same schema.

1 Click the source application in the navigation tree under the source application type. The tree expands to display the source application components.

2 Click Databases under the source application. The Databases overview screen displays.

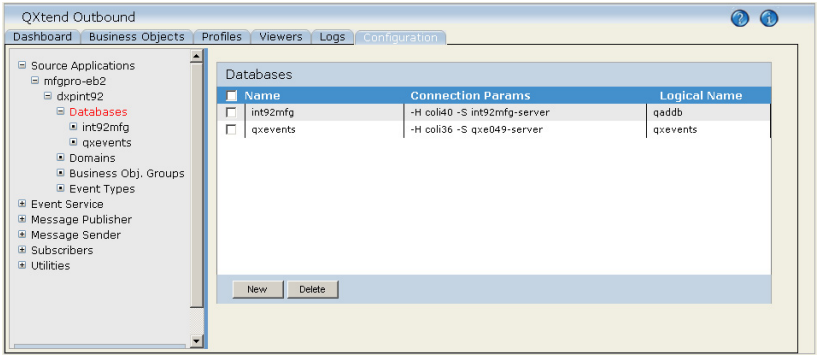
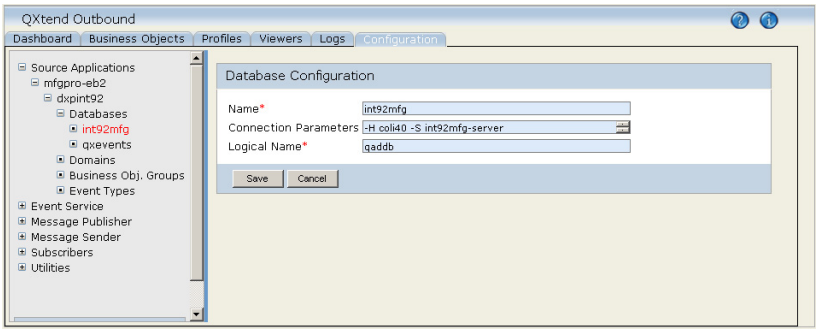


Fig. 5.11
Databases

3 Click New to add a database. You must define the production `qaddb` databases as well as the `qxevents` database.

Fig. 5.12
Database Configuration



Name. Specify the physical file name of the source application database (without the `.db` extension).

Connection Parameters. Specify any Progress connection parameters the system should use to connect to this database. For details, see the *Progress System Administration Guide*.

Logical Name. The `qxevents` database must have the logical DB name `qxevents`.

4 Click Save to save the database definition.

5 Repeat these steps for each database in the source application.

If you encounter reconnection errors, see "Unable to Invoke Proxy Call" on page 39.

Databases are disconnected whenever you modify the database configuration. By default, the database is automatically reconnected when you have finished the modifications and choose Save.

Add Source Application Domains

Domains exist in the MFG/PRO eB2.1 databases. If no domain is defined here for the source eB2.1 database, then data from all domains in the source database is extracted. If subscriber applications require the domain designation of extracted data, you must add one or more domains to the source application. Otherwise, domains are optional.

1 Click Domains under the source application. The Domains overview screen displays.

2 Click New to add a domain.

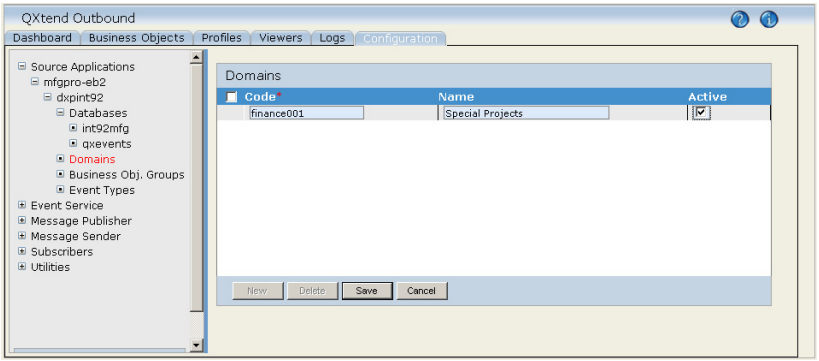


Fig. 5.13
Domains

Code. Enter a code identifying a domain in the source application.

Description. Enter a brief description.

Active. Indicate if this domain is active. If this is not checked, future data processing that accesses this domain will not occur.

3 Click Save to save the domain definition.

4 Repeat these steps for each domain in the source application.

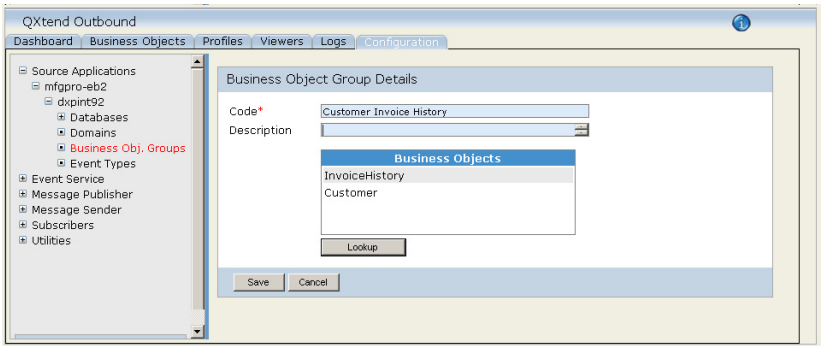
Add Source Application Business Object Groups

The advantage of creating a business object group is that the tables within the group are processed in the sequence you set up. For instance, you may always want a customer's base data processed (ad_mstr and cm_mstr) prior to sales data (so_mstr and sod_det).

1 Click Business Object Groups under the source application. The Business Object Groups overview screen displays.

2 Click New to add a new business object group. The Business Object Group Details screen displays.

Fig. 5.14
Business Object Group Details

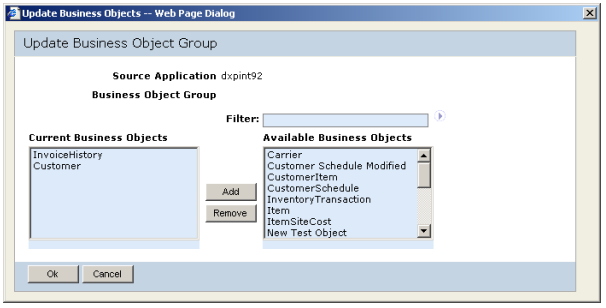


Code. Enter a name for the business object group.

Description. Enter a brief description.

3 Click Lookup to add the business objects to the group. The Update Business Object Group lookup displays, listing available objects.

Fig. 5.15
Update Business Object Group Lookup



4 Optionally, you can enter a filter such as `cust*` to reduce the number of business objects you see. Click the arrow to the right of the filter box to display either filtered or unfiltered business objects.

5 Select business objects from the Available list and click Add. To remove groups, highlight the group name in the Current Business Objects list and click Remove.

6 Click OK to save the business objects.

7 Click Save to save the new business object group.

Edit Source Application Event Types

Source application event types are the tables that are activated for this source application. Before any data can flow through QXO, you must manually select which event types to turn on for each source application you define. The event types are not active by default because there is a performance impact associated with the lookups QXO must do for each event.

You can edit the available events either to make particular tables inactive for this source application, or to require QXtend Outbound to distinguish updates on one or more tables. These tasks are accomplished under the event types node under the appropriate source application.

If you deactivate a table, the trigger for that table in the source application still executes, but no record is written.

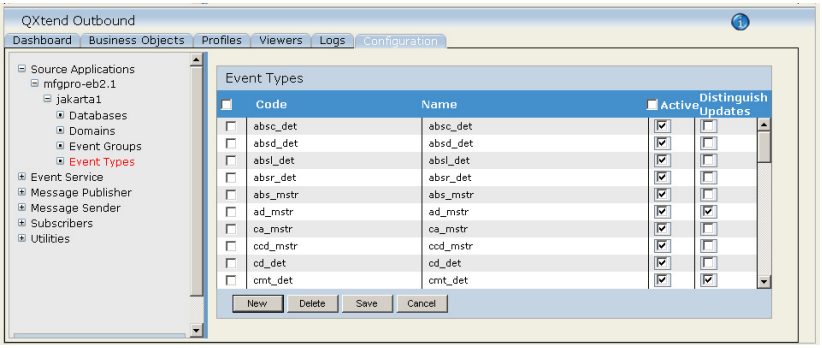
The distinguish updates option indicates that updates to the business object in the source application are captured and messaged individually. Without this option selected, modifications may be consolidated into a single QDoc message depending on how close in time the modification events occur. This option would be used when each transactional change to the data must be captured individually; for example, in an audit trail scenario.

When distinguish updates is enabled and an MFG/PRO user attempts to update or delete a record for which an event is already in the queue, a warning message displays and the user cannot complete the action until the event has been processed.

Important In order to correctly sequence events in a multi-threaded environment, you must use both the distinguish updates option and business object groups. However, the sequence of events for separate tables within a single business object is not guaranteed with this protocol. All such changes may be combined into one or more published QDocs.

1 Click Event Types under the source application. The Event Types overview screen displays.

Fig. 5.16
Event Types



Active. Indicates if data should be extracted for this type of event. If not, the triggers still execute in MFG/PRO, but no data is extracted.

Important You must activate the event types you want to use before any data extraction can take place.

Distinguish Updates. When set, updates to the business object in the source application are captured and messaged individually. Without this option selected, modifications may be consolidated into a single QDoc message.

2 Click Save to save your changes.

Add Source Application Event Types

See “Creating New Business Objects” on page 113.

The business objects supplied with QXO use only a subset of MFG/PRO tables, which correspond to the set of default event types to be activated. If you create new business objects to accommodate your own business requirements that reference data from tables not already defined as events, you must add new event types and activate them.

You will also need to do this if your business object includes custom tables or references tables in a non-MFG/PRO database.

To add a new event, follow these steps:

1 Click Event Types under the source application. The Event Types overview screen displays.

2 Click New. An empty row displays at the top of the screen for input.

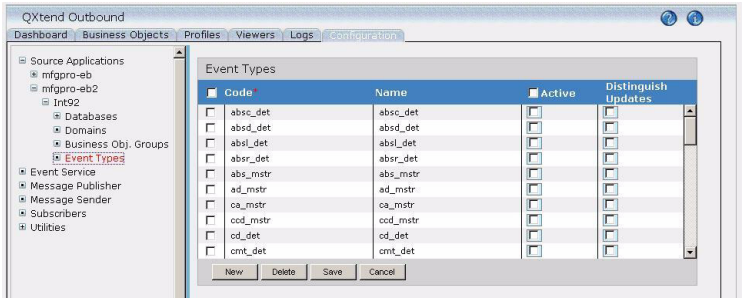


Fig. 5.17 Adding a New Event Type

3 Specify a code and name for the event type. The event code must match exactly the name of the table in the application schema that it is associated with. The name is descriptive text.

4 Select the Active check box to extract data for this type of event.

5 Select Distinguish Updates to create individual messages for each update to the business object in the source application. Without this option selected, modifications may be consolidated into a single QDoc message.

6 Click Save to save your changes.

Additional steps are required to use new event types with business objects. These are described in “Creating New Business Objects” on page 113.

Defining Services

QXO uses three types of services: event, message publisher, and message sender. The profiles for all three types of services contain the same parameters. The only difference is:

- Event services are associated with source applications.
- Message publisher services are associated with business objects.
- Message sender services are associated with subscribers.

You choose the source application, business object, or subscriber using a lookup that displays available values and lets you move names from one list to another.

Define an Event Service

Event services are the QXO processes that initially connect to the `qxevents` database to identify what changes have occurred in the source application, and then to the production database to extract the data and write it to the `qxodb`. The actual process an event service follows is:

- Poll for and identify an eligible event in `qxevents`.
- Identify affected business objects in QXO.
- Build a data object to hold the data.
- Extract the data from `qaddb` into the data object.
- Store the data object in `qxodb`.

Event services are defined to operate with a specified set of source applications.

1. In the Configuration tab, click Event Service in the navigation tree. The Event Services overview screen displays.
2. Click New to create a new event service.

