QAD Enterprise Applications 2009
Enterprise Edition

User Guide

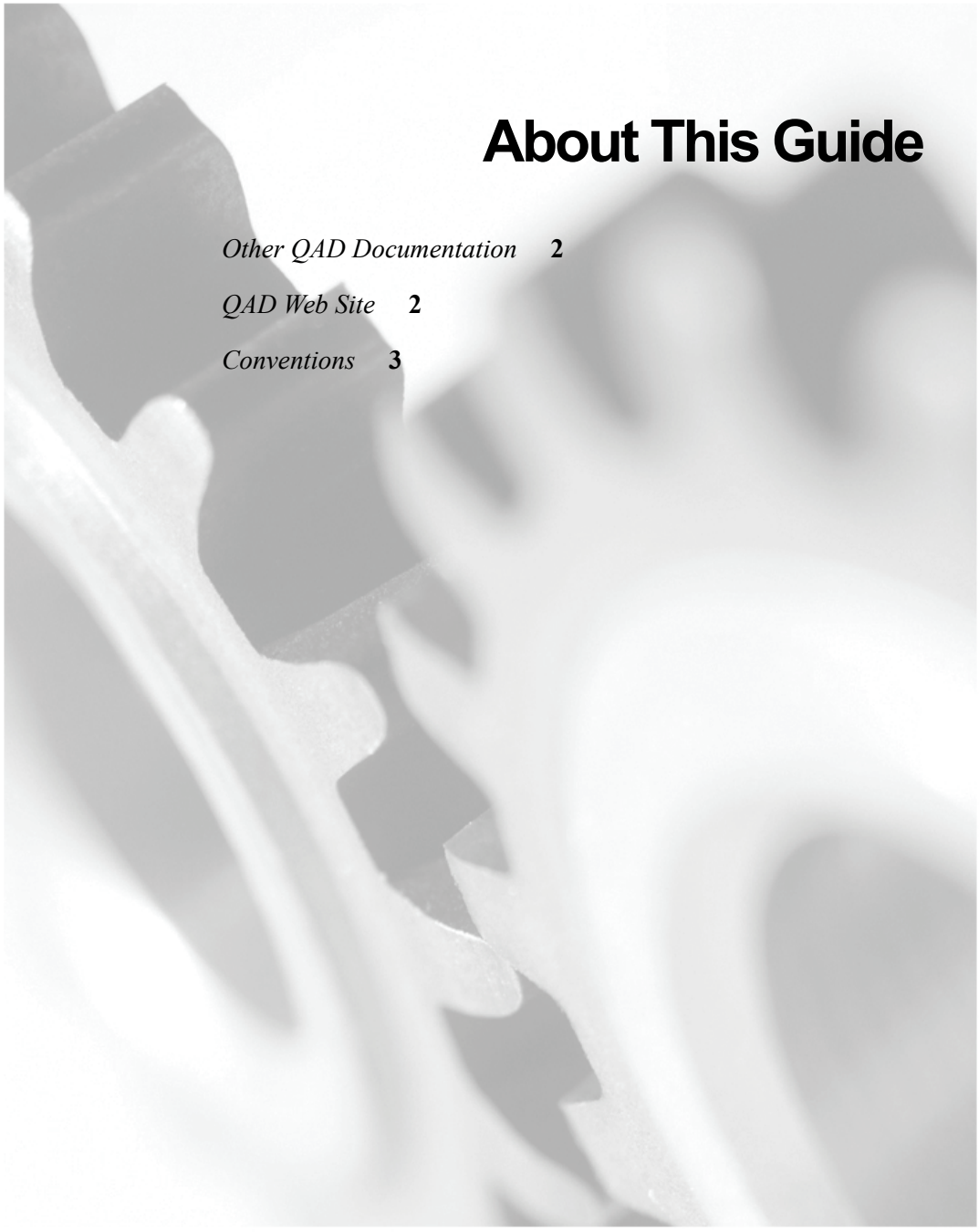
QAD System Administration

Domain Constants
System Interface
Printers
Batch Processing and Daemons
System Control
CIM Interface
Database Management
Reports and Utilities
Customizing Business Logic

78-0744C
QAD 2009 Enterprise Edition
April 2009

QAD Inc.
100 Innovation Place
Santa Barbara, California 93108
Phone (805) 684-6614
Fax (805) 684-1890
http://www.qad.com

Contents



About This Guide

This guide covers the system administration programs within QAD Enterprise Applications 2008. Most of these programs are on the System Administration menu (36).

Other QAD Documentation

- For an overview of new features and software updates, see the *Release Bulletin*.
- For software installation instructions, refer to the appropriate installation guide for your system.
- For conversion information, refer to the *Conversion Guide*.
- For an overview of system features and instructions on navigating the user interface, see *User Guide: QAD User Interfaces*.
- For detailed information on using system features, refer to the relevant user guide.
- For technical details, refer to *Entity Diagrams* and *Database Definitions*.

For a complete list of QAD Documentation, visit the QAD Online Support Center at:

http://support.qad.com/

QAD Web Site

The QAD Web site provides a wide variety of information about the company and its products. You can access the Web site at:

http://www.qad.com

For users with a QAD Web account, product documentation is available for viewing or downloading from the QAD Online Support Center at:

http://support.qad.com/

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

Most user documentation is available in two formats:

- Portable document format (PDF) files can be downloaded from the QAD Web site to your computer. You can view and print them with the free Adobe Acrobat Reader.
- HTML files let you view user documentation through your Web browser and use search tools for easily locating topics of interest.

Conventions

Screen Illustrations

System functions are available in a .NET-based graphical user interface (UI); a subset of functions can be used in a simplified character interface. All screenshots in the documentation show the .NET UI.

Navigation in the two UIs is not the same. The user guide text follows the navigation model of the .NET UI for moving from one screen to the next. In the character interface, the Progress status line at the bottom of a program window lists the main UI-specific keyboard commands used in that program. In the .NET UI, alternate commands are listed in the Actions menu.

For complete keyboard command summaries for UI navigation, refer to the appropriate chapters of *User Guide: QAD User Interfaces*.

Typographic

This document uses the text or typographic conventions listed in the following table.

If you see:	It means:
`monospaced text`	A command or file name.
`italicized monospaced text`	A variable name for a value you enter as part of an operating system command; for example, *`YourCDROMDir`*.
`indented command line`	A long command that you enter as one line, although it appears in the text as two lines.

If you see:	It means:
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.

Chapter 1

Introduction to System Administration

Introduction to System Administration

As part of any initial implementation, you must perform a number of setup tasks, including the following:

- Set up system-wide data such as generalized codes, printers, batch queues, menus, messages, and language code.
- Define users and security. These activities are described in *User Guide: QAD Security and Controls.*
- Set up the basic corporate structure that forms the basis for all business activities. This setup includes shared address data; domains, which are sets of related business entities that share a common base currency and chart of accounts; entities; and other shared data. This setup is defined in *User Guide: QAD Financials A.*

  Note Shared Services Domain is a separately licensed module. Unless you purchase appropriate licenses, the system prevents you from having more than one active domain per database.
- Define base data such as items, sites, and locations. This activity is described in *User Guide: QAD Master Data*

This book covers the first topic, setting up system-wide data, as well as common administrative tasks that are required in a running system. It is divided into two sections, reflecting these distinct activities:

- Database Setup, which includes domain constants, system interface, printers, batch processing and daemons, and configuring database settings.
- Database Administration, which includes the CIM interface, database management, and reports and utilities.

The System Administration menu includes tasks typically performed by system administrators. Most functions located on this menu (36) are discussed in this volume. However, a few areas are discussed in other volumes:

- Corporate Structure Setup (36.1) is described in *User Guide: QAD Financials A*.
- Operational Accounting Controls (36.9) are described in the various user guides for the functional areas they affect.

- System Security (36.3) and Enhanced Controls (36.12) are described in *User Guide: QAD Security and Controls.*
- Configured Messaging (36.4.6.13) applies only to scheduled orders and is discussed in *User Guide: QAD Scheduled Order Management.*
- External Interfaces (36.5) and Q/LinQ (36.8) are discussed in various technical references.

This volume does not cover the various utilities on the System Administration menus numbered above 24. For documentation of these programs, see the procedure help or the opening program screen of each utility.

Types of Functions

The system includes both standard Progress programs and component-based functions. The component-based technology extends features of the user interface. These activities can be executed only from the .NET UI.

Because of differences in the underlying technology of the two types of programs, some administrative functions apply to one or the other, and some apply to both. For example, you set up menu locations for standard programs and component-based functions using Menu System Maintenance. However, batch setup applies only to standard programs and daemon setup applies only to component-based functions.

The type of program affected by an area of system administration is pointed out in the discussion of each system administration feature.

Domain Constants

The programs on the Domain Constants menu (36.2) control calendars and codes used throughout the system. These include shop and holiday calendars, reason and generalized codes, and rounding methods.

See “Domain Constants” on page 13 for details.

In addition, you can set up number sequences using number range management (NRM) functions, which support regulatory controlled document numbering. NRM controls the content and sequencing of a numeric series, as well as preventing gaps in a series.

Finally, you can specify fields in tables for detailed change tracking and reporting.

System Interface

See “System Interface” on page 39 for details.

The System Interface menu contains programs that control menus, screen labels, messages, multi-language installations, and help. You can set up user function keys, define your e-mail system, and specify login scripts.

System interface functions include programs for creating browses and associating them with fields and programs, and managing stored browse data. In addition, you can also define alternate programs to execute when menu items are selected and specify programs to be run from other programs.

To meet specialized needs, you can set up user-defined fields for both standard programs and component-based functions.

Printers

See “Printers” on page 101 for details.

The Printer Management menu contains programs for setting up system printers, specifying default printers for a single user or all users, and creating batch print requests.

Batch Processing and Daemons

See “Batch Processing and Daemons” on page 109 for details.