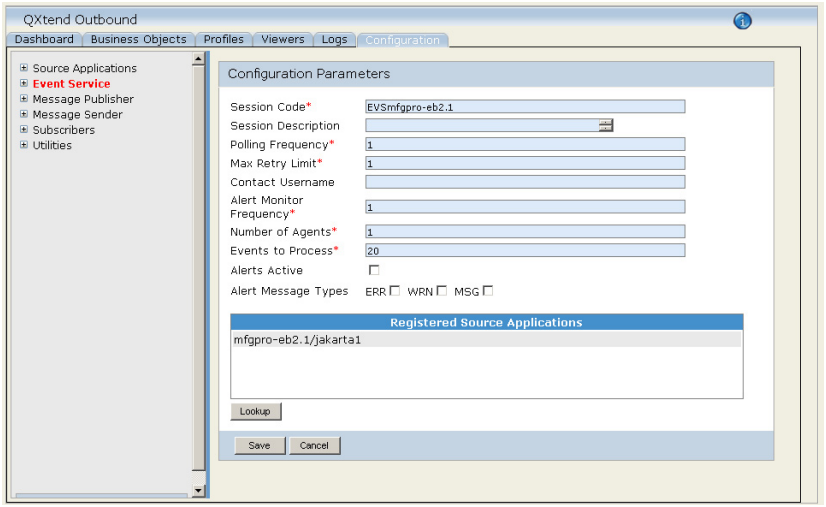


Fig. 5.18
Event Services Configuration Parameters

Session Code. Enter a code (maximum 10 characters) for this service. Using a naming convention such as EVS<*serviceName*> makes it easier to identify the service as an event service later on.

Session Description. Specify a description (maximum 15 characters).

Polling Frequency. Enter the number of seconds for the service to wait between polls to the `qxevents` database.

Max Retry Limit. Enter the maximum number of times during a single session that the service will attempt to reprocess a failed application event (maximum 9999).

Contact Username. Specify the e-mail address of the user who should receive alerts generated for this service. The alerts are sent using the default MAPI e-mail client.

Alert Monitor Frequency. Specify the number of minutes between alert messages when the Alert Message Type is set to MSG and thresholds have been set.

See “Alert Message Types” on page 92.

Number of Agents. Specify the number of agents to start for the service when the AppServer is started.

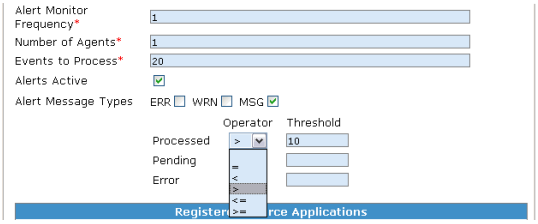
Events to Process. Enter the maximum number of events to be processed by a source application before switching to a different one. This is used only when an event service is defined with more than one source application, and is used to ensure throughput across all registered source applications.

Alerts Active. Indicate if alerts are enabled for sessions started based on this session profile.

Alert Message Types. Check the message types that you want to be raised as alert conditions for this profile. Choose one or all of the following: MSG (used for metric-related alert messages), ERR (errors), WRN (warnings).

When you select MSG, additional options are displayed as shown in Figure 5.19.

Fig. 5.19
Setting MSG Alert Metrics



The MSG alert metrics let you specify alert thresholds for Processed, Pending, or Error conditions. In Figure 5.19, a threshold on processed messages is set. When more than 10 events have been processed by an instance of this event service and the Alert Monitor Frequency is set to 1, an alert will be issued every minute after this. The Alert Monitor Frequency ensures that needless alerts are not issued when processed messages hit 12, 13, 14, and so on.

Operator. Choose the numeric operator for each processing condition.

Threshold. Enter the number of events to occur before an alert is issued.

3 Click Lookup to register source applications with this service. A lookup displays available and currently assigned source applications. Use the Add and Remove buttons to update the lists.

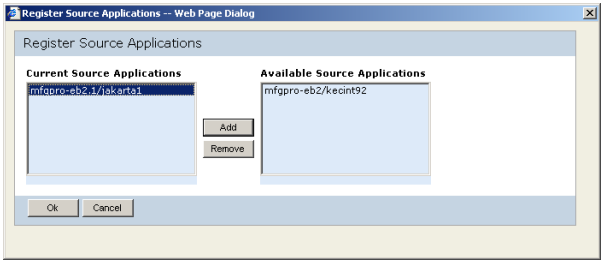


Fig. 5.20
Source Application Lookup

4 Click OK to save and exit the Register Source Applications screen.

5 Click Save to save the event service configuration.

Define a Message Publisher

Message publishers poll the `qxodb` database for new data objects created by an event service. They:

- Poll for and identify an eligible data object in `qxodb`.
- Identify affected business objects in QXO.
- Identify applicable profiles.
- Build an XML QDoc using a profile.
- Store the QDoc in `qxodb`.
- Create a delivery request for each associated subscriber.
- Store the delivery requests in the subscriber's delivery queue.

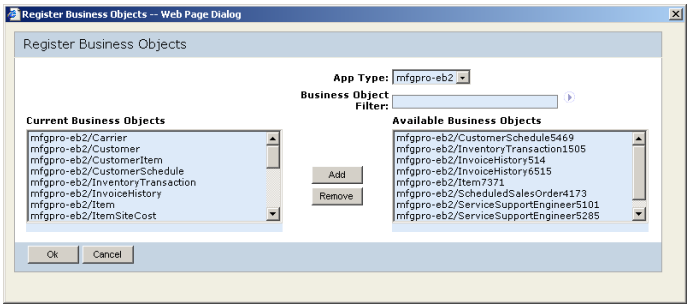
As described in "Defining Services" on page 90, the parameters for message publishers are identical to those for event services.

1 In the Configuration tab, click Message Publisher in the navigation tree. The Message Publishers overview screen displays.

2 Click New to create a new message publisher. Enter values as described in "Define an Event Service" on page 90.

3 Click Lookup to register business objects with this publisher. A lookup displays available and currently assigned business objects. Use the Add and Remove buttons to update the lists.

Fig. 5.21
Business Object Lookup



4 Click OK to save and exit the Register Business Objects screen.

5 Click Save to save the message publisher configuration.

Define a Message Sender

Message senders poll the `qxodb` database for new QDocs created by a message publisher. They:

- Poll for delivery requests in the subscriber's delivery queue.
- Send QDocs to the designated subscribers.

Message senders can forward the QDoc either through a Web service call to a URL, or by dropping the QDoc into a directory. As described in "Defining Services" on page 90, the parameters for message senders are identical to those for event services.

1 In the Configuration tab, click Message Senders in the navigation tree. The Message Senders overview screen displays.

2 Click New to create a new message sender. Enter values as described in step 2 in "Define an Event Service" on page 90.

3 Click Lookup to register subscribers with this sender. A lookup displays available and currently assigned subscribers. Use the Add and Remove buttons to update the lists.

4 Click OK to save and exit the Register Subscribers screen.

5 Click Save to save the message sender configuration.

Defining Subscribers

A subscriber is a delivery point for a QDoc or other published XML document. Another application can be the delivery point, or an instance of QXtend Inbound. Subscribers have an associated delivery method: either a directory location or a Web service. The subscriber is responsible for retrieving the outbound document.

To bring a new subscriber up to date, a force publish function is available. You can also restrict specific source applications to ensure a subscriber sees only the data intended for them.

1 In the Configuration tab, click Subscribers in the navigation tree. The Subscribers overview screen displays.

2 Click New to create a new subscriber.

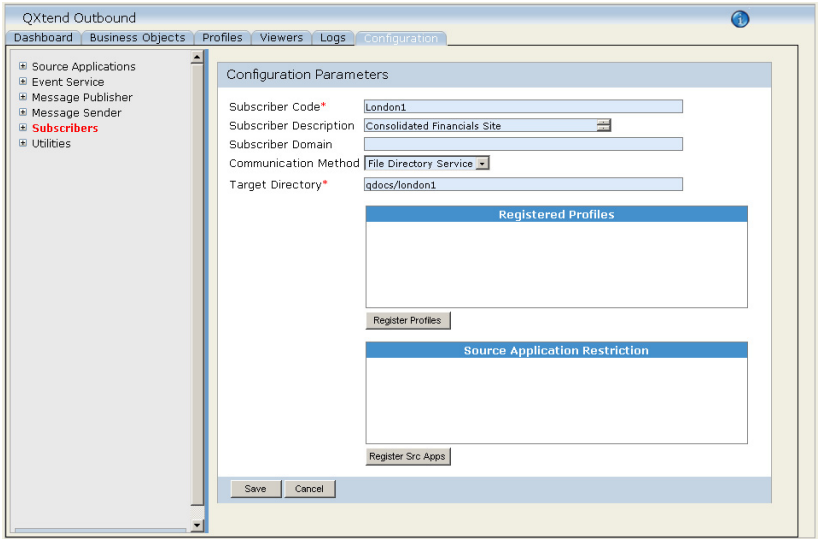


Fig. 5.22 Subscriber Configuration Parameters

Subscriber Code. Enter a code (up to 15 characters) that identifies the external system or application.

Subscriber Description. Enter a description (up to 35 characters).

Subscriber Domain. Identify any domain this subscriber requires (for eB2.1 source applications only).

Communication Method. Choose the method used to send messages.

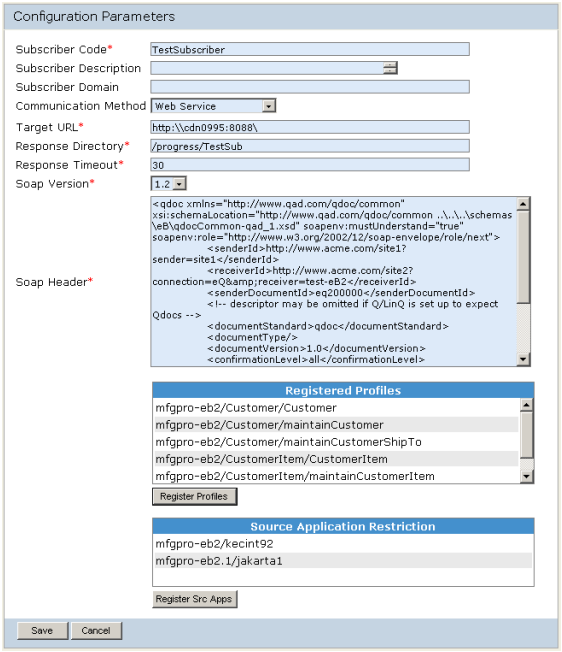
File directory service: In this method, outbound QDocs are placed in a directory for the subscribing application to read. For this method, you specify a target directory.

Web service calls: In this method, outbound QDocs are delivered to a URL where the subscriber is set up to listen for incoming documents. For this method, you specify a target URL.

For the fields specific to a Web Service subscriber, see Figure 5.23 and the field descriptions following it.

Target Directory. For file directory service, specify the location on the file system where outbound QDocs for this subscriber should be placed. If a full path is not specified, QDocs are placed in the current working directory.

Fig. 5.23
Web Service URL Subscriber Entry



Target URL. For Web service, specify the URL where outbound QDocs for this subscriber are sent. This URL must be set up as a Web service.

Response Directory. Specify the location on the file system where response QDocs for this subscriber are placed. If a full path is not specified, responses are placed in the current working directory.

Response Timeout. Enter a number of seconds for QXO to wait for a response from the subscriber. If no response is received in that time period, an error is raised for the QDoc in QXO.

SOAP Version. Web services require SOAP headers on all incoming communications in order to validate the sender and destination; QXO supports SOAP versions 1.1 or 1.2.

SOAP Header. Copy the SOAP header from a valid QDoc directed to this subscriber and paste it in here. The SOAP header provides sender and destination details.

A standard SOAP header is:

```
<qdoc xmlns="http://www.qad.com/qdoc/common" xsi:schemaLocation=
"http://www.qad.com/qdoc/common ..\..\..\schemas\eB\qdocCommon-
qad_1.xsd" soapenv:mustUnderstand="true" soapenv:role="http://
www.w3.org/2002/12/soap-envelope/role/next">
    <senderId>http://www.acme.com/site1?sender=site1</senderId>
    <receiverId>http://www.acme.com/site2?connection=
    eQ&amp;receiver=test-eB2</receiverId>
    <senderDocumentId>eq200000</senderDocumentId>
    <!-- descriptor may be omitted if Q/LinQ is set up to expect
    Qdocs -->
    <documentStandard>qdoc</documentStandard>
    <documentType/>
    <documentVersion>1.0</documentVersion>
    <confirmationLevel>all</confirmationLevel>
    <dateTimeCreated>2002-07-10T09:00:00</dateTimeCreated>
    <senderDocumentRef>eqso5678</senderDocumentRef>
    <senderDocumentRef>1</senderDocumentRef>
    <receiverDocumentRef>SO14821</receiverDocumentRef>
    <receiverDocumentRef>2</receiverDocumentRef>
</qdoc>
```

If you require a specific user-defined SOAP header for a specific Web service requirement, enter that header here.

3 Identify the profiles used to format QDocs for this subscriber by clicking on Register Profiles. A standard lookup displays, listing available profiles and those currently assigned. Use the Add and Remove buttons to move selected profiles from one list to the other.

4 If you want to restrict this subscriber to receiving data from one or more given source applications, click Register Src Apps. Add the source applications this subscriber should receive data from. When none are specified, the subscriber can receive from all valid source applications.

5 Click Save to save and exit the Configuration Parameters screen. The new subscriber displays in the overview screen.

If the subscriber is new, you can click Force Publish in the Subscriber overview screen to force the publication of a QDoc for every record associated with a profile or set of profiles. This operation runs through every existing notification in the `qxevents` databases and queries the source applications accordingly. It does not query the source application for every customer, or sales order, or other business object or profile related to the subscriber.

You can also click Generate Schema to generate the XML schemas for every profile registered to this subscriber.

Subscriber Visibility

Some planning and consideration are needed when applying filters to profiles and business objects. Depending on what filters you define in the business objects or, more typically, in the profiles, a subscriber may have interrupted visibility to some data changes.

Example The table shows the QDocs generated for one sales order for two different profiles; one receives all sales orders and the other filters only for sales orders over $5,000.

Table 5.1 Subscriber Visibility Scenario

Description	Price	Subscriber 1: All orders	Subscriber 2: Over $5,000
Initial order	$4933	✓	
Add item	$5345	✓	✓
Reprice with 10% discount	$4810	✓	
Add on-site assembly	$6250	✓	✓
Split assembly to separate SO	$4810	✓	

Subscriber 1 sees all the changes, while Subscriber 2 only sees the two changes that temporarily raise the so_price value above $5,000.

Importing XML

Under the Utilities node in the Configuration tab, you can import XML documents. This is a required step during initial system implementation to populate the database with QAD-supplied default data, and can be used to transfer XML business object and profile definitions from one QXtend Outbound instance to another.

See “Managing Business Object XML Files” on page 122 for information on dumping XML from QXtend Outbound.

Initial Load of Business Objects and Profiles

Prior to running this load, the source applications must be set up in QXtend Outbound using the correct naming convention.

Important In order to load the QAD standard business objects and profiles, the source application types must be named correctly. The only permissible names are:

- mfgpro-eb
- mfgpro-eb2
- mfgpro-eb2.1

The names must be all lowercase.

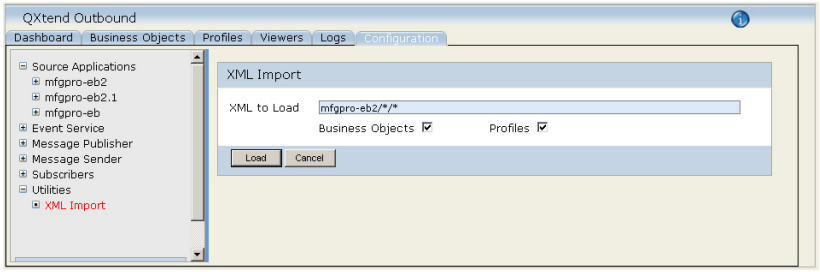
The standard QAD objects to be loaded reside in:

```
QXOsrvInstallDir\wrk\progress\qxtend\boXML\
SourceAppTypeName\BusinessObjectName\ProfileName
```

This is also the directory location and naming convention used when you export business objects and profiles as described in “Managing Business Object XML Files” on page 122. You must copy the files from the originating QXO server to the target server prior to loading them into another instance of QXtend Outbound.

1 In the XML Import screen under the Configuration tab, enter the source application type as defined in Source Applications, and two asterisks to denote a full import for both business objects and profiles. For example, for eB2 enter:

Fig. 5.24 XML Import of QAD Business Objects and Profiles

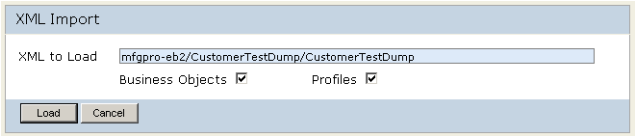


2 Select both checkboxes and choose Load. A processing note appears during the load.

3 After the load, messages display to inform you of the successful loads of each object.

To load specific files, such as those originating from another instance of QXtend Outbound, follow the same steps, but enter the source application and the specific object names you want to import.

Fig. 5.25 XML Import of Custom or Individual Objects



Implementing Calculated Fields

Calculated fields require a specific return value in a specific format, and correct connection parameters for the AppServer. See “Defining Source Applications” on page 80 for details on setting up AppServer parameters.

A sample program, `samplecalc.p`, is shipped with QXtend Outbound source code. The full program is listed here.

```
define input parameter row-id as character no-undo.
define output parameter cResult as character no-undo.

assign
    row-id = replace(row-id,chr(230),":")
    cResult = subst("Result for [&1]",row-id).
```

This program wraps the input with “Result for [<input>]”. This program is provided as an example of the format that a calculated field program should take.

The important part is the signature and the input and output parameters. The row-id is a colon-delimited (:) list of key field values used to determine the record that should be acted on; it is not the rowid in the Progress sense.

The AppServer Parameter field on the source application configuration contains all the necessary parameters for Progress to connect to the AppServer. A typical set of AppServer parameters might be:

```
-AppService qxoappserver -H co9906
```

This allows QXtend to access the calculated field program on an AppServer named `qxoappserver` on the machine `co9906`. The contents of this field are executed in the code like the following:

```
create server hAppServer.
hAppServer:connect("-AppService qxoappserver -H co9906").
run value("myprogram.p") on hAppserver (input c_id, output
c_result).
```

If you are implementing a calculated field program that accesses the MFG/PRO databases, QAD recommends that the program use an application server separate from the one defined for QXtend Outbound. For simple calculated field programs where the logic applied is local to the calculation program, it is safe to use the same application server as QXtend Outbound.

Calculated Fields Program Example

The following program used by QXtend Outbound is an example of a calculated field program. The code converts the value stored in a database field into another format in the outgoing QDoc. The resulting format is suitable for use in an incoming QDoc through QXtend Inbound.

```
/* $RCSfile: cp-pipartcode.p,v $                                      */
/* Copyright 1986-2004 QAD Inc., Carpinteria, CA, USA.                */
/* All rights reserved worldwide.  This is an unpublished work.       */
/*                                                                    */
/* Description: Checks the value of the pi_part_code field            */
/* extracted from MFG/PRO. If it matches qadall variable we           */
/* place a blank for pi_part_code in the published business object    */
/*                                                                    */
/* $Id:$                                                              */
/*                                                                    */
/*====================================================================*/

define input parameter row-id as character no-undo.
define output parameter cResult as character no-undo.

define variable qadall as character initial "qadall--+--+--+--+" no-
undo.
define variable piPartCode as character initial "pi_part_code" no-
undo.
define variable i as integer no-undo.
define variable c-string as character no-undo.
define variable fieldName as character no-undo.

do i = 1 to num-entries (row-id,CHR(230)):

    c-string = entry(i,row-id,CHR(230)).
    fieldName = entry(1,c-string,':').

    if fieldName = piPartCode then do:
        if entry(2,c-string,':') = qadall then do:
            cResult = "".
        end.
        else do:
            cResult = entry(2,c-string,':').
        end.
    end.
end.
```

Chapter 6

Managing Business Objects

The following material covers implementation of QXtend Outbound business objects.

Introduction

A business object is a set of related data, such as the data that makes up a sales order or a customer record. In the source application, the data that makes up a business object usually resides in a set of related tables, even though this set of tables may be maintained in a single menu program. In MFG/PRO and most other source applications, menu programs are the closest representation of a business object.

You define a business object in QXtend Outbound that encompasses all the possible attributes of a particular type of document, order, or data set. You define which source application tables you want, how the tables are related to each other, and which fields in each table to include. You can define custom fields as well, to contain either a fixed value every time the business object is updated, or a value calculated at each update.

However, not all subscribers for your system need the entire business object. Most subscribers need distinct components of a business object; shipping needs address and inventory data, sales needs the sales figures, accounting needs sales and tax data, and so forth.

To select which components of a business object are sent to which subscribers, QXtend Outbound lets you define profiles. The way you define a profile is nearly identical to the way you define a business object.

Business objects and profiles are closely linked. When you create or copy a business object, a default profile with the same name is automatically created for it. When you add or delete tables or fields from a business object, the change is automatically reflected in the associated profiles as well. If you change the name of a business object, a new default profile with the same name as the new business object name is created for it. Any existing profiles associated with the original business object remain unchanged.

QAD and Custom Business Objects and Profiles

QXO is shipped with a selection of critical business objects and profiles. These can either be used as they are—or more likely—copied and used as templates for your own business objects and profiles.

For a listing of standard QXO business objects, see Table 1.1 on page 12.

Important Only an experienced professional with an understanding of the target application database structure should customize QAD-supplied business objects. QAD does not support custom or non-QAD-defined business objects. To ensure support for your customizations, engage QAD Services personnel to create new or modified business objects.

Typically, a business object has one primary table and other related tables that are joined to it. The business object definition identifies the join relationships. You can also specify filters to limit the data retrieved from the joined table. Then for each field in the table you can specify:

- Whether the field is always published regardless of profile settings
- Whether the field is part of a primary table index
- Any fixed constant field values
- Programs to call to return the value for the field

When you start with a copy of a QAD business object or build your own, you can add additional tables and fields. You can also add your own custom fields to tables. These represent static values that you want to include whenever data from this business object is published. If, for example, you are designing a business object that will create an outbound QDoc that corresponds to an inbound QDoc, you might need to include values for fields that control the behavior of the MFG/PRO UI and that are not stored in the database.

To streamline the creation of business objects, you can copy and paste data objects from one business object to another. For example, if you are creating a new object that references an address, you can copy the address data object associated with a different business object and paste it into the one you are creating.

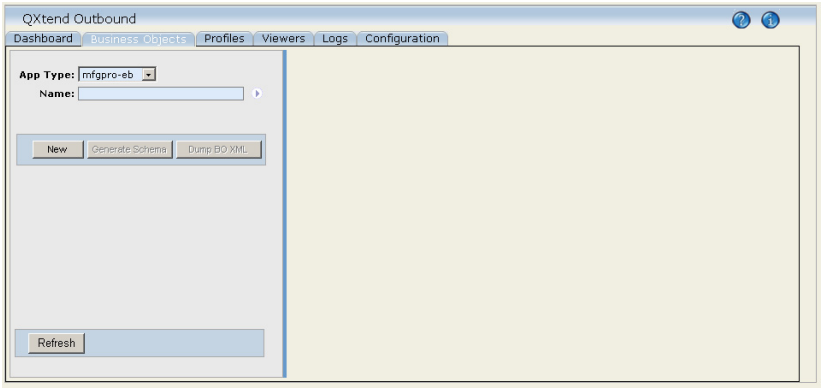
Profiles are views of business objects tailored for the requirements of specific subscribers. They are the means you use to select which components of a business object are sent to subscribers. The method for

defining a profile is nearly identical to the method for defining a business object; however, unlike business objects, profiles always start as a copy of a default profile.

Viewing Business Objects

When you click the Business Objects tab, the screen illustrated in Figure 6.1 displays.

Fig. 6.1
Business Objects



See "Defining Source Applications" on page 80.

App Type. Select the source application type from the drop-down list. Source applications are defined in the Configuration tab. The last selected value is saved and used as the default entry the next time this screen is viewed.

See "Filtering Names" on page 77.

Name. Use the Name field to find business objects to view or update.

Click the arrow next to Name to display all available business objects with the Name field left blank, or to search within the available business objects for objects that match your Name entry. Objects display in a tree view as shown in Figure 6.2.

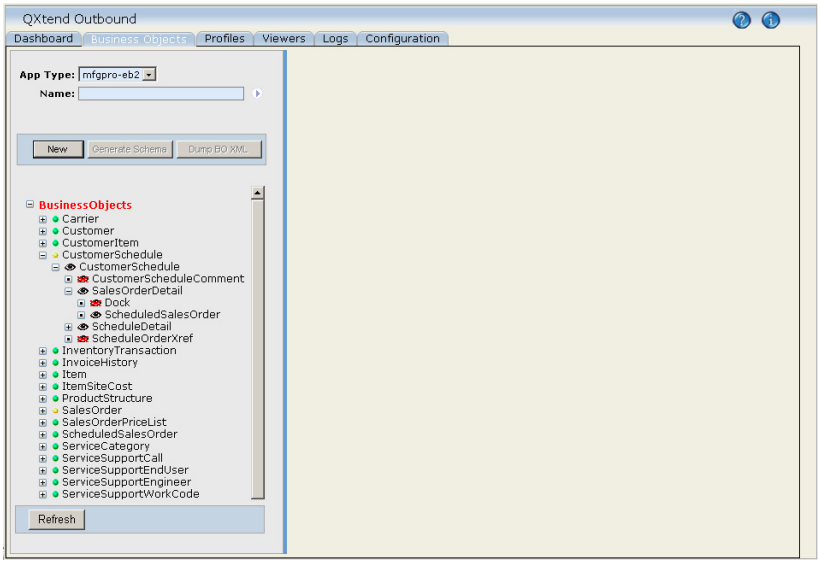


Fig. 6.2
Business Object Tree View

Performance Considerations

Each business object contains one or more data objects maintained in a hierarchy. The entire set of data objects under a business object is referred to as a *business object tree*. Since QXtend Outbound must search up and down the tree for different functions, each tree is validated in both directions during the initial load of business objects.