Use programs from the Batch Processing/Daemon menu to set up background jobs and manage and monitor daemon processes, which run in the background and perform required utility tasks when the system is being used.

System Control

See “System Control” on page 159.

The System Control menu contains critical control programs that must be set up before the system is used, that affect all users, domains, and entities.

CIM Interface

CIM (computer integrated manufacturing) is one way to load legacy or non-Progress data into the QAD database. Using CIM, data can be added using standard program validation.

See "CIM Interface" on page 179 for details.

Database Management

The system provides utilities for monitoring database size, performing dumps and loads, reloading archive files, and managing database sequences. Delete/archive followed by dump/load is the standard means of controlling database size and fragmentation in Progress databases.

See "Database Management" on page 193 for details.

User licensing utilities and programs for managing time zones are also included in database management.

Reports and Utilities

A number of system-wide reports and utilities are provided on the System Administration menu.

See "Reports and Utilities" on page 227 for details.

The system cross-reference programs display information about field, program, and table relationships in your database. If you customize your implementation, this is an essential set of programs.

The system can use a Progress application server (AppServer) to run applications remotely. The AppServer must be defined first to make it available.

Customizing Component-Based Functions

In addition to using such tools as Design Mode and user-defined fields to modify the way component-based QAD applications work, you also can customize business component code for an installed application using a standard Progress 4GL editor.

See "Customizing Component-Based Functions" on page 9.

Section 1

Database Setup

This section includes information on system setup activities.

Chapter 2

Domain Constants

The programs on the Domain Constants menu control calendars and codes used by various functions within a domain.

Overview

Domain constants provide basic data used by many system functions. All codes defined by the functions listed in Table 2.1 are domain specific. Since a domain represents a distinct business operation, codes can be quite different between domains. If you need to use the same code in more than one domain, you must set it up for each domain that requires it.

Table 2.1
Domain Constants Menu (36.2)

Number	Menu Label	Program
36.2.1	Holiday Maintenance	mghdmt.p
36.2.2	Holiday Browse	mgbr017.p
36.2.5	Calendar Maintenance	mgscmt.p
36.2.6	Calendar Inquiry	mgsciq.p
36.2.13	Generalized Codes Maintenance	mgcodemt.p
36.2.14	Generalized Codes Browse	mgbr004.p
36.2.15	Generalized Codes Validation Rpt	mggencrp.p
36.2.17	Reason Codes Maintenance	mgrnmt.p
36.2.18	Reason Codes Browse	mgbr007.p
36.2.19	Reason Codes Report	mgrnrp.p
36.2.21	Number Ranges Menu ...	
36.2.21.1	Number Range Maintenance	nrsqmt.p
36.2.21.2	Sequence Browse	nrbr001.p
36.2.21.5	Sequence Number Maintenance	nrnxmt.p
36.2.21.13	Sequence Number History Report	nrsqrp.p
36.2.21.23	Sequence Delete/Archive	nrsqup.p
36.2.22	Change Tracking Maintenance	mgtblcmt.p
36.2.23	Change Tracking Browse	mgbr223.p

Maintaining Holiday and Shop Calendars

The shop calendar is required for planning, manufacturing, and distribution modules. The calendar indicates what days the plant is open and how many hours are worked each day. This information is used:

- To schedule start and due dates for MRP planned orders, master schedule orders, and work orders
- To schedule operations for work orders and repetitive schedules

- To schedule the procurement or shipment of materials through association with suppliers and customers

Use Calendar Maintenance (36.2.5) and Holiday Maintenance (36.2.1) to maintain the calendars.

Calendar Maintenance

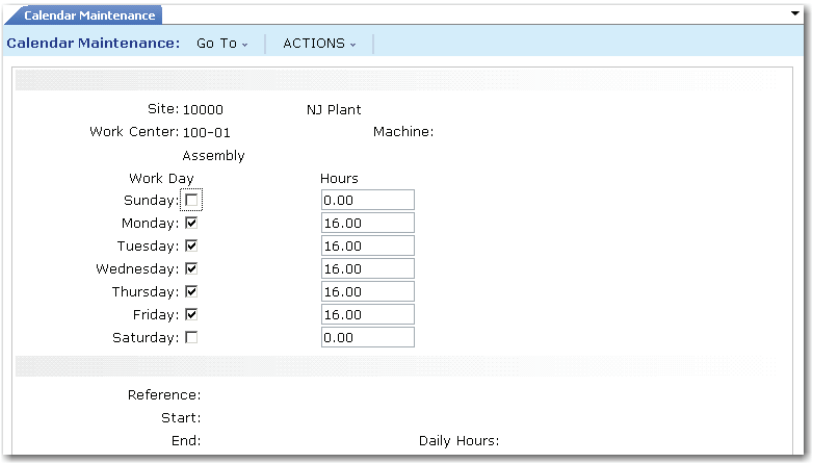
Use Calendar Maintenance (36.2.5) to specify normal work days and normal work hours for each site and its work centers. You create *shop* calendars for manufacturing using Calendar Maintenance, but you use Customer Calendar Maintenance (7.3.1) to create customer calendars. At least one calendar must exist.

You can create unique shop calendars by specifying some fields while leaving others blank. A default shop calendar has a blank site, work center, and machine. The system searches for a shop calendar in the following order:

- For the specific site, work center, and machine combination
- For site and work center with a blank machine
- For site with both work center and machine blank

If shift patterns vary because of overtime, increased or reduced shifts, or plant shutdowns, enter exception hours. Set up exceptions for a date range by specifying the number of hours that are added to or subtracted from normal work hours.

Fig. 2.1
Calendar Maintenance (36.2.5)



In a calendar, the check box is selected for each work day and is deselected for each non-work day. Manufacturing order due dates are scheduled only on work days. Each work day has a production capacity in hours. This should exclude breaks and nonproductive time. Manufacturing operations can be scheduled only up to the production capacity of the day.

Shop calendars are typically defined in this order:

1. Create a system calendar by leaving site and work center blank.
2. Create a calendar for each site with blank work centers. CRP uses this calendar to calculate capacity, including holidays.
3. Create work center calendars with site and work center filled in.

The system searches for a calendar from the most specific to the least specific—specific site, work center, and machine combination first and blank site, work center, and machine last.

You can specify exceptions, such as overtime or machine downtime for preventive maintenance. The system uses exception information only when preparing operation schedules, but not when calculating manufacturing order due dates.

Example On April 2, two hours of overtime are scheduled at site 10000. Enter OVERTIME as the reference code, April 2 as both start and end dates, and +2 as Daily Hours.

If an exception occurs on a day that is not part of the standard work week, add that exception to an existing day rather than changing the standard work week. Many scheduling programs assume that the work week has a certain number of days. Adding a day to the standard work week can result in inaccurate schedules.

Holiday Maintenance

Use Holiday Maintenance (36.2.1) to schedule holidays and other nonwork days that apply to an entire site.

Fig. 2.2 Holiday Maintenance (36.2.1)

Holidays are days that no one works; the plant is shut down and no production is scheduled. Manufacturing orders are never due and operations are not scheduled on a holiday.

Establishing Generalized Codes

When you install a new QAD database, a number of system and reference fields accept any kind of data, as long as it does not exceed the field length. You can customize the user interface by adding generalized codes and lookups.

Before implementing a module or particular functional area, the implementation team should determine which fields should have generalized codes and lookups.

Generalized codes are domain specific since these codes may vary widely based on the type and location of the business operation. For example, customer types, sales distribution channels, and buyer/planner codes could differ between a domain representing a business in England and one in Germany.

Important Some programs that update system-wide data such as User Maintenance (36.3.1) reference generalized codes. These generalized codes must exist in all domains or you may encounter errors editing a user record depending on what your current working domain is.

When using generalized codes, you can control three different conditions:

- What the acceptable values in a field are. Define these values in Generalized Codes Maintenance (36.2.13).
- Whether a list of acceptable values displays in a lookup browse on the field. Specify this in Drill-Down/Lookup Maintenance (36.4.8.1).
- Whether the codes you have created are the only acceptable codes (that is, whether the list is validated). This may require you to add a validation expression to the data dictionary.

See “Adding Validation” on page 20.

Field Validation

Before entering a list of generalized codes for a field, you must know the field’s name and size. In the character interface, the field name displays in a pop-up window when you press Ctrl+F with your cursor in the field. If the pop-up window indicates generalized codes validation, the system automatically verifies data entered in the field against the list of generalized codes.

You can also use Generalized Codes Validation Report (36.2.15) to view a list of all fields in the database that have schema validation assigned. This is the preferred method in the QAD .NET UI.

Note The system performs validation only when generalized codes have been defined for a field.

Example You have divided your customers into regions. The cm_region in the customer master is updated in Customer Data Maintenance (2.1.1). As part of the implementation process, you assign each customer to one of two regions. To ensure that only standard region codes are used, define them as generalized codes. Specify cm_region for the field name, the values US and X-US for the two regions.

Adding Generalized Codes

Figure 2.3 illustrates Generalized Codes Maintenance (36.2.13).

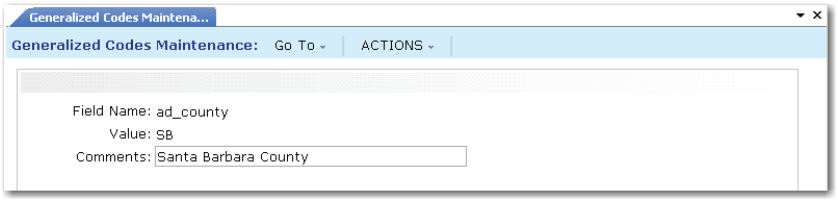


Fig. 2.3
Generalized Codes Maintenance (36.2.13)

Specify a field name and then enter valid values and comments. Values cannot exceed the length of the field. The comment displays next to the value in the lookup.

Adding a Lookup

To set up a lookup to display generalized codes, use Drill-Down/Lookup Maintenance (36.4.8.1). Enter the field name where you want the lookup and `gplu072.p` as the procedure to execute.

See “Maintaining Drill Downs and Lookups” on page 61.

This program creates the lookup with values from the assigned field. If the lookup should only be accessed from a particular screen, enter that program name as the calling procedure.

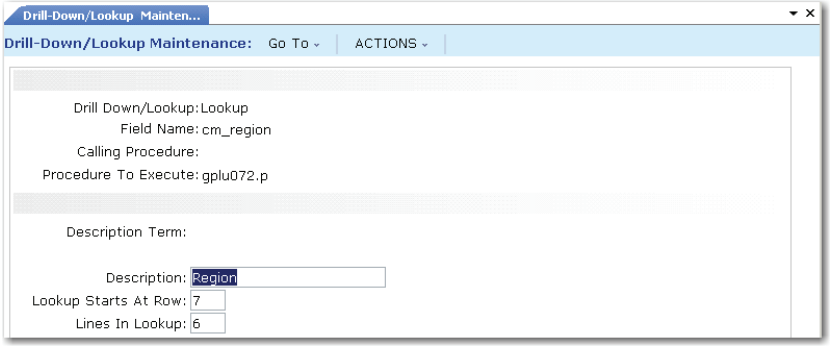


Fig. 2.4
Drill-Down/Lookup Maintenance (36.4.8.1)

The description defaults from the data dictionary, but can be changed here. If no description exists, the field name is a local variable. The description displays as the title of the lookup.

Note If you have not defined codes in Generalized Code Maintenance for a field with generalized codes validation, a lookup icon is displayed next to the field in the QAD .NET UI. If you then define codes for the field, the lookup icon is changed to a drop-down list in which the new codes are selectable. This change is visible when you log out and back in to the system, or when you switch domains.

Adding Validation

Generalized code validation, like field security, requires a special validation expression in the database dictionary that references the file `gpcode.v`.

Some fields already reference `gpcode.v`. These display in the Generalized Codes Validation Report. If you want to activate generalized code validation for other fields, you must change the data dictionary.

You can do this directly using full Progress or, if you have encrypted source, you can use the utility `utdbfx70.p`. Once you have added a validation expression, you must recompile the affected programs. For instructions on how to do this, refer to the *Progress Programming Handbook*.

To add validation for a local variable, you must insert the validation directly in the source code.

Important If you change the data dictionary, keep careful records and be prepared to repeat the change when new product versions that update the data dictionary are installed.

Using Reason Codes

Reason codes are used in security functions, sales quotes, sales order maintenance, purchase order returns, shop floor reporting, repetitive reporting, and the Product Change Control (PCC) module. They are also used if you have enabled change tracking and in several optional modules, such as WIP Lot Trace, Electronic Signatures, and Shipment Performance. Add other custom uses as needed.

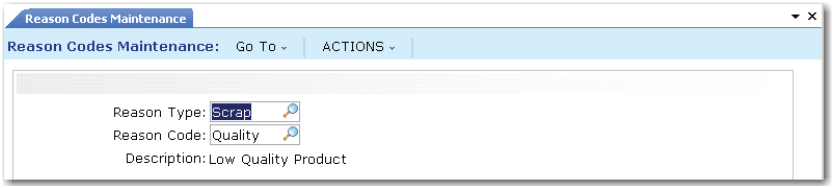


Fig. 2.5
Reason Codes Maintenance (36.2.17)

- Use codes of type User_Act for the Active Reason field in User Maintenance (36.3.1) and the Auto-Deactivation Reason field in Security Control (36.3.24).
- Use codes of type ESIG to indicate why a user is authorizing the data in an e-signature enabled program.
- Use codes of type QUOTE in the Reason Lost field of sales quotations.
- Use codes of type CORRINV to specify why an invoice must be corrected in Sales Order Maintenance (7.1.1).
- Use codes of type DOWN or DOWNTIME in the Reason field of labor feedback programs (16.20.1 through 16.20.4).
- Use codes of type ORD_CHG to associate changes made in Sales Order Maintenance to order detail, such as a change to the order line quantity or due date.

See “Tracking Changes” on page 33.

- Use codes of type DOWN, DOWNTIME, REJECT, REWORK, ADJUST, and SCRAP for reporting in Repetitive and Advanced Repetitive programs. Use these same codes with the optional manufacturing WIP Lot Trace module.
- Use codes of type SCRAP with Advanced Repetitive subcontract shipping programs and Scrap Transaction Maintenance (3.14).

See *User Guide: QAD Sales*.

- Use codes of type SHIPQTY and SHIPTIME with the Shipment Performance module.
- Use codes of type RTV (return to vendor) to define reasons entered in Purchase Order Returns (5.13.7).

Note Codes used in the PCC module are user-defined. They specify severity levels related to approval of change documents.

Managing Number Ranges

Some countries impose sequencing requirements related to tax filings or statutory reporting. In many countries, companies are legally required to prevent gaps in the numbering of official documents.

Additionally, certain business practices require different business units within the same corporation to maintain separate sequencing for similar documents such as purchase orders, sales orders, and supplier invoices.

Example In Italy, the number of an official document is strictly related to the date the document was printed, and it is a common practice to have multiple number ranges for shipment and invoice documents. In Brazil, the number of an official document is related to a specific physical site, requiring multiple number ranges with a prefix identifying a site code.

Number range management (NRM) supports varied sequencing requirements on a global scale. Features include gap control and multiple number ranges for the same document type.

NRM Overview

NRM generates sequence numbers built from one or more segments, each with its own set of characteristics and behavior.

You can add or remove segments during sequence definition, but once a sequence has been used to generate or validate numbers, you cannot change its structure.

Figure 2.6 illustrates a sample sequence with five segments: three fixed-value segments (NY and two dashes), one incrementing integer segment (1234), and one date-driven segment (06:15:07).

Fig. 2.6
Example Sequence Number

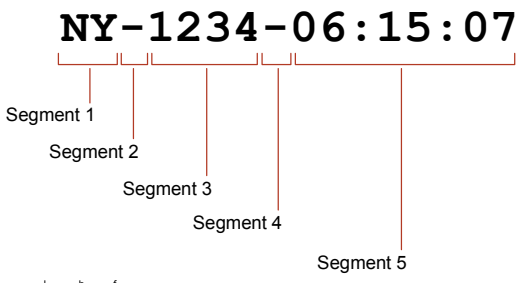


Table 2.2 describes the three segment types.

Table 2.2
Number Segment Types

Segment Type	Description	Required
Incrementing Integer	A range of values, with a lower bound, an upper bound, initial, and reset value.	Yes. Each sequence number must have one and only one incrementing integer segment.
Date-Driven	A value that depends on the transaction effective date or the fiscal period that corresponds to the effective date. The format is a compound string that allows the optional display of date components such as year, month, week, day, including delimiters between components. Delimiters separate the individual components of a segment. For example, 06:15:07 uses colons as delimiters.	No. Each sequence can have one date-driven segment.
Fixed-Value	Any printable character except a comma. For example, NY may be a fixed-value segment assigned by a client in New York. A fixed-value segment is not changed in any way during sequence number generation.	No.

Sequence Number Generation

To update a sequence number, the system examines each segment separately. Only date-driven or incrementing integer segment types are modified. A fixed-value segment is never changed.

Control Segments

You can set up a date-driven segment as a control segment. In this case, changing its value causes the incrementing integer segment to reset to its assigned reset value. When a control segment does not exist or does not change, the incrementing integer segment is incremented.

Sequence Parameters

Create sequence numbers and define sequence parameters using Number Range Maintenance (36.2.21.1). A distinct segment editor defines the format and parameters of each segment type.

Internal and External Sequences

There are two types of sequence number: internal and external.

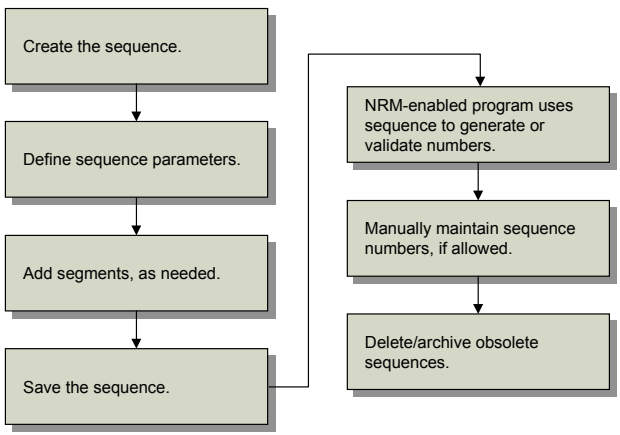
Internal sequences automatically generate numbers in ascending order as needed. NRM examines each segment in the sequence to determine whether to update its value. A fixed-value segment remains unchanged during sequence number generation.

External sequences accept a sequence number entered externally and validate it against a sequence definition. NRM verifies that the number belongs to the set defined by the sequence and that it has not yet been used. The system parses the number into segments and validates each segment against the corresponding segment in the sequence definition.

Sequence Life Cycle

Figure 2.7 illustrates the life cycle of a sequence.

Fig. 2.7
Sequence Life Cycle



To set up a sequence, create an ID, define general parameters, and add appropriate segments. Once a sequence is defined, a system program uses it either to obtain a new number or validate user-entered numbers. Programs must be specially designed to use NRM sequence numbers.

If you attempt to discard or void a number, the system checks the sequence definition to ensure that this is allowed.

You can delete and archive unneeded sequences.

NRM Sequences

Programs must be specifically enabled to use NRM. Currently, NRM sequences are used in several system functions:

Fixed Assets

An optional NRM sequence number can be specified in Fixed Asset Control (32.24) for automatically generating fixed asset ID numbers.