If the available indexes in one of the source application tables does not sort the records correctly for the searches being performed, whole-index search may occur. In these instances, the entire index may be scanned. The search still works but takes much longer to complete.

When you are populating a business object tree with a data set—for example, during a forced load of data from your source application to one or more subscribers—QXtend Outbound knows the business objects required, but not all the data objects. In this case, it searches down the tree. Where the indexes are inadequate, a whole-index search occurs. However, these are generally one-time events and are not a performance concern.

When an event occurs in a source application, QXtend Outbound knows which data objects are impacted, but not which business objects they belong to. In this case, the search goes up the tree. These searches occur during production processing and are of greater concern. Where the indexes are inadequate, a reverse whole-index search occurs.

Business Object Validations and Data Watches

Both types of searches are tested during business object validation. A validation occurs against the connected source applications during the initial load of business objects. The result of this validation displays as an icon next to each business object in the business object navigation tree.

- A green dot shows that the validation was successful.
- A yellow dot displays when reverse whole-index searches are required for one or more data objects in the business object tree during one or both of the validation passes.
- A red dot displays when invalid joins appear, or when a business object has not been validated.

To control which data objects are watched, a Data Watch field appears on each data object. QAD has set this option for the default business objects to minimize the number of reverse whole-index searches and yet retrieve all essential business data. Two business objects—Customer Schedules and Sales Orders—still have unavoidable reverse whole-index searches.

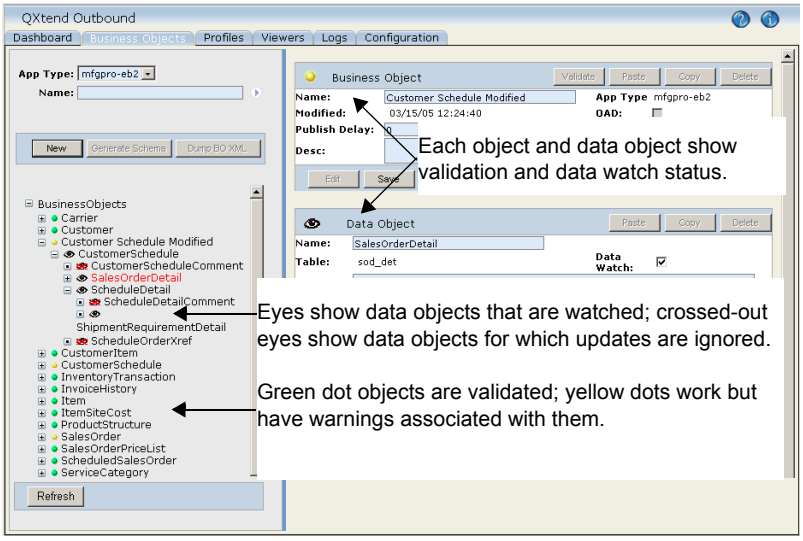


Fig. 6.3
Validations and Data Watch Symbols on Business Objects

When you open one of the business object nodes in the navigation tree, each data object has an eye icon.

- An eye alone means that the data object is watched. A watched data object is aware of updates to that object's tables in the source application.
- An eye crossed out means that data object is unwatched. An unwatched data object ignores updates to the object tables, but the data for that object is still retrieved when events affect other data objects in the parent business object.

To modify these or other business objects, see the next section.

Modifying Business Objects

Since you cannot edit the QAD-supplied business objects, you must either copy an existing business object and modify it, or create a new one. When you create or copy a business object, a default profile of the same name is also created. To create a copy and modify a business object, follow these steps.

1 Select the source business object from the tree view. If it is a QAD-supplied object, the only button active is Copy.

2 Click Copy. The new object is created including all related data objects. A new name is provided for the object by default.

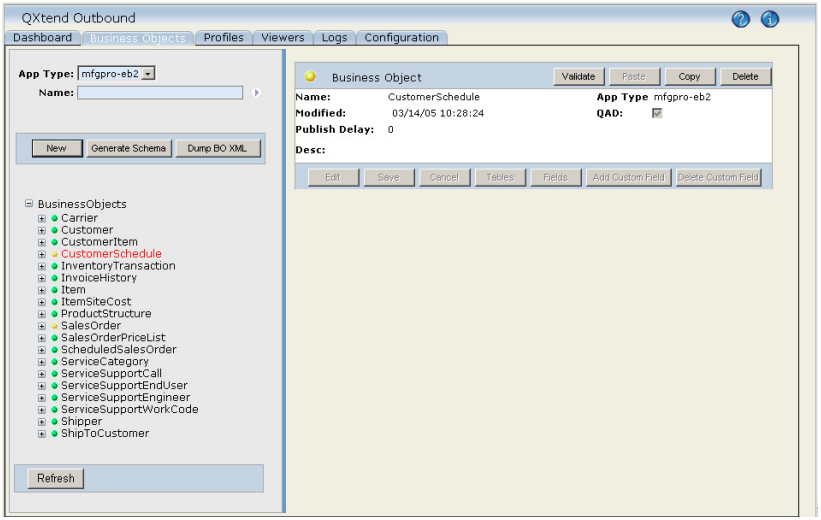


Fig. 6.4 Business Object Definition Showing the Green (Validated) and Yellow (Warning) Dots

3 Enter values for the new business object as follows:

Name. Update the name as needed.

Modified. No edit allowed. This displays the date and time this business object was last saved.

Publish Delay. Set this to a nonzero value for business objects that may generate multiple notifications to the `qxevents` database prior to completion of data entry. For example, sales order entry creates one

or more notifications as different tables are updated before data entry is complete. Setting the value to 60 seconds or more allows time for data entry to complete and only the final data to be exported.

Desc. Enter a description of the object. Consider including the creation date, object author, and purpose.

4 To add or delete tables from the copied business object, click Tables. Update Data Objects displays. Click the arrow next to Table Filter to populate the Available Tables list.

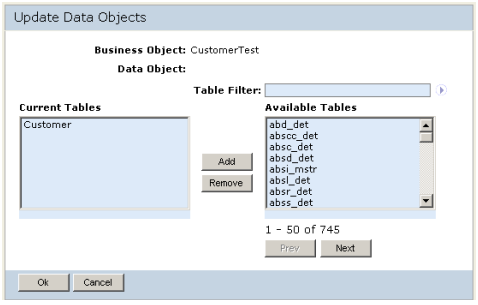


Fig. 6.5
Business Object Table Selection

5 Select tables in either list and use the Add or Remove buttons to move the tables between lists. When the Current Tables list contains the tables you require for the business object, click OK. You return to the Business Object screen.

Note If you need to add new tables that are not listed, review the instructions in “Creating New Business Objects” on page 113.

Prior to using the new business object, you must validate it and optionally modify the Data Watch attribute on the data objects to improve performance.

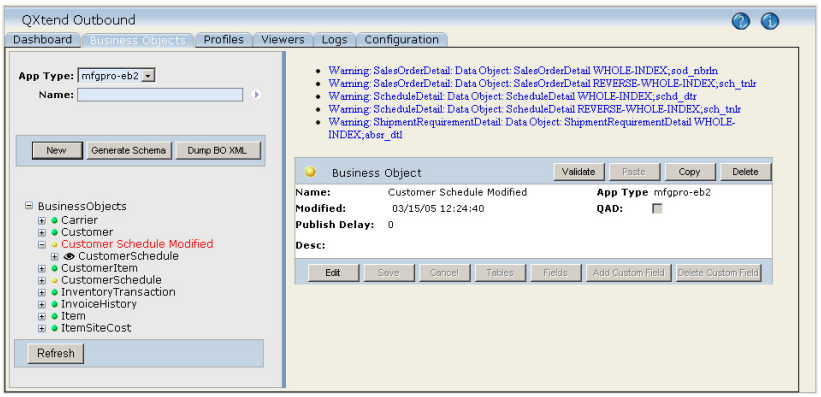
Validating and Optimizing Copied or New Business Objects

When you copy or create a business object, the new object has a red dot next to it in the navigation tree. This means the business object has not yet been validated; it cannot be used until it has been validated.

1 On any new or copied business object, click Validate. Any validation warnings or errors appear above the business object definition. The whole-index warnings are WHOLE-INDEX for top-down searches

where the business object is known but not the data objects, and REVERSE WHOLE-INDEX where the data objects are known but not the business objects they reside in.

Fig. 6.6 Validation Messages for a Copied Business Object



For any data object you received a REVERSE WHOLE-INDEX warning on, determine whether you require refreshed data from an update to the object. For example, on a sales order object, you may not want to refresh data when comments are updated, but do want to when addresses and sales data are changed.

2 Select the data object in the navigation tree to open it.

3 Click Data Watch on or off as needed. When turned off, events in the data object's source application tables are ignored. However, the data for the object is retrieved when events occur in other watched data objects for the business object.

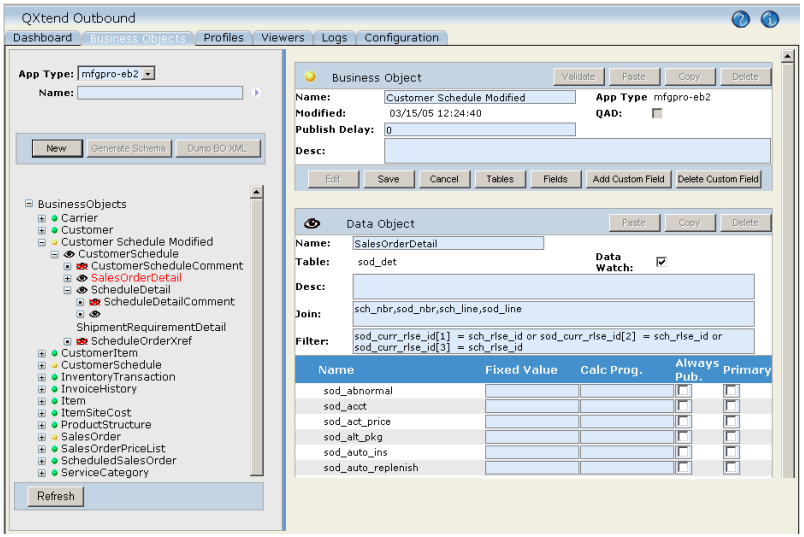


Fig. 6.7
Data Watch Selected on the SalesOrderDetail Data Object

4 Click Validate again to verify your changes.

5 Click Save to save the new business object. The screen redisplays with the Edit button available.

You can edit the modified business object at any time including the object name, publish delay, description, and tables.

Creating New Business Objects

You can create new business objects to accommodate your own business requirements. If you do this, you should keep the following considerations in mind.

- QXtend is installed with a set of event types. These event types represent a subset of MFG/PRO tables that already include schema triggers. If you need to reference data in an MFG/PRO table that is not included in this subset, you must add schema triggers to the table.
- QXtend is also installed with trigger files for every standard MFG/PRO table. Only the subset of these trigger files associated with the standard event types is compiled. If you add triggers to a table, you must also compile the associated trigger files.

- QXtend is configured to work with standard MFG/PRO tables. However, it can be extended to work with business objects based on custom tables or in a side database or a non-MFG/PRO database. A non-MFG/PRO database must be Progress based and support the use of triggers. To implement these kinds of business objects requires that you create your own trigger files, since these are not included when QXtend is installed.

Instructions are provided here for creating a new standard business object and creating business objects for custom tables or non-MFG/PRO tables.

Creating a Standard MFG/PRO Business Object

Use the following steps to create a new business object that references data from standard MFG/PRO tables.

1. In the left-hand Business Objects frame, click New. The Business Object definition screen displays.
2. Enter values as described in step 3 under “Modifying Business Objects” on page 110.
3. Use the Tables button to add tables to the object. Click OK to save the table definitions.
4. After the new tables appear as data objects in the navigation tree, click on Validate to test the new business object tree.