See *User Guide: QAD Financials B*.

General Ledger Daybooks

GL daybooks let you group and report GL transactions. Unposted transactions include the daybook code and daybook entry number. NRM generates entry numbers based on the ID of the daybook.

See *User Guide: QAD Financials A*.

Logistics Accounting

If you are using the optional Logistics Accounting module, two NRM sequences must be defined in Logistics Accounting Control (2.15.24) for distribution order shipments and sales order shipments.

In addition, if you use PO Shipper/Invoice Maintenance (5.13.14) to receive goods into a transit location, shipper/invoices use a unique NRM generated number.

See *User Guide: QAD Purchasing*.

Shipping

See *User Guide: QAD Sales*.

Many countries legally require businesses to maintain strict control when assigning numbers to shipping documents. This is also true when multiple number ranges are assigned to the same type of shipping document. To meet this need, NRM is required for all shipper functionality.

WIP Lot Trace

See *User Guide: QAD Manufacturing*.

An optional NRM sequence number can be specified in WIP Lot Trace Control (3.22.13.24) for generating WIP lot and serial numbers in the various functions that trace them.

Kanban

See *User Guide: QAD Lean Manufacturing*.

If you use dispatch lists to communicate kanban card authorizations to your suppliers, you must specify an NRM sequence in Kanban Control (17.24). The system uses the sequence to generate an ID number during dispatch list processing.

Legal Documents

In some countries, the transportation of merchandise requires a document to prove legality and possession of the inventory being moved. Typically, the legal document includes such elements as the document number, ship-from address, ship-to address, item number and description, quantity, and so on.

See *User Guide: QAD Sales*.

When you assign a legal document sequence ID with a ship-from address or document format, the system uses that sequence to generate numbers for legal documents created during shipping transactions.

Note The current release only supports issuing transactions. Receiving legal documents will be supported in a later release.

Sales Order Numbers

You can set up optional NRM sequences to generate numbers for new orders entered in Sales Order Maintenance (7.1.1) and Pending Invoice Maintenance (7.13.1).

See “Defining Sequences for Sales Orders” on page 31 for information.

Setting Up Sequences

Create sequences and define sequence parameters using Number Range Maintenance (36.2.21.1). NRM uses a unique sequence ID to retrieve data and generate new numbers. Use Sequence Browse (36.2.21.2) to view the defined structure of a sequence.

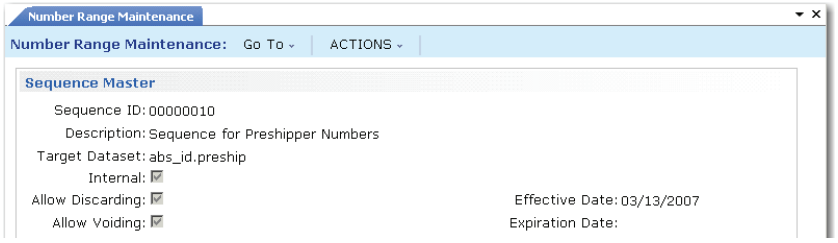


Fig. 2.8 Number Range Maintenance (36.2.21.1)

Sequence ID. Enter a code uniquely identifying a sequence. Create a new sequence or use Next/Previous to retrieve an existing sequence.

Description. Enter a description of this sequence, up to 40 characters.

Target Dataset. Enter the dataset identifier associated with this sequence. The target dataset can indicate who owns the sequence or where its numbers are used. A sequence owner can be a process, a document, or any other entity that the client program can recognize.

Note The target dataset could be the name of the principal database field where numbers from the sequence are used.

You cannot create a new sequence that intersects an existing sequence with the same target dataset—creating two sequences that could generate the same sequence number for the same target field.

For example, if sequences A and B both target field so_nbr, they cannot have a common element that could cause conflicts.

The following three target datasets are used with shippers:

- abs_id.shipper is used for sales order shippers.
- abs_id.preship is used by sales order pre-shippers.
- abs_id.mbol is used by master bills of lading.

For Fixed Assets, specify dataset fa_id.

For Logistics Accounting, specify:

- la_so_ship_id for sales order shipments
- la_do_ship_id for distribution order shipments

For shipper/invoices, specify dataset abs_id.poshipper.

For Kanban, specify dataset knbd.dispatch_id.

For legal documents, specify dataset abs_id.LegalDoc.

For sales orders, specify a dataset that begins with so_nbr.

See "Defining Sequences for Sales Orders" on page 31.

Note Additional language detail setup is required to use NRM to generate sales order numbers.

Internal. Specify whether the sequence numbers are supplied by an external source or automatically generated by NRM. Select the check box if numbers are generated by NRM.

Allow Discarding. Using a number causes it to be registered. This field determines whether a registered number can be discarded, leaving a gap in the sequence.

Clear (the default): Gaps are not allowed and numbers cannot be discarded from this sequence.

Selected: You can discard previously registered numbers from this sequence by reversing the register operation. NRM then erases all record of the sequence number, and the discarded number is replaced by a gap.

Allow Voiding. Determines whether you can mark a registered number as void.

Clear: Numbers in this sequence cannot be voided. This is the default.

Selected: You can void numbers and specify a brief description why. Voiding is recorded as a separate event in the sequence history.

Effective Date. Indicates the earliest date when this sequence can be used.

Expiration Date. Indicates the latest date when this sequence can be used.

Segment List

After you define the initial parameters for a sequence, Segment List and Editor frames display. The segment list shows the type and settings for each segment defined in the sequence. Segments display in ascending order, based on segment number.

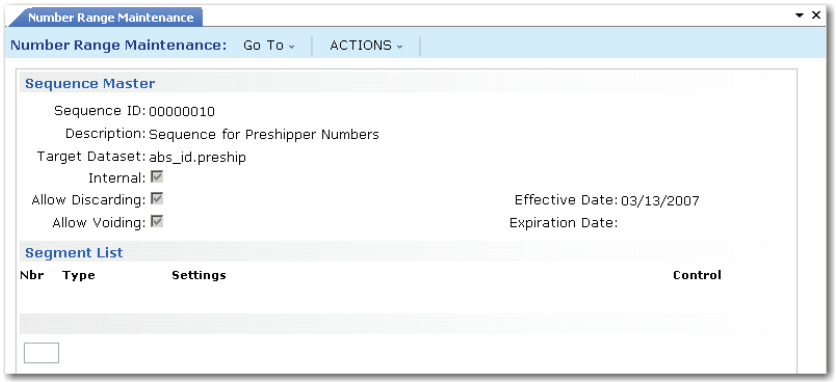


Fig. 2.9 Number Range Maintenance (36.2.21.1), Segment List Frame

Segment Editors

The segment editor used depends on the type of segment being defined. Use the editor to create or modify the segment format definition and assign a new segment number. There are four types of segment editors.

- *Fixed* segment editor for fixed value segments
- *Integer* segment editor for incrementing integer segments
- *Date* segment editor for date-driven segments
- *Fiscal* segment editor for date-driven segments, relative to fiscal periods

Fixed Segment Editor

Use the fixed segment editor to establish a fixed string value. You can use any printable character except a comma.

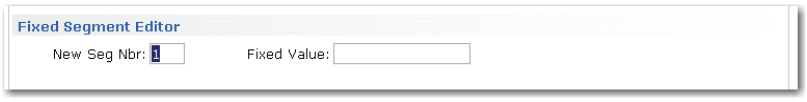


Fig. 2.10 Fixed Segment Editor

Integer Segment Editor

Use the integer segment editor to specify the initial, reset, minimum, and maximum values for a segment.

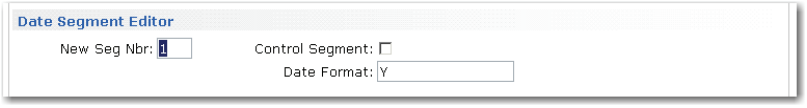
Fig. 2.11
Integer Segment Editor

Date Segment Editor

Use the date segment editor to tell NRM how to display a date component of a sequence number. Specify codes representing date components such as year, month, day. You can add components in any order, with optional delimiters. In the date segment 07/02, a forward slash is the delimiter. You can use any printable character except a comma or another date component as a delimiter.

You can indicate if this segment is a control segment. Changing the value of a control segment causes the incrementing integer segment to reset to its assigned reset value. The new value in the control segment ensures that the sequence numbers generated after resetting are unique within the target dataset.

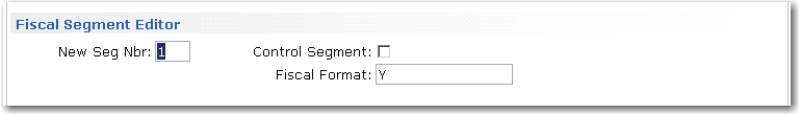
Fig. 2.12
Date Segment Editor



Fiscal Segment Editor

Use the fiscal segment editor to tell NRM how to display a fiscal date component of a sequence number.You can add fiscal segments only if the sequence has an expiration date. Codes represent a component of a fiscal period. Otherwise, this editor is exactly the same as the date segment editor.

Fig. 2.13
Fiscal Segment Editor



Setting Sequence Values

Use Sequence Number Maintenance (36.2.21.5) to set the next value for an existing sequence, when:

- The sequence is internal.
- Allow Discarding is selected.

The default in Sequence Value is the last number that was used. If you update it, the system validates the new value against the segment order and settings. It then increments the new value the next time the sequence is used.