   If you receive warnings, use the instructions in “Validating and Optimizing Copied or New Business Objects” on page 111 to modify data watch settings.
5. Click Save to save the new object.
6. Ensure that all tables included in the new business object are active event types.

   a. If the table is defined as an event type, all you need to do is follow the steps in “Edit Source Application Event Types” on page 87 to activate it.

   b. If the table is not already defined as an event type, follow the steps in “Add Source Application Event Types” on page 88 to add it. Then complete steps 7 and 8.

7 If your business object includes data from one or more tables that are not already in the subset of tables that includes triggers, you must add replication write and replication delete triggers to the MFG/PRO schema for each table.

a Locate the following file and open it in a text editor:

MfgproInstallDir\qxtend\qxo\progress\mfgpro\src\df\delta.df

b Remove existing entries.

c Add an entry for each new table you are activating. The name of the replication write trigger is based on the table name prefix with `rw.t` appended; the delete trigger appends `rd.t`. For example, to activate the abs_mstr table, create the `absrw.t` and `absrd.t` triggers.

```
UPDATE TABLE "abs_mstr"
TABLE-TRIGGER "REPLICATION-WRITE" NO-OVERRIDE PROCEDURE
"absrw.t" CRC "?"
UPDATE TABLE "abs_mstr"
TABLE-TRIGGER "REPLICATION-DELETE" NO-OVERRIDE PROCEDURE
"absrd.t" CRC "?"
```

d Launch MFG/UTIL and choose Database|Load Database Schema (.df) File.

e Connect to your MFG/PRO qaddb database.

f Select the `delta.df` file you modified with the triggers and choose OK.

8 After adding schema triggers, you must compile the associated trigger files.

a Create a compile list file named `NewTriggers.wrk` in:

QXOMFGsrvInstallDir\qxo\src

b List all of the new trigger files that correspond to the triggers you are adding to the schema. In the previous example, the work file would list:

```
absrw.t
absrd.t
```

See the required compile options in Table 4.3 on page 56.

c Launch MFG/UTIL and choose Programs|Compile Procedures. Compile the work file specifying the new compile list name.

Creating a Business Object for Custom Tables

Use the following steps to create a new business object that references data from custom MFG/PRO tables, a custom side database, or a non-MFG/PRO Progress database.

1 In the left-hand Business Objects frame, click New. The Business Object definition screen displays.

2 Enter values as described in step 3 under “Modifying Business Objects” on page 110.

3 Use the Tables button to add tables to the object. QXtend reads all the tables from the connected database and displays them for selection. Click OK to save the table definitions.

4 Click Save to save the new object.

5 Create event types for each table to be referenced using the instructions in “Add Source Application Event Types” on page 88. Ensure that the new event types are active.

For each table associated with an active event type, add a replication write and replication delete trigger to the schema. Create a `.df` file with an entry for each table you are activating using the instructions in step 7 under “Creating a Standard MFG/PRO Business Object” on page 114 as a model.

For custom MFG/PRO tables, you can use MFG/UTIL to load the `.df` file. For other applications, use the tools supplied with your application.

6 Trigger files are provided with QXtend Outbound for all standard MFG/PRO tables. For non-standard tables, you must create replication write and replication delete trigger files for each affected table. Use the following code samples as a model.

Trigger procedure for replication write for <table-name>:

```
/*Record Outbound Event*/
{qxoevent.i
    &TABLE-NAME = '<table-name>'
    &ROW-ID = string(rowid(<table-name>))
    &TRIGGER-TYPE = 'WRITE'}.
```

Write example for MFG/PRO custom table `cust_det`:

```
TRIGGER PROCEDURE FOR REPLICATION-WRITE OF cust_det.
/* RECORD OUTBOUND EVENT. */
{qxotrig.i
    &TABLE-NAME = 'cust_det'
    &ROW-ID = string(rowid(cust_det))
    &TRIGGER-TYPE = 'WRITE'}.
```

Replication delete example for `cust_det`:

```
TRIGGER PROCEDURE FOR REPLICATION-DELETE OF cust_det.
/* RECORD OUTBOUND EVENT. */
{qxotrig.i
    &TABLE-NAME = 'cust_det'
    &ROW-ID = string(rowid(cust_det))
    &TRIGGER-TYPE = 'DELETE'}.
```

When MFG/PRO is not being used, the trigger format should follow these examples.

Write example for non-MFG/PRO custom table `cust_det`:

```
TRIGGER PROCEDURE FOR REPLICATION-WRITE OF cust_det.
/* RECORD OUTBOUND EVENT. */
{qxoevent.i
    &TABLE-NAME = 'cust_det'
    &ROW-ID = string(rowid(cust_det))
    &SESSION-ID = <some source app session info>
    &DOMAIN-CODE = <domain, if applicable>
    &USER-ID = <some source app user info>
    &TRIGGER-TYPE = 'WRITE'}.
```

Replication delete example for `cust_det`:

```
TRIGGER PROCEDURE FOR REPLICATION-DELETE OF cust_det.
/* RECORD OUTBOUND EVENT. */
{qxoevent.i
    &TABLE-NAME = 'cust_det'
    &ROW-ID = string(rowid(cust_det))
    &SESSION-ID = <some source app session info>
    &DOMAIN-CODE = <domain, if applicable>
    &USER-ID = <some source app user info>
    &TRIGGER-TYPE = 'DELETE'}.
```

Important Make sure you use single quotes around `&TABLE-NAME` and `&DOMAIN-CODE` and also use them if `&SESSION-ID` and `&USER-ID` are blank.

The `&DOMAIN-CODE` value is intended for use with MFG/PRO eB2.1, which supports separate logical partitions within a database. If your application database can be split into categories similar to MFG/PRO domains, you may want to use this entry; otherwise, leave it blank.

See "Add Source Application Domains" on page 84.

If you do decide to use the domain value, you must define the domain to QXtend Outbound.

7 After adding schema triggers and creating the corresponding trigger code, you must compile the code using your application compiler. For MFG/PRO custom tables, you can use MFG/UTIL.

8 Add the trigger file directory to the application PROPATH so that the code can be executed.

Deleting Business Objects

To delete a business object you created, follow these steps. This process deletes the business object and any profiles that are based on it. You cannot delete QAD-supplied business objects.

1 Select the business object from the navigation tree. The business object displays. The Edit, Copy, and Delete buttons are active.

2 Click Delete to remove the profile. You are prompted to confirm.

3 Click OK to continue.

Modifying Business Object Data Objects

Once you have created or modified a business object, you can modify the related data objects.

1 In the object tree on the left, open the new business object and select a table within the tree. The data object details for that table display in the right-hand frame.

If you select the top-level table, no join information displays. If you select one of the lower-level tables as shown in Figure 6.8, the join information displays.

2 Click Edit in the Business Object frame at the top left. The editable fields in the data object become active.

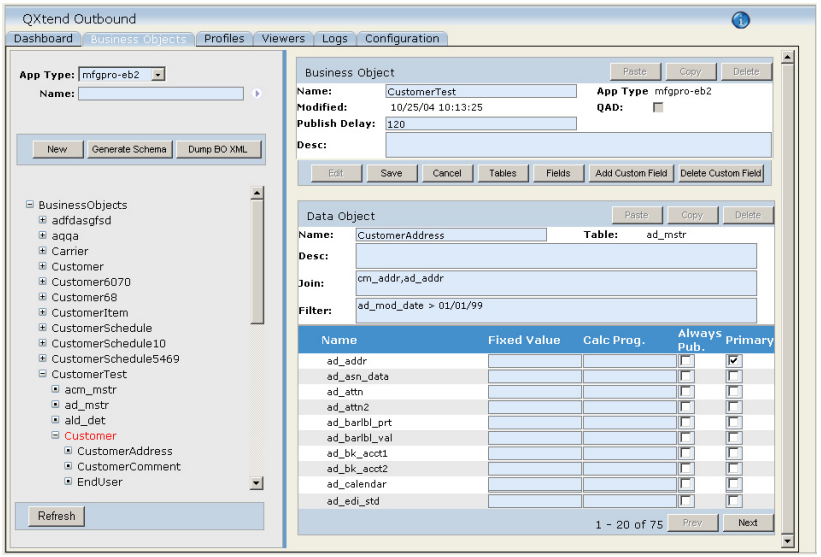


Fig. 6.8
Address Table Data Object

3 Enter values for the data object as follows:

Name. Enter a name (maximum 35 characters) for this data object. Data object names must be unique within a business object.

Table. The system displays the application table name.

Description. Enter a brief description (maximum 30 characters).

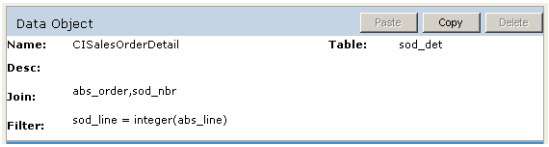
Join. Enter a comma-separated list of join field pairs, each of which is also comma-delimited. Each pair contains the parent table join field, then the child table join field.

The joins are used to navigate upward to the top of the business-object hierarchy when processing application events.

Filter. Optionally, enter filter criteria to limit the data stored after extraction from the application. The filter uses Progress 4GL expressions.

Figure 6.9 shows a relatively complex join and filter designed to join the abs_mstr and sod_det using the fields abs_order and sod_nbr, and then limiting or filtering the data extracted to those records where the sod_line value is equal to the abs_line value as converted to an integer.

Fig. 6.9
Complex Join and Filter



4 You can modify the behavior of individual fields in the table. Each field in the table displays in a table with active fields. Modify the values for each field as follows:

Fixed Value. Optionally, specify a fixed value that you always want to use in the QDoc for this field.

The system determines node values in this order: a calculated value, a fixed value, the value supplied in the event message.

Fixed values are typically used with custom fields, described in step 6.

Calc Program. Optionally, enter the name of a Progress program (`.p`) to be run on the QXtend AppServer for the associated source application. This program takes a character string that identifies a row in a database table and calculates and returns a value to be inserted in the QDoc node.

The system determines node values in this order: a calculated value, a fixed value, the value supplied in the event message.

Calculated values are typically used with custom fields.

Always Publish. Indicate if the field should be published whenever the data object that contains it is published, regardless of whether the field was changed by the application event.

Primary. Indicate if this field is part of the primary index for the table associated with this data object. For QAD-supplied business objects, the primary fields are already set.

5 Buttons in the Business Object frame allow additional configuration of the data object. Click Fields to display the Update Fields screen.

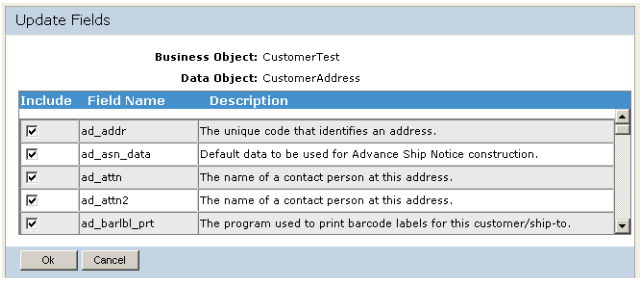


Fig. 6.10
Update Data Object Fields

By default all fields in a QAD-supplied business object are included. To exclude a field from the business object, clear the Include check box for the field.

6 Click Add Custom Field to add a new field to the data object.

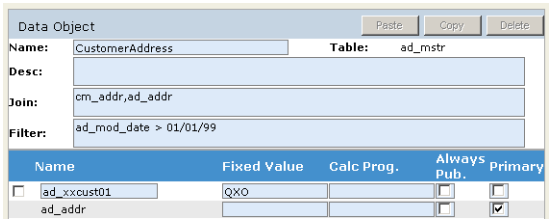


Fig. 6.11
Adding a Custom Field

Enter a field name, and add a fixed value or Progress calculation program if needed.

To delete a custom field, click the check box to the left of the field and click Delete Custom Field.

7 Click Save to save the data object and any changes you have made.

Copying or Deleting Data Objects

You can copy a data object from one business object and paste it into another. This eliminates configuration steps where multiple business objects such as sales orders, invoice history, and customer all share a data object such as addresses. You can also delete data objects if they were copied into the wrong business object.

1. While the source business object is in edit mode, select the source data object in the left-hand navigation tree.
2. In the Data Object frame, click Copy.
3. Use the navigation tree to open the target business object, and data object, as necessary.
4. In the target business object Data Object frame, click Paste. The copied data object is pasted in.
5. To delete a data object from a business object, open the data object using the navigation tree. Click Delete in the Data Object frame.

Managing Business Object XML Files

Two additional business object functions let you generate an XML schema from the business object in order to validate the XML structure, and dump the business object and associated profiles in an XML format that can be loaded into another instance of QXtend Outbound.

Navigate to the business object you want to work with. In the navigation tree frame click:

Generate Schema. Generate the XML schema from the business object in order to validate its structure.