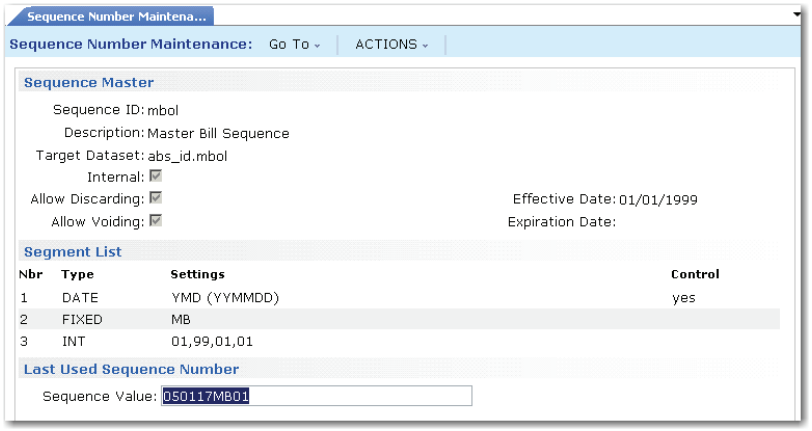


Fig. 2.14
Sequence Number Maintenance (36.2.21.5)

Defining Sequences for Sales Orders

You can define NRM sequences that assign numbers to sales orders in Sales Order Maintenance (7.1.1) and Pending Invoice Maintenance (7.13.1).

1 In Language Detail Maintenance (36.4.2), for Data Set so_seq_id and Field Name seq_type, create one or more mnemonics, assigned to Numeric Code 6 or higher.

Note Lower numbers are reserved for future functionality enhancements.

2 In Number Range Maintenance, create a new sequence ID and assign it to a dataset with the following value:

`so_nbr.`*`mnemonic`*

Where *`mnemonic`* comes from the language detail record defined in step 1.

You can define multiple sequences as needed if you want to assign order numbers based on site or some other criterion.

When any sequence definition is assigned to a dataset beginning with so_nbr, navigation changes in Sales Order Maintenance and Pending Invoice Maintenance during order entry. The system displays an additional Sequence ID field before assigning an order number. The look-up browse displays only IDs that have a dataset value beginning with so_nbr.

If you select a value, the system uses NRM to generate an order number based on the associated definition. If you leave Sequence ID blank, the system uses Sales Order Control (7.1.24) to determine the next order number.

Viewing Sequence Number History

When a client program uses a sequence to dispense or validate numbers, the system creates history records. Use Sequence Number History Report (36.2.21.13) to view history data on internal and external sequences.

You can view the sequence definition, which sequence numbers have been used, and which sequence numbers have not been used, including gaps. This report helps you to identify missing documents by reporting numbers that are not recorded in the sequence history.

Deleting and Archiving Sequences

Use Sequence Delete/Archive (36.2.21.23) to delete sequences and associated history. You can optionally archive information to an external file and later restore it using Archive File Reload (36.16.5).

Once sequence history is deleted, it no longer appears on the Sequence History Report.

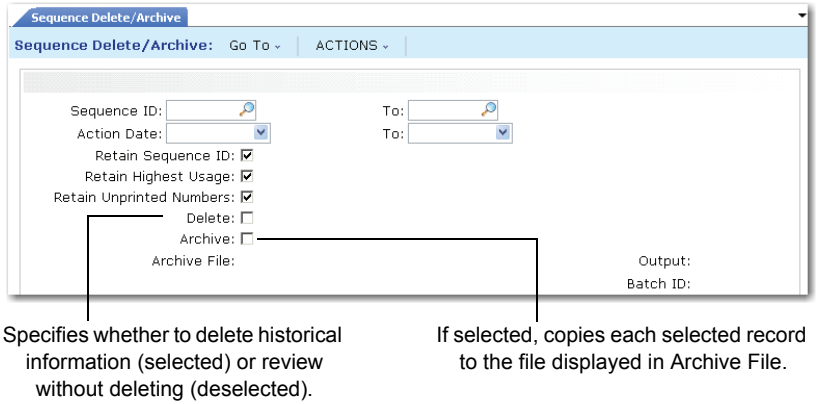


Fig. 2.15 Sequence Delete/Archive (36.2.21.23)

Tracking Changes

Use Change Tracking Maintenance (36.2.22) to mark sales order detail fields for change tracking. For line detail information in discrete sales orders, you can:

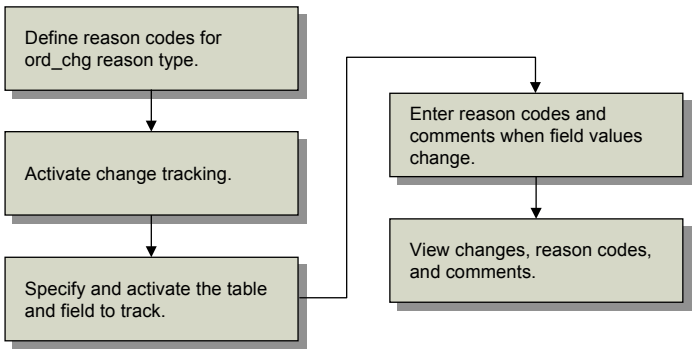
- Specify which field to track.
- Activate or deactivate tracking.
- Delete any records for fields that no longer require tracking.
- Allow users to enter a reason code and comments when the value of a marked field changes.
- Print the changes, reason codes that explain the changes, and any associated comments on a Booking Transaction Report (7.15.14).

See *User Guide: QAD Sales.*

Change Tracking Implementation Overview

When implementing change tracking, you work with different programs to set up codes, activate change tracking, specify what to track, then view results. Figure 2.16 illustrates the basic change tracking implementation flow.

Fig. 2.16
Change Tracking Implementation Flow



Defining Change Tracking Reason Codes

See "Using Reason Codes" on page 20.

You must define reason codes that explain changes to sales order detail in Reason Codes Maintenance (36.2.17). You specify ord_chg as the reason type. You must define at least one reason code for the ord_chg type to implement change tracking.

Create reason codes that fit your company's most common reasons for changes to sales order details. For example, you can create a Delete reason code for deleted orders.

Activating Change Tracking

Activate change tracking by selecting Keep Booking History in Sales Order Control (7.1.24).

Specifying Fields to Track

Use Change Tracking Maintenance (36.2.22) to:

- Specify which table contains the fields you want to track.
- Specify which fields to track.
- Delete any records for fields that no longer require tracking.

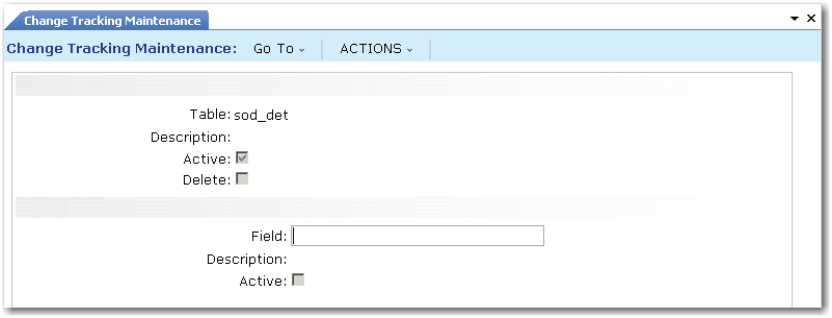


Fig. 2.17 Change Tracking Maintenance (36.2.22)

Table. Enter the database table that contains the field that is being tracked for changes. Currently, Change Tracking Maintenance tracks only the sales order detail (sod_det) table.

Description. Enter a brief description (24 characters) of the database table.

Active. Select to track changes for the database table you specified. Clear to deactivate tracking. The default is clear.

You must select Active for both the table and the field before change tracking begins.

Delete. Select to display the reason code pop-up in Sales Order Maintenance when the user deletes an entire sales order line. Clear if you do not want the reason code pop-up to display. The default is clear.

Note You must select Active and specify a field to track.

Once you complete these fields and click Next, the following frame appears.

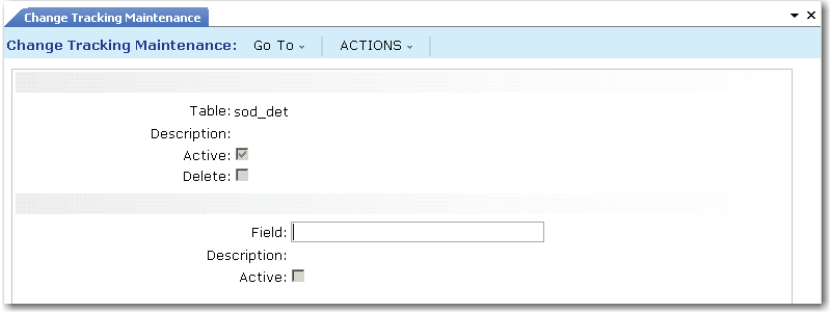


Fig. 2.18 Change Tracking Maintenance, Field Frame

Field. Enter the field to track. Currently, you can only track fields belonging to the sales order detail (sod_det) table.

Note To find the field name in the character interface, press Ctrl+F while your cursor is located in the field. In the QAD .NET UI, the field name displays as a field tip when your cursor moves over a field.

Description. Enter a brief description (24 characters) of the field.

Active. Select to activate tracking for the field you specified. Deselect to deactivate tracking (the default).

Review the tables and fields you specify and their active or delete status using Change Tracking Browse (36.2.23).

Reason Code Pop-Up

After you activate change tracking and specify a table and field to track, when the user changes or deletes the value of the field, a reason code pop-up displays. Currently, only the sales order detail table can be tracked; therefore, the reason code pop-up displays in Sales Order Maintenance (7.1.1).

Select a code that indicates the reason you are changing the value of the field or deleting the line. The reason type associated with the code must be ord_chg.

Even though you can track multiple fields, you are only prompted once with the reason code pop-up. Use the comment screen to explain multiple changes you made to the sales order line.

Viewing Changes

To view changes you tracked, use Booking Transaction Report (7.15.14). The report displays the reason and comments related to a discrete sales order line change.

To display the changes:

- Set Summary/Detail to Detail.
- Select Include Reason/Comments.