See “Importing XML” on page 99.

Dump BO XML. Create a file with the XML representation of the business object and its associated profiles. These files can be loaded into another instance of QXtend Outbound using the XML Load utility on the Configuration tab.

The dump file is created with the following naming convention:

QXtendInstallDir\boXML*AppType**BOName**ProfileName*

Chapter 7

Managing Profiles

The following material covers implementation of QXtend Outbound profiles.

Introduction

Profiles are views of business objects tailored for the requirements of specific subscribers.

To select which components of a business object are sent to which subscribers, QXtend Outbound lets you define profiles. The way you define a profile is nearly identical to the way you define a business object.

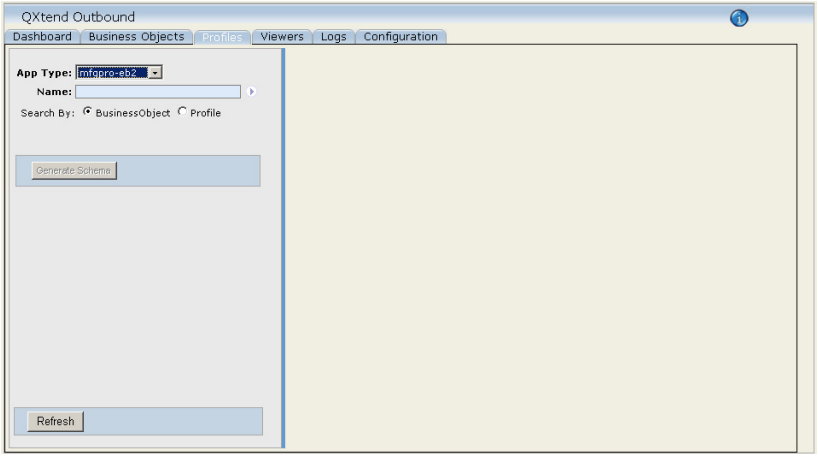
Business objects and profiles are closely linked. When you create or copy a business object, a default profile with the same name is automatically created for it. When you add or delete tables or fields from a business object, the change is automatically reflected in the associated profiles as well. If you change the name of a business object, a new default profile with the same name as the new business object name is created for it. Any existing profiles associated with the business object remain unchanged.

Viewing Profiles

Profiles are the means you use to select which components of a business object are sent to subscribers. The method of defining a profile is nearly identical to the method of defining a business object; however, unlike business objects, profiles always start as a copy of a default profile.

When you click the Profiles tab, the screen illustrated in Figure 7.1 displays.

Fig. 7.1
Profile Screen



App Type. Select the source application type from the drop-down list. Source applications are defined in the Configuration tab. The last selected value is saved and used as the default entry the next time this screen is viewed.

See "Defining Source Applications" on page 80.

Name. Use the Name field to find profiles to view or update. Click Business Object to search by business object names; click Profile to search by profile names.

See "Filtering Names" on page 77.

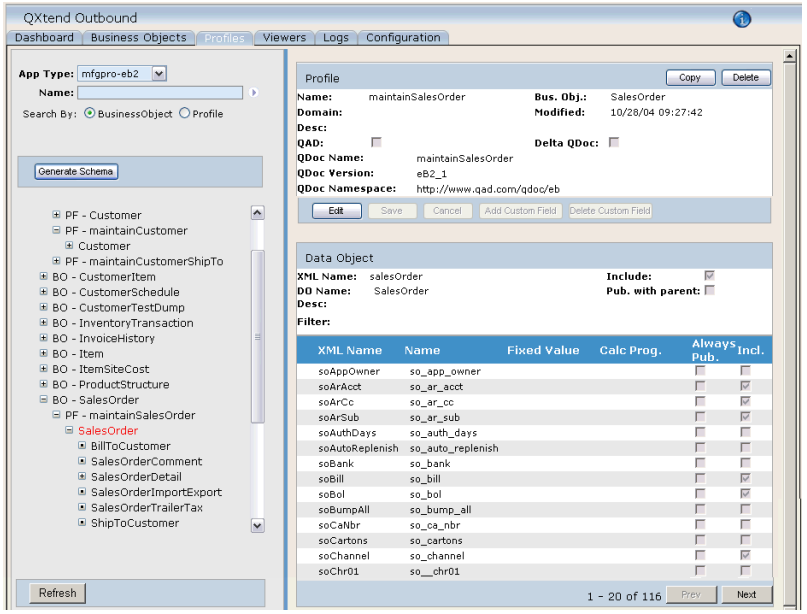
Click the arrow next to Name to display all available business objects with the Name field left blank, or to search within the available business objects for objects that match your Name entry.

Modifying Profiles

Since you cannot edit the QAD-supplied profiles or create a new one, you must copy an existing profile and modify it. To create a copy and modify a profile, follow these steps.

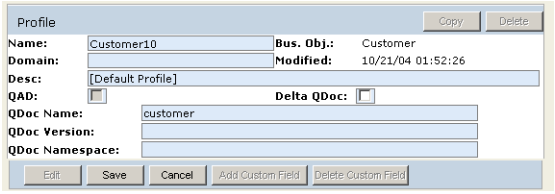
1. Select the source profile from the navigation tree. If it is a QAD-supplied object, the only button active is Copy.
2. Open the nodes in the profile in the navigation tree to view the data objects in the profile.

Fig. 7.2
Profile Copy



3 Click Copy. The new profile is created including all related data objects. The new profile definition screen displays. A new name is provided for the profile by default.

Fig. 7.3
Profile Definition



4 Enter values for the new profile as follows:

Name. Update the name as required.

Business Object. The business object this profile is based on displays in this field.

Domain. For MFG/PRO eB2.1 databases, specify the domain associated with the profile. If you leave this field blank in MFG/PRO eB2.1, the profile applies to all domains in the database. If a domain value is specified, the profile only sends QDocs for events extracted from the specified domain.

For other application types, you can use domain according to your own business requirements or leave it blank.

Modified. No edit allowed. This displays the date and time this business object was last saved.

Desc. Enter a description of the object. Consider including the creation date, object author, and purpose.

QAD. Indicates a QAD-standard profile.

Delta QDoc. Indicates whether this profile will send delta QDocs or complete QDocs. A delta QDoc excludes non-primary elements of the business object that have not changed since the last update. If a field is tagged Always Publish in the business object, it overrides the delta setting here.

QDoc Version. Use this to distinguish between MFG/PRO QDoc versions. This is the version designation that appears in the QDoc headers and is one of the values required to validate the QDoc. The values for the supported MFG/PRO releases are:

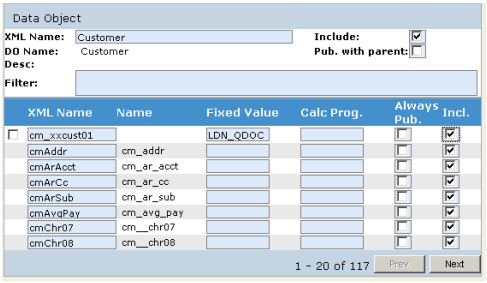
- eB
- eB2
- eB2_1

QDoc Namespace. Specify a URL for QDocs generated using this profile. Like the QDoc version, the QDoc namespace is a required value in the QDoc headers to validate the source of the QDoc. The following sample of a QDoc header shows the location and content of the QDoc namespace and QDoc version in bold type.

```
<maintainSalesOrder xmlns="http://www.qad.com/qdoc/eb2_1"
xmlns:enc="http://www.w3.org/2002/12/soap-encoding" xmlns:qcom=
"http://www.qad.com/qdoc/common" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance" xsi:schemaLocation="http://www.qad.com/
qdoc/eb ..\..\..\schemas\eB2\maintainSalesOrder-eB2_1.xsd"
version="eB2_1" scopeTransaction="false" suppressResponseDetail=
"false" xml:lang="en">
    <salesOrder>
        <operation>A</operation>
        <soNbr>bcl80</soNbr>
```

5 Click Add Custom Field to add a new field to the data object.

Fig. 7.4
Adding a Custom Profile Field



Enter a field name, and add a fixed value or Progress calculation program if needed. The field name must contain only alphanumeric characters; the name is validated to ensure no illegal characters are included and that the field name is a valid XML node.

To delete a custom field, click in the selection box to the left of the field and click Delete Custom Field.

6 Click Save to save the new profile and any changes you have made.

Deleting Profiles

To delete a profile you have copied and modified, follow these steps. This process deletes the profile and the data objects stored within it. You cannot delete the QAD profiles or data objects.

If you delete all user-configured profiles associated with a business object, a new default profile is generated. This occurs because a business object cannot exist without a profile. In this scenario, the delete essentially refreshes the default profile for the business object.

In addition, if a QDoc does not have an associated profile because the profile has been deleted, the profile name displays as UNAVAILABLE in the various reports and views.

1 Select the profile from the navigation tree. The profile displays. The Edit, Copy, and Delete buttons are active.

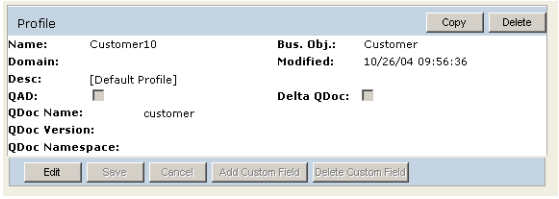


Fig. 7.5
Profile Header

2 Click Delete to remove the profile. You are prompted to confirm.

3 Click OK to continue.

Modifying Profile Data Objects

Once you have created a copy of a profile and saved it, you can modify the data objects within the profile.

1 Select a non-QAD profile in the navigation tree and open the data object you want to modify or delete.

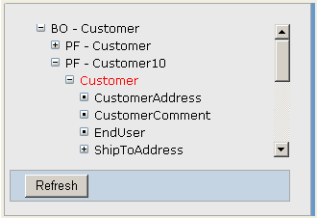
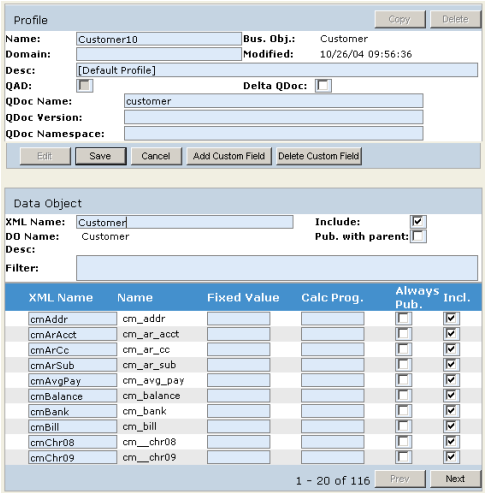


Fig. 7.6
Profile Data Object Nodes

2 The data object displays in the right pane. Click Edit to access the profile and data object fields.

Fig. 7.7
Profile and Data Object Definition



3 Use the descriptions in step 4 in “Modifying Profiles” on page 125 to modify the profile settings. Use the following field descriptions to modify the data object.

XML Name. The name of the XML element or node for the data object in published QDocs.

Include. Indicate if you want this data object included in QDocs generated based on this profile. This setting applies to all of the fields in the object. When you include the object, you can also determine on a field-by-field basis which fields to include.

DO Name. The system displays the name given this data object in the business object used as a template.

Publish with Parent. Indicate if you want the fields in this data object to be published automatically whenever data in the business object’s parent is published. For example, when a field associated with a sold-to customer changes, you may always want to publish the sold-to customer address.

The hierarchical display in the navigation tree shows the parent/child relationship between data objects.

Description. Enter a brief description.

Filter. Optionally, enter filter criteria to limit the data extracted from the application. The filter uses Progress 4GL expressions.

4 Below the profile data object, you can modify settings related to each field in the profile.

XML Name. Enter the XML element node name used to publish this field in outbound messages.

Fixed Value. Optionally, specify a fixed value that you always want to use in the QDoc for this field.

The system determines node values in this order: a calculated value, a fixed value, the value supplied in the event message.

Fixed values are typically used with custom fields.

Calc Program. Optionally, enter the name of a Progress program (`.p`) to be run on the QXtend AppServer for the associated source application. This program takes a character string that identifies a row in a database table and calculates and returns a value to be inserted in the QDoc node.

The system determines node values in this order: a calculated value, a fixed value, the value supplied in the event message. Calculated values are typically used with custom fields.

Always Publish. Indicate if the field should be published whenever the data object that contains it is published, regardless of whether the field was changed by the application event.

Include. Indicate if you want this field included in messages generated based on this profile.

5 Click Save to save the data object changes.

Managing Profile XML Files

An additional profile function lets you generate an XML schema from the profile in order to validate the XML structure.

1. In the navigation tree, select the profile you want to work with.
2. Click Generate Schema. This generates the XML schema from the profile.

Chapter 8

Using the QXO Dashboard

The following material covers the QXtend Outbound dashboard.

Using the QXO Dashboard

The Dashboard displays by default with a system overview. There are two basic views—Overview and Services—displayed in the navigation tree. Click Overview to view the output from your defined source applications. Click Services to view details for event services, message publishers, message senders, and subscribers.

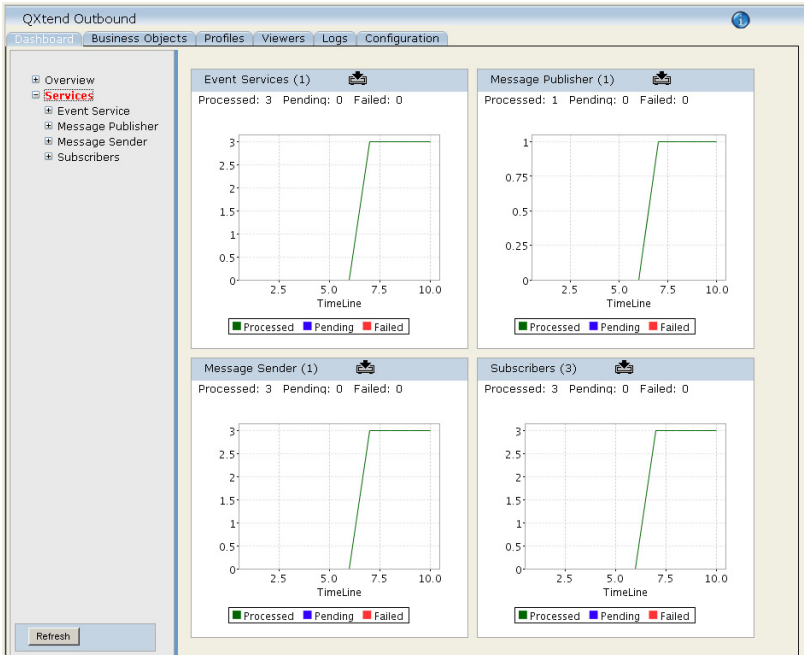


Fig. 8.1 Services Dashboard Showing the Addition of a Customer in a Source Application

Note The input date format on the dashboard UI is American for the current release, and is not configurable. Database date formats can be either dmy or mdy, but both the `qxevents` and the `qxodb` databases must have the same format. Note also that time data being returned from the database, such as Date-Time fields on the Logs screen or modification time fields on the profiles, are in the format of the `qxodb` database.

Use the navigation tree on the left to drill down to further details. The charts that display for each source application, service, and subscriber view use a standard timeline format showing the previous hour of activity

broken into ten six-minute periods. Adjust the view by resetting the start and finish times in the individual views. The minimum total period is 10 minutes.

Click the Download icon in the title bar of a chart to view the data represented in the chart in an HTML table. You can then save this data for import to a spreadsheet application such as Excel and create your own graphs.

Fig. 8.2
Download Icon

Note in Figure 8.1 the numbers at the top of each chart. One event service, one message publisher, and one message sender are running. Three subscribers are active. The event service processed three events resulting from adding a customer record. The message publisher processed these events as a single message. The message sender then created three QDocs to meet the needs of the three different subscribers.

Viewing Services Summaries

When you select one of the top-level services nodes—Event Service, Message Publisher, and so on—a summary of all the services of that type displays. In the table at the top of the screen, each instance of the service and its status are displayed. The possible statuses are:

STRT. The instance is starting. Blue.

STOP. The instance is stopped. You can restart the instance using the Start button at the bottom of the screen. Red.

IDLE. The instance is idle. Green.

PROC. The instance is processing an event or a message. Yellow.

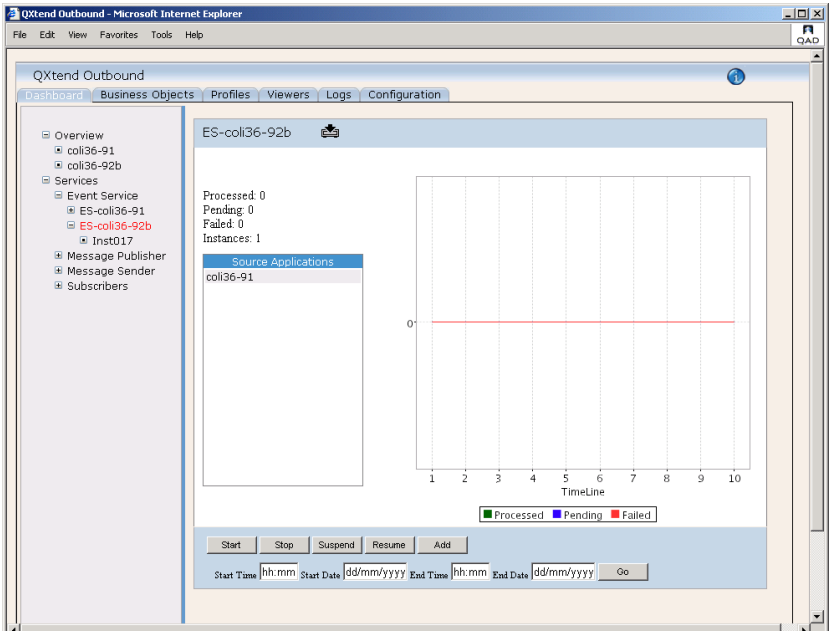
SUSP. The instance is suspended. You can resume the instance using the Resume button at the bottom of the screen. Light gray.

DEAD. The instance is dead. You can replace the instance by creating a new one. Dark gray.

Viewing Specific Services

Figure 8.3 shows how the Dashboard looks after you have drilled down into the event services view. The views for message publishers, senders, and subscribers are identical, except that subscribers do not require the buttons displayed in the other screens.

Fig. 8.3
Event Services Dashboard



The status of the individual service instance displays as a round, colored icon next to the instance name. See “Viewing Services Summaries” on page 135 for a summary of the color codes.

Fig. 8.4
Status Color Code for Services Instance

You can use the buttons on the bottom of the screen to perform the following actions:

Start. Start all services if not already started.

Stop. Stop all services if not already stopped.

Suspend. Suspend all services if not already suspended.

Resume. Resume all services that have been suspended.

Add. Add a new instance of the service based on this session profile.

Use the start and end time fields to limit the data displayed in the chart.

Monitoring Source Applications

Under the Overview node, you can monitor your source applications. Each source application is listed under the Overview node. Click the Overview link to see a summary of all source application activity.

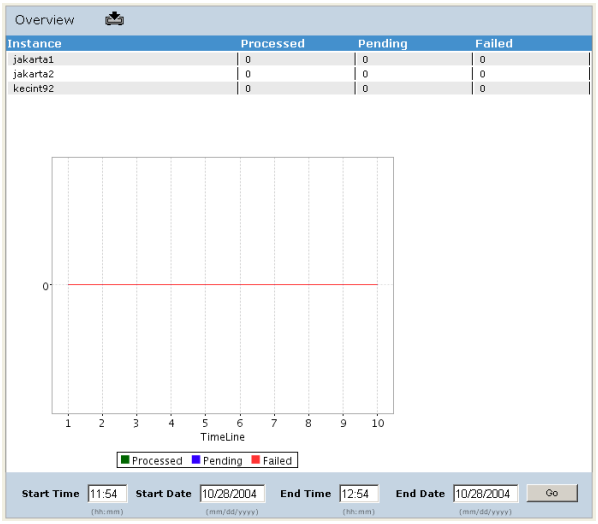
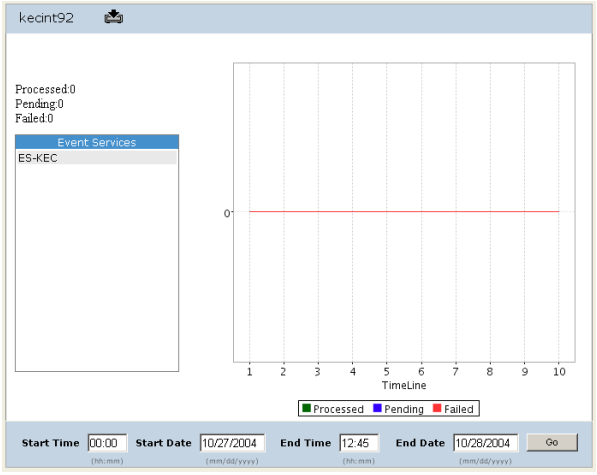


Fig. 8.5
Source Application Overview

The table at the top shows the individual source applications. Instance is the source application name. Processed, Pending, and Failed shows the number of events in each stage. The chart shows the cumulative processing for all source applications.

Open the Overview node in the navigation tree and click any one of the source applications to view specifics.

Fig. 8.6
Individual Source Application View



The event services associated with the source application appear in the table to the left.

Chapter 9

Using QXO Viewers

The following material covers the QXtend Outbound viewers.

Using Viewers

Use the Viewers tab to view:

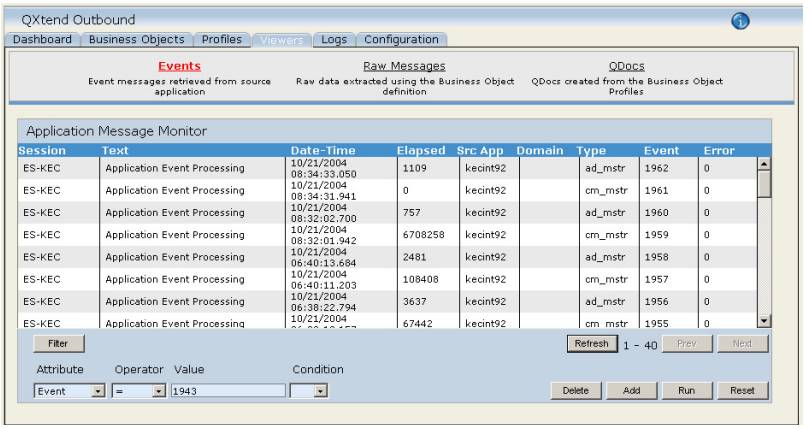
- Messages generated when QXO successfully reads an event from the `qxevents` database
- Raw data extracted from a source application for all profiles that are monitoring that data
- QDocs created from the business object profiles

See "Using Table Filters" on page 78 for details on using filters.

Information displays in filterable tables. Each view displays the 50 most recent records. Next and Previous buttons move up and down through the records. If alerts occur, you can display and filter them also.

Application Event Viewer

Fig. 9.1 Application Event Viewer



When application data being monitored by QXtend Outbound is changed, a record is written to the `qxevents` database and queued for processing by the event service. The event service polls the `qxevents` database for new events and when it finds one, it extracts the related application data from MFG/PRO and saves it in the `qxodb` database in the form of raw data.

A message is recorded each time an event is processed. The message indicates the time the event was processed, any errors that occurred, the source application that caused the event, and other pertinent information.

Filtering View Output

To filter the output of a view, use the following steps:

See also “Using Table Filters” on page 78.

1 Under the Viewers tab, select the view you want to filter from the menu along the top. The full view displays.

2 Click Filter to open the filter panel.

3 In the filter panel, enter a filter Attribute—session, type, date-time, and so forth as displayed across the top of the viewer—an Operator, and the Value you want to filter by.

There are two different sets of operators possible, one for numeric values, the other for text as shown in Figure 9.2.

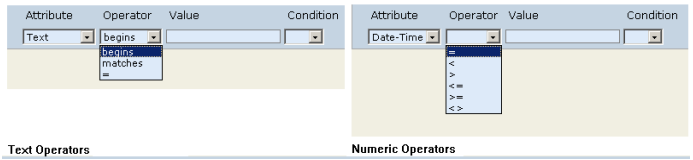


Fig. 9.2
Text Operators

4 If you want to search by multiple values, enter a Condition and click Add. The next filter line displays.

5 Click Run to filter the viewer.

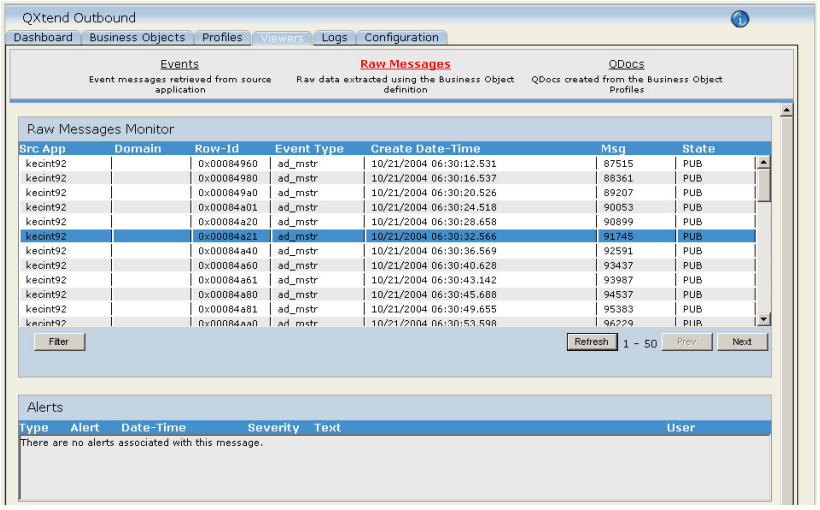
Note If you change the filter values, then click Next or Previous before clicking Run, your filter changes are lost.