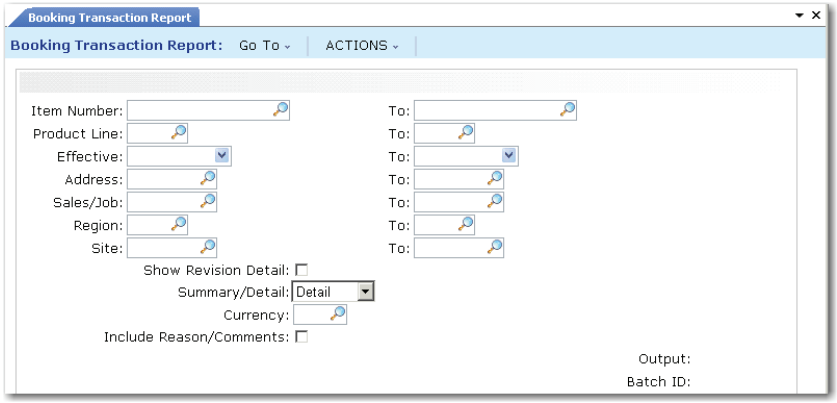


Fig. 2.19
Booking Transaction Report (7.15.14)

Chapter 3

System Interface

The System Interface menu contains programs that control menus, messages, multi-language installations, browses, and help.

Languages, menus, messages, user functions, help, labels, and e-mail definitions are all system-wide data and apply to all domains in a database.

Overview of User Interface

QAD application functions can be used in two ways:

- All functions are available in a .NET-based graphical user interface (UI).
- A subset of functions can be used in a simplified character interface.

For programs that run in the .NET UI, some present simple HTML screens; other use an advanced component-based technology that supports tabbed navigation and other technologies that are not available with standard programs.

The functions on the System Interfaces menu let you manage different aspects of information that display on the UI and that affect how users interact with the system. Some of these functions apply only to the character UI, some to .NET UI, some to component-based functions, and some to all functions.

Table 3.1 lists all the menu items on the System Interface Menu. The Program column indicates if this is a standard Progress program (.p) or a component with activities. The UI column indicates which user interface the program can be run from. Some programs—such as the component activities—can only be opened in the .NET UI; others can be used in both character and the .NET UI. The column labeled Affects indicates which programs the System Interface program affects:

- All programs, including standard Progress programs in the character UI, Progress programs running as HTML screens in .NET UI, and component-based functions in the .NET UI
- Standard programs in both character and .NET UI
- Standard programs in character only
- Standard programs in .NET UI only
- Component-based activities in .NET UI only

Note While it may be possible to execute a program from both the .NET UI and the character UI, the effect of the program may be seen in only one UI. For example, you can set up menu URLs from the character UI or the .NET UI, but the links are only active in browses run from the .NET; they cannot be executed from character browses.

Table 3.1
System Interface Menu

Menu No.	Menu Description	Program	UI	Affects
36.4.1	Language Maintenance	Component	.NET	All programs
36.4.1.1	Language Create	Activity		
36.4.1.2	Language Modify	Activity		
36.4.1.3	Language View	Activity		
36.4.1.3	Language Delete	Activity		
36.4.2	Language Detail Maintenance	mglngumt.p	Both	Standard programs in either UI
36.4.3	Report Translation Maintenance	Component	.NET	Component-based reports
36.4.3.1	Report Translation Create	Activity		
36.4.3.2	Report Translation Modify	Activity		
36.4.3.3	Report Translation View	Activity		
36.4.3.4	Report Translation Delete	Activity		
36.4.4	**Menu System Menu**			
36.4.4.1	Menu System Maintenance	mgmemt.p	Both	All programs
36.4.4.2	Menu System Report	mgmerp.p	Both	All programs
36.4.4.7	Menu Substitution Maintenance	mgmsmt.p	Both	Standard programs in character UI
36.4.4.8	Menu Substitution Browse	mgbr013.p	Both	Standard programs in character UI
36.4.4.13	Program Information Maintenance	mgpgmimt.p	Both	Standard programs in either UI
36.4.4.14	Program Information Browse	mgbr060.p	Both	Standard programs in either UI
36.4.4.15	Program Information Update	mgpgmiut.p	Both	Standard programs in either UI
36.4.4.22	Business Component View	Activity	.NET	View only
36.4.6	**Message Menu**			
36.4.6.1	Message Maintenance	mgmsgmt.p	Both	Standard programs in either UI

Table 3.1 — *System Interface Menu* — (Page 1 of 4)

Menu No.	Menu Description	Program	UI	Affects
36.4.6.2	Message Browse	mgbr006.p	Both	Standard programs in either UI
36.4.6.3	Message Report	mgmsgrp.p	Both	Standard programs in either UI
36.4.6.13	Configured Message Maintenance	mgcfmmt.p	Both	Standard programs in either UI
36.4.6.14	Config Msg Verif Report	rcrp14.p	Both	Standard programs in either UI
36.4.8	**Browse Menu**			
36.4.8.1	Drill-Down/Lookup Maintenance	mgdlfhmt.p	Both	Standard programs in either UI
36.4.8.2	Drill-Down Browse	mgbr001.p	Both	Standard programs in either UI
36.4.8.3	Lookup Browse	mgbr005.p	Both	Standard programs in either UI
36.4.8.5	User Tool Maintenance	mgtoolmt.p	Both	Standard programs . in .NET UI only
36.4.8.7	Stored Search Maintenance	Component	.NET	Component-based browses in .NET UI
36.4.8.7.1	Stored Search View	Activity		
36.4.8.7.2	Stored Search Export Browse Defaults	Activity		
36.4.8.7.3	Stored Search Delete	Activity		
36.4.8.9	Browse URL Maintenance	mgburlmt.p	Both	Standard browses in .NET UI
36.4.8.10	Browse URL Browse	mgbr221.p	Both	Standard browses in .NET UI
36.4.8.13	Browse Maintenance	mgbwmt.p	Both	Standard programs in either UI
36.4.8.14	Browse Master Browse	mgbr003.p	Both	Standard programs in either UI
36.4.8.15	Browse Detail by Field	mgbr040.p	Both	Standard programs in either UI
36.4.8.18	View Maintenance	mgvwmt.p	Both	Standard programs in either UI

Table 3.1 — *System Interface Menu* — (Page 2 of 4)

Menu No.	Menu Description	Program	UI	Affects
36.4.8.19	View Browse	mgbr020.p	Both	Standard programs in either UI
36.4.10	User Function Maintenance	mgufmt.p	Both	Standard programs in character UI
36.4.11	User Function Browse	mgbr008.p	Both	Standard programs in character UI
36.4.12	**User-Defined Field Maintenance**			
36.4.12.1	User-Defined Field Create	Activity	.NET	Component-based activities. See the section on Customization in *User Guide: QAD Financials A* for details on these activities.
36.4.12.2	User-Defined Field Modify	Activity		
36.4.12.3	User-Defined Field View	Activity		
36.4.12.4	User-Defined Field Delete	Activity		
36.4.13	**Field Help Menu**			
36.4.13.1	Field Help Maintenance	mgflhusr.p	Both	Standard programs in either UI
36.4.13.2	Field Help Report	mgflhdrp.p	Both	Standard programs in either UI
36.4.13.3	Field Help Book Report	mgflbook.p	Both	Standard programs in either UI
36.4.13.4	Procedure Help Report	mgflprrp.p	Both	Standard programs in either UI
36.4.13.13	Field Help Dump	mgfldmp.p	Both	Standard programs in either UI
36.4.13.14	Field Help Load	mgflld.p	Both	Standard programs in either UI
36.4.14	User Option Telnet Maintenance	mgusrtmt.p	Both	Standard programs in .NET UI
36.4.15	User Option Report	mgusrrp.p	Both	Standard programs in .NET UI
36.4.17	**Label Menu**			
36.4.17.1	Label Master Maintenance	gplblmt.p	Both	Standard programs in either UI
36.4.17.2	Label Master Browse	gpbr405.p	Both	Standard programs in either UI

Table 3.1 — *System Interface Menu* — (Page 3 of 4)

Menu No.	Menu Description	Program	UI	Affects
36.4.17.5	Label Detail Maintenance	gplbldmt.p	Both	Standard programs in either UI
36.4.17.6	Label Detail Browse by Field	gpbr406.p	Both	Standard programs in either UI
36.4.17.7	Label Detail Browse by Program	gpbr407.p	Both	Standard programs in either UI
36.4.17.24	Label Control	gplblpm.p	Both	Standard programs in either UI
36.4.18	Customization Export	Activity	.NET	Component-based functions in .NET
36.4.19	Customization Delete	Activity	.NET	Component-based functions in .NET
36.4.20	E-mail Definition Maintenance	mgemmt.p	Both	Standard programs in either UI
36.4.21	**Report Setup Menu**			
36.4.21.1	Report Maintenance	rprprsmt.p	Both	Standard programs in either UI
36.4.21.2	Report Synchronization	rprpsync.p	Both	Standard programs in either UI
36.4.21.3	Report Parameter Maintenance	rprpptmt.p	Both	Standard programs in either UI
36.4.21.4	Report Parameter Synchronization	rpptsync.p	Both	Standard programs in either UI
36.4.21.13	Admin User Variant Maintenance	rpuvmt01.p	Both	Standard programs in either UI
36.4.21.14	Personal User Variant Maintenance	rpuvmt02.p	Both	Standard programs in either UI
36.4.21.21	Report Resource Import	rprroim.p	Both	Standard programs in either UI
36.4.21.22	Report Resource Export	rprroex.p	Both	Standard programs in either UI
36.4.21.24	Report Control	rppm.p	Both	Standard programs in either UI
36.4.22	Session Master Maintenance	mgsessmt.p	Both	All programs

Table 3.1 — *System Interface Menu* — (Page 4 of 4)

Using Multiple Languages

A single QAD database can be set up with multiple domains, each with a distinct language. With a Unicode database, languages with different code pages can be supported in the same database. Defining a Unicode database is an installation option described in the relevant installation guide for your system.

Unicode is a data storage and display protocol—essentially a combination of a code page and collation table. The QAD Unicode database uses the UTF-8 code page, which includes all characters from all languages; the collation table sets the sort order for those characters for display or reporting. The advantage of Unicode is that one database can contain as many languages as you require. Without Unicode, a single database can include only languages that share the same code page—such as French, German, and Spanish—or languages with compatible code pages—such as English and Chinese.

Each user is also assigned a language. A user logging in through the .NET UI can access any domain in a Unicode database regardless of their associated language. The character client, however, does not support Unicode. A user logging in through the character UI must log in to a domain with a code page that is compatible with the language they are assigned, as defined in Domain Create (36.1.1.1.1). See *User Guide: QAD Financials A* for details about how domains are defined.

Example A database includes three domains: French, Chinese, and English. John is assigned the Chinese language. John can log in to all three domains in the .NET UI. When using the character UI, John cannot log in to the French domain; an error displays if he tries to do this.

Multi-language capabilities are used in two areas:

- Screens and screen elements such as labels and messages are displayed in multiple languages.
- Data is stored and displayed in multiple languages.

All single set of r-code can be used with all supported languages. Language-specific data is loaded during installation. Based on the language associated with the user, screen labels, menus, messages, and field help is displayed in the appropriate language.

Labels, menus, and messages are stored in the production database. Field help is in the field help database, `mfghelp.db`.To retrieve data in multiple languages, each of these types of information is stored once in each language.

Most orders include comments, which often must be in multiple languages. These can be stored in multiple languages and retrieved by language ID. You can also customize menus and messages and assign a language ID so the system knows which entry to display.

Standard programs support only a limited amount of data that is stored and displayed by language ID. This data is defined in Language Detail Maintenance (36.4.2). However, component-based functions provide extended translation features. You can provide your own language translations for descriptions of shared data such as accounts, sub-accounts, and cost centers. These features are described in *User Guide: QAD Financials A*. You can also provide translations for custom labels assigned to component-based reports.

Multiple Languages and Unicode

.NET UI clients are fully Unicode compliant; a user logged in through the .NET UI can access any domain, regardless of its defined code page.

Character clients do not support Unicode and must log in to a domain with a compatible code page, as defined in Domain Create. This is to prevent the possibility of data corruption when characters cannot be interpreted correctly. Once logged in through the character UI, users are limited in which domains that they can switch to.

However, some domain switching can happen behind the scenes when processes such as Distribution Requirements Planning (DRP) are being used. In a Unicode database, these processes should be run from batch to prevent code page issues.

If your implementation includes a Unicode database and you have some users executing functions from character clients, you should be aware of the following restrictions.

Creating Sites

When you create a site in one domain, connection records for that site are optionally created in other domains in the database. This is to support cross-domain features such as Distributed Requirements Planning (DRP).

If any of the domains in the database have incompatible code pages, creating site connection records is not possible from the character UI. If you need to create these records, you must use Site Maintenance (1.1.13) from the .NET UI.

Running Batch Jobs

If you routinely run batch jobs across multiple domains in a Unicode database, you should set up scripts with the correct Unicode startup parameters to avoid code page issues.

See “Invoke Batch Processing from CIM” on page 113.

Running DRP and MRP

DRP is one of the areas where the system may need to change to a different domain based on the sourcing relationships for items that you have set up. If you have set up relationships between sites in domains that have incompatible code pages, this switching cannot be done when DRP is run from a character client. This is true of stand-alone DRP using programs on 12.13, or running Materials Requirements Planning (MRP) when the DRP/MRP Combined field is selected in DRP Control (12.13.24).

If the system detects that the domains are incompatible, an error is generated and DRP will not complete its execution.

To avoid errors, run DRP or MRP in batch mode with a Unicode database; for example, use the following parameters with your Progress batch command to use the UTF-8 code page:

```
-cpinternal utf-8 -cpcstream utf-8
```

For details on setting up the batch job, see “Invoke Batch Processing from CIM” on page 113.

Setting Up Multiple Languages

A base set of language codes is supplied with the system. System-defined languages cannot be changed or deleted.

Use the Language Maintenance activities (36.4.1) to view all records, create new ones, modify user-defined records, or delete user-defined records. Language activities are described in *User Guide: QAD Financials A*.

To implement multiple languages in a database, you must:

1. Load the appropriate language data, either during initial system implementation or later. Loading translated labels marks the language as installed so that it is available to be associated with users.
2. Create countries for users in Country Create (36.1.3.1.1). Each country to be associated with a user must have an ISO country code specified in the Alternate Code field in Country Code Data Maintenance (2.14.1). Regional display defaults are determined by the ISO country, such as date and number formats.
3. Designate the default language and country code for each user in User Maintenance (36.3.1). This ensures that when the user logs on, the system displays data in that person's language with the correct regional formatting.

See *User Guide: QAD Security and Controls*.

If the language is the same for all users but multiple language comments are required for orders, you only need to define the separate language codes using the Language function. A number of codes for supported languages are already defined.

Language Detail Maintenance

Some program options appear on the screen using alphabetic codes or words. Internally, these options are controlled by numeric codes. Mnemonics and labels provided in English are not always appropriate in other languages. Use Language Detail Maintenance (36.4.2) to change, add, and delete mnemonic codes and labels.

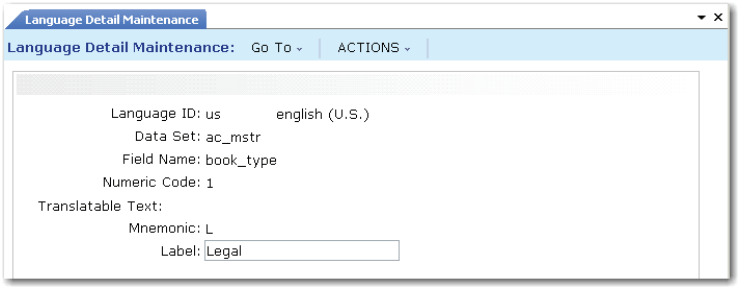


Fig. 3.1
Language Detail Maintenance (36.4.2)

Data Set. Enter the program name, a database table name, or an abbreviation of the functionality for a field.

Field. Enter the field name associated with the data set.

Numeric Codes. These are the values used by the programs. A mnemonic code can be assigned for each numeric code. Codes cannot be added or edited.

Mnemonic. Mnemonic codes are already assigned for each field with several system-specified options. These codes can be changed, added, or deleted using this program.

Label. Default labels already exist for the different mnemonic codes. These labels can be changed, added, or deleted using this program.

Report Translation

The Report Translation activities (36.4.3) let you create, view, modify, and delete custom labels used in component-based reporting functions.

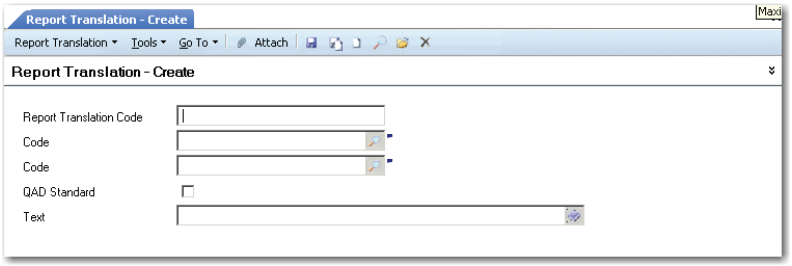
Using these activities, you can modify the standard QAD translations or create new translations. Labels are grouped by business component and, then, by report. The labels that are common to all reports of a business component are stored with a blank report name. The labels that are specific to one report reference the report name.

The report labels are stored in the database and are retrieved at run-time (report execution) in the correct language and sent to the Crystal reports viewer. This viewer receives the report data and labels from the application and formats them according to the layout in the `.rpt` file.

For standard QAD labels, the QAD Standard check box is selected and the Report Translation Code cannot be changed.

If you want to add a new label to a component and make it available to all reports for that component, ensure that the label is defined in the `.rpt` file with the correct Report Translation Code.

Fig. 3.2
Report Translation Create (36.4.3)



Report Translation Code. Specify the Report Translation Code to use on reports and to which the translation is linked.

Business Component. Specify the business component to which the report belongs.

Report Name. Specify the report to which the translation applies.

QAD Standard. Select to indicate if the translation is a standard QAD translation.

Text. Type the translated text.

Customizing Menu Information

Menus are loaded as part of initial system implementation. Both standard Progress programs and component-based functions are set up in the menu system.

- A standard Progress program is generally accessed from the menu, although you can add links from one program to another. See “Creating Access to Other Programs” on page 65 for details on linking programs.
- Activities associated with components can be selected from the menu; they can also be accessed from a right-click menu in the component view; see Figure 3.3.

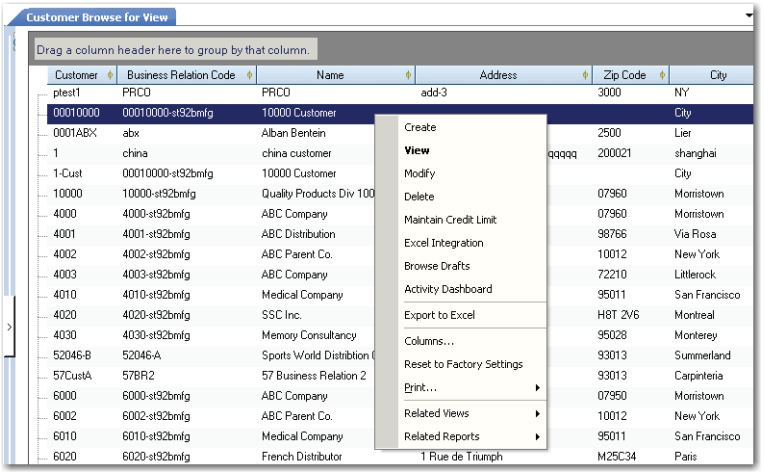


Fig. 3.3
Right-Click Menu in Component View

Executing Menu Items

From the .NET UI, you can execute standard programs and component activities in these ways:

- Type enough of the menu number or menu label in the search box to uniquely identify the program and press Enter.
- For standard programs only, type the program name, such as `mgmemt.p`, in the Search box and press Enter.
- Double-click the menu item in the menu listing.
- Choose a program or activity from the Go To menu from in an open program screen.