6 Click Delete or Reset to close the filter pane.

Raw Message Viewer

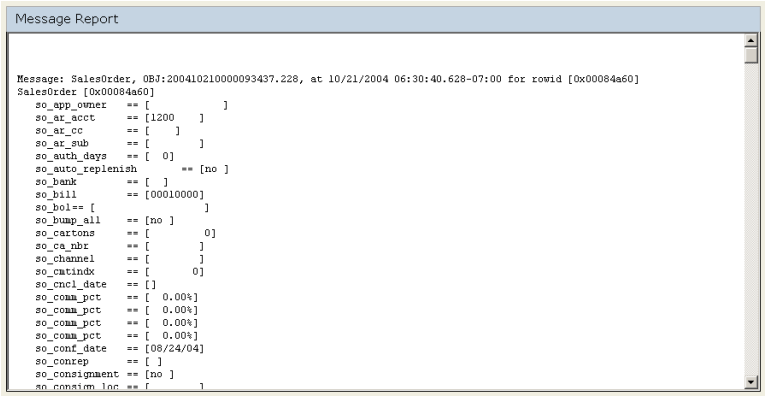
The raw message viewer displays information about the messages stored in the `qxodb` database by the event services. Click any message to view associated alerts and the message content.

Fig. 9.3
Raw Message Viewer



Scroll down to view the raw message content.

Fig. 9.4
Raw Message Content



QDoc Viewer

You can view the XML documents with the QDoc Viewer illustrated in Figure 9.5.

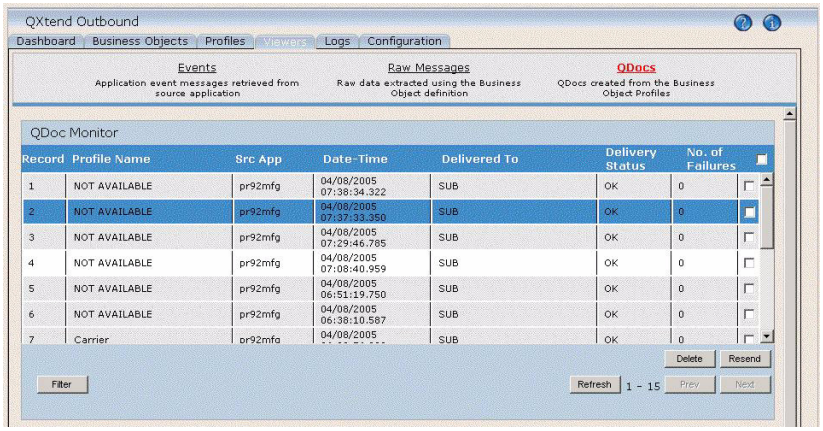


Fig. 9.5
QDoc Viewer

Delete. Use the Delete button to remove a QDoc from the viewer. This also deletes the QDoc record in the `qxodb` database.

Resend. Use Resend to forward a QDoc to any subscribers that are registered for the QDoc profile. A resend changes the status to PEND. After delivery, the status changes to ERR or OK depending on the success of the delivery.

Scroll down to view the QDoc.

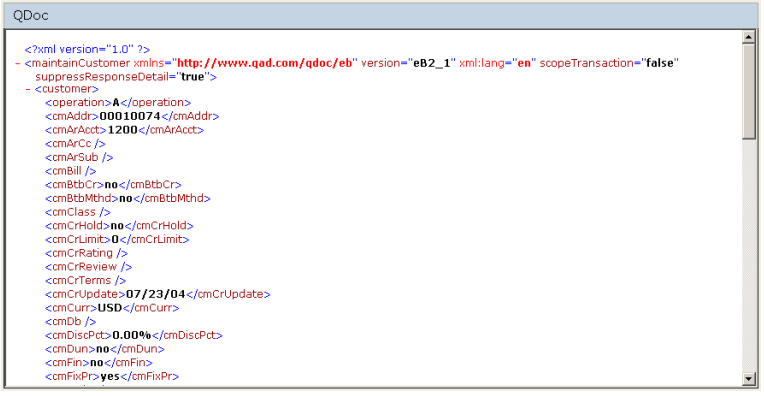


Fig. 9.6
QDoc Content

Chapter 10

Using the QXO Log Monitor

The following material covers the QXtend Outbound logs.

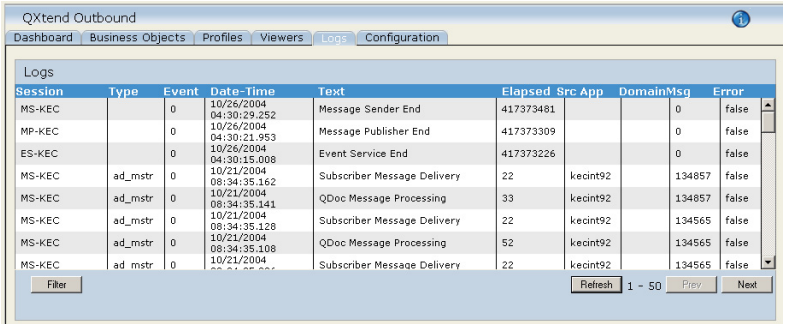
Viewing QXO Logs and Exceptions

Use the Log tab to view log and alert messages for all transactions in the QXtend Outbound system and filter them by various criteria.

Messages are written to the log by all of the QXO services when data is extracted and when QDocs are created and published.

Figure 10.1 illustrates the Logs portion of this screen.

Fig. 10.1
Log Tab



The system returns 50 records. If more than 50 records exist, use the next and previous buttons to display the next group of 50 records. Use the scroll bar to scroll up and down within a group of 50 records. Click Refresh to find new records created since the initial screen display.

See “Filtering View Output” on page 141.

Use the Filter button to toggle the display of the filter criteria.

Note In the current release, logs and alerts remain in `qxodb` indefinitely and cannot be cleared without accessing the database directly. A method to clear these logs and alerts will be provided in a later release.

Additional Logs

For further troubleshooting, review the logs located in *`QXOsrvInstallDir`*`\wrk\log`. The log naming convention for the various services is `session-`*`<numeric-instance-segment>`*`.log`. For example, a log file for an event service instance named Inst001 will be named `session-001.log`.

There are no subscriber logs. Troubleshooting subscriber problems should be done on the subscriber side.

To control log detail, you can edit your start-sess.sh or start-sess.bat file. Open the file in a text editor, locate the LOGLEVEL entry, and change the logging level as desired:

0 Only level 0 messages and one required level 1 message displays.

1 Level 0 and 1 messages display.

2 Level 0, 1, and 2 messages display.

By default this value is set to 0. This is the recommended setting. If you are having trouble debugging an issue, you can change the logging level to display additional errors and messages.

The AppServer broker and server logs, if set up as described during installation, will be named `QXOS.broker.log` and `QXOS.server.log`, respectively.

See “Creating the AppServer” on page 38.

Glossary

4GL. An abbreviation for fourth-generation programming language, such as the Progress language. Direct APIs to the Progress code are more efficient than calls to the user interface.

API. Application program interface. A set of routines, protocols, and tools that connect applications, usually for the purpose of sharing data. The QXtend interoperability framework removes the need for specially-coded APIs.

B2B. Business-to-business; the exchange of products, services, or information between businesses rather than between businesses and consumers.

Business Object. A set of tables and fields defined within QXtend Outbound, and used to extract data from a source application.

CRUD. Create-read-update-delete. Used to describe a software application that can perform those actions on database records.

Event. A change to data in a targeted source application that is of interest to one or more subscribers to QXtend Outbound. A notification of each relevant change is written to the `qxevents` database where QXtend Outbound polls for it.

Event Service. A process that runs on the QXtend Outbound server that polls the `qxevents` database for change notifications. When one is encountered, the event service queries the source application database for the changed data and writes the data as a raw message to the `qxodb` database.

Extensible Markup Language (XML). A specification designed especially for Web documents that allows the definition, transmission, validation, and interpretation of data between applications and organizations.

HTTP (Hypertext Transfer Protocol). The set of rules for exchanging text, graphic images, sound, video, and other multimedia files on the World Wide Web.

Java. An object-oriented programming language created by Sun Microsystems. Java is a device-independent language. Programs compiled in Java can be run on any computer. Java programs can be run as free-standing applications or as applets placed on a Web page.

Java Runtime Environment (JRE). A subset of the Java Development Kit for end users and developers who want to redistribute the Java runtime environment. The Java runtime environment consists of the Java virtual machine (JVM), the Java core classes, and supporting files.

Message Publisher. A process that runs on the QXtend Outbound server that polls the `qxodb` database for raw messages. When one is encountered, the message publisher

republishes the message in QDoc format. It uses the profiles to create the QDocs required by each interested subscriber. The QDocs are written back to the `qxodb` database.

Message Sender. A process that runs on the QXtend Outbound server that polls the `qxodb` database for QDocs. When one is encountered, the message sender sends it to each interested subscriber.

Namespace. In XML, a unique identifier for a collection of element type and attribute names. In an XML document, any element type or attribute name can have a two-part name consisting of the name of its namespace and then its local—or functional—name.

Profile. A subscriber-specific definition of a subset of a business object. Default profiles from QAD are loaded into the system and can be copied and modified to customize them.

PROPATH. An environment variable containing the list of directories Progress searches when looking for a program to execute.

QDoc. An inbound (to MFG/PRO) data document in XML format that conforms to generated schemas and events files from the QGen utility.

QDoc Schema. The QDoc schema defines the structure and data types of the XML. The schema is the building block for any API request.

qxevents Database. A database added to the MFG/PRO source application where change notifications are written after a data update in the source application.

qxodb Database. The QXtend Outbound database, installed on the QXtend Outbound server. This database contains raw messages, QDocs, and log records.

QXtend Inbound. The companion to QXtend Outbound for QAD applications, this product accepts QDocs from external sources and enters them into an MFG/PRO, or other QAD product, database.

Simple Object Access Protocol (SOAP). A protocol for exchanging information in a decentralized, distributed environment in XML format.

SOAP. See *Simple Object Access Protocol (SOAP)*.

Source Application. Any Progress-based production system set up with triggers and a `qxevents` database to notify QXtend Outbound of a relevant change.

Subscriber. A destination for QDocs defined in QXtend Outbound. The destination can be either a Web service URL or a directory location.

Tomcat. The servlet container used in the official reference implementation for the Java Servlet and Java Server Pages (JSP) technologies. Tomcat is developed in an open and participatory environment and released under the Apache Software Foundation license.

Triggers. A program defined in the database of a source application that automatically records a change to specific tables whenever a change occurs.

Uniform Resource Identifier (URI). A method of identifying or reserving a point of content on the internet, such as a page of text, a graphic image file, or a program. A URI typically describes:

- The mechanism used to access the resource
- The specific computer that the resource is housed in
- The specific name of the resource (a file name) on the computer

The most common form of URI is a uniform resource locator (URL).

Uniform Resource Locator (URL). A text string that indicates the location of an intranet or Internet resource.

Universal Unique Identifier (UUID). A hexadecimal number including a time stamp and a host identifier. Applications use uuids to identify many kinds of entities.

User Interface (UI). The portion of an application that is visible to the user and the mechanism by which the end user interacts with the application, enters information into the application, and sees the results of the interaction.

UUID. See *Universal Unique Identifier (UUID)*.

WAR. See *Web Archive File (WAR)*.

Web Archive File (WAR). A compressed file containing a Web application and its related files. Assists in easily deploying an entire application.

Web Services. Vendor-neutral, XML-based, remote procedure call (RPC) protocol that allows any system to run programs in other, dissimilar systems.

XML. See *Extensible Markup Language (XML)*.

XML Schema Definition (XSD). An abstract representation of the elements in an XML document that can be used to verify that each item of content adheres to the associated element's description. The XSD standard follows the W3C recommendation.

Index