In the character UI, you can execute only standard programs and only in these ways:

- Type the program name, such as `mgmemt.p`, or number, such as 36.4.4.1, at any menu prompt.

  Note If you are currently on another branch of the menu tree (for example the 1.4 menu), enter a period before the menu number (.36.4.4.1). Type a partial number from a submenu, such as 4.4 while located at menu 36.

- Press a function key that is assigned to this program or select the program from the User Menu. See “Defining User Menu and Function Keys” on page 83 for details.

Menu Structure

The menu system controls what displays when a user logs in. It is designed like a product structure, recorded as single-level relationships between a parent menu item and a child item. At the top level in the character UI, the parent item is the Main Menu (Menu 0).

Note The menu groups represented by the folders in the .NET UI are referenced through the letter A. For example A.1 is Sales, A.2 is Manufacturing and so on.

At lower levels, the parent item is a submenu such as the Call Management Menu (11.1) or an executable function.

Menus are stored in a table indexed by language ID. Each user has a default language. When a user logs on, the system determines the user language and displays menu text in that language.

As a user moves through menus and makes selections, the Execution File specified in Menu System Maintenance controls the function or submenu that displays. Selecting 2. Addresses from the Main Menu sets the Execution File to 2, telling the system to access Menu 2. Selecting 12 from Menu 2 sets the Execution File to admgmt06.p, telling the system to run Company Address Maintenance.

QAD applications are delivered with all offered menus and functions. You can remove menus for programs that you do not use by either taking them off the menu or controlling them with menu security.

Note It is easier to update your software releases if menus are not modified. Instead, use menu security for functions you do not use. In the character UI, you can set up User Menus for commonly used menus and functions. In the .NET UI, each user can use the Favorites feature to define a personal menu subsystem of commonly used functions.

Menus and Security

Users can only see functions they have been given access to, based on their role membership. Menu items are assigned to roles in Role Permissions Maintain (36.3.6.5). Each user is assigned to one or more roles in Role Membership Maintain (36.3.6.6) in the context of specific domains and entities.

When a user logs in, the system builds the menu for that user based on the login domain and entity (workspace in the .NET UI) and the set of roles assigned to the user. The menus represent the combination of all functions the user has access to in that domain and entity.

Note When logging in through the character UI, a user must have access to the primary entity of the selected domain; in the .NET UI, the workspace represents any valid domain/entity combination for which the user has been granted access.

When you add an item to the menu, access must be explicitly granted before anyone can see that menu item and run the function, including yourself. You must use Role Permissions Maintain to add the menu item to the appropriate roles.

Note Users cannot run any program that is not on the menu. If you want users to be able to run custom programs or utilities from an application session, you must add the program to the menu and give the appropriate user roles access to it.

See *User Guide: QAD Security and Controls* for details on how to set up security.

Menu System Maintenance

Use Menu System Maintenance (36.4.4.1) to assign menu labels and execution files to menu numbers.

In the QAD .NET UI, the interface to Menu System Maintenance has been enhanced to make it easier to use. For further information, see *User Guide: QAD User Interfaces*.

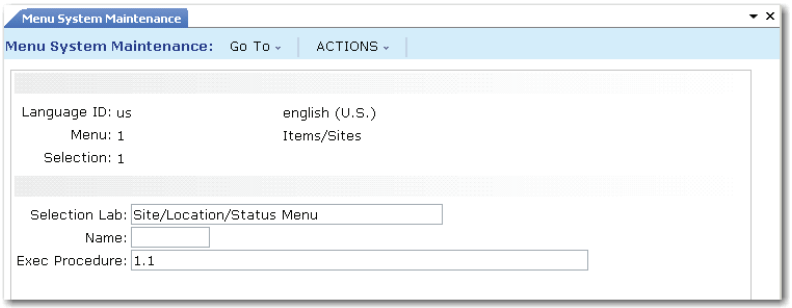
You can control the menu numbers and the names associated with programs in several ways.

- Move menu items.
- Change menu names.
- Create shortcut names for menu items.

Note If you make these changes, they may be lost during software updates when menus are reloaded.

Important Menus are cached in memory when you log in to the system. You must log out and log in again to see any changes made with this program. In addition, if you add menu items, you must grant access to them before anyone can see them. See the previous section on menus and security.

Fig. 3.4
Menu System Maintenance (36.4.4.1)



The Name field lets you call programs using keywords. For a program buried deep in the menu structure, you can add a name and then execute the program by typing that name on any menu command line.

Example Many of your users create sales orders. Enter SO in the Name field for 7.1.1. Users can then type SO at the menu prompt or Search box and Sales Order Maintenance (7.1.1) displays.

The execution procedure can be:

- A menu number such as 1.1
- A Progress program such as `sosomt.p`
- A component-based activity specified in the form of a uniform resource name (URN) such as `urn:cbf:BCreditor.Modify`
- A process map specified in the form of an URN such as `urn:pmap:IndustryProcessLevel1`
- A browse collection specified in the form of an URN such as `urn:collection:fcb4af10-e778-4db5-9461-766f5b7e2891`

Setting Up Menu Substitutions

Use Menu Substitution Maintenance (36.4.4.7) to set up a link between two standard programs so that when users select one from a menu, the other program displays. This is useful for substituting custom versions of existing programs.

Menu substitution affects standard programs in the character UI in these ways:

- Replaces browses with inquiry programs
- Replaces standard programs with custom versions

Note This program applies to standard Progress programs in the character environment only; you cannot use it with component-based functions and it does not affect the menu in the .NET UI. In the .NET UI, standard programs and browses always display.

In the character interface, which program is invoked for a particular user depends on whether menu substitution is enabled in the user record in User Maintenance (36.3.1).

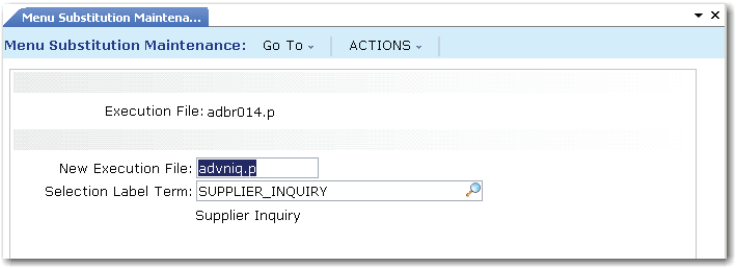


Fig. 3.5
Menu Substitution Maintenance (36.4.4.7)

1. Enter the program name in Execution File. Users selecting this program from a menu will actually be running the one entered into the New Execution File field.

2. Enter the substitute program name in New Execution File. This is the name of the program to replace the one entered in Execution File. Users will run this program when they select the one entered in the Execution File field. You can use wildcards. For example, if you want to replace all inquiry programs with the browse versions, you enter `*iq*` in the Execution File field and `*br*` here.

3 Enter a label term in Selection Label Term. The long label contained in this term appears in the title bar and menu list of the substituted program.

Defining Program Information

The program information table contains a record for each standard menu-level program, defining characteristics that affect the way it runs. Each standard program in the .NET UI must have a record both in Menu System Maintenance and in Program Information Maintenance (36.3.21.1).

Note Component activities are defined in Menu System Maintenance, but not in Program Information Maintenance.

In addition to menu-level programs, lookups must be defined in order for the lookup icon to display next to a field in an HTML screen.

Program information records are loaded with other default data during system installation and can be viewed in Program Information Maintenance or Program Information Browse (36.4.4.14). You must manually create records for any custom programs that you want users to be able to access as HTML programs from the .NET UI. Otherwise, they will open in Terminal mode.

See "Setting User Telnet Options" on page 87 for details on defining telnet settings.

Note Terminal mode lets you run programs with a character-mode UI, using a telnet session. The system must be configured to support telnet sessions for the program to display.

The following table lists the default settings for different program types.

Table 3.2 Default Display Settings in .NET UI

Program Type	Web Logic Implemented	Type
Browses, lookups	Selected	Blank
Special HTML programs such as Browse Maintenance and Kanban workbenches	Selected	Blank
HTML reports and inquiries	Clear	Desktop
HTML maintenance programs	Clear	Desktop
Terminal-mode utilities	Clear	Blank

The Multi Domain field indicates a standard program that updates data that applies to all domains in a database. When this is selected:

- In the .NET UI, the string All Domains displays in the Menu Properties window for the associated menu program. Otherwise, the domain name displays.
- In the character UI, the string All Domains displays in the menu title bar when Header Display Mode is set to 2 or 3 in Security Control (36.3.24).

Appropriate default settings for the Multi Domain field are set during installation. For example, generalized codes apply to each domain separately so Multi Domain is not selected by default for `mgcodemt.p`. Country code data applies to the database as a whole so Multi Domain is selected by default for `adctrymt.p`.

Important You can update the setting for your custom programs or change it if you want the current working domain to continue to display even when a user is updating a table that applies across domains. This change affects what displays on the UI only. The program continues to update data for all domains.

Adding Records

To execute a program with an HTML display from the .NET UI menus, add a record to Program Information Maintenance for each custom program. When a record does not exist for a program, the system assumes that it should run in Terminal mode.

If you want to create records for a number of programs at once, use Program Information Update (36.4.4.15) to scan them and automatically create records.

See page 58 for details.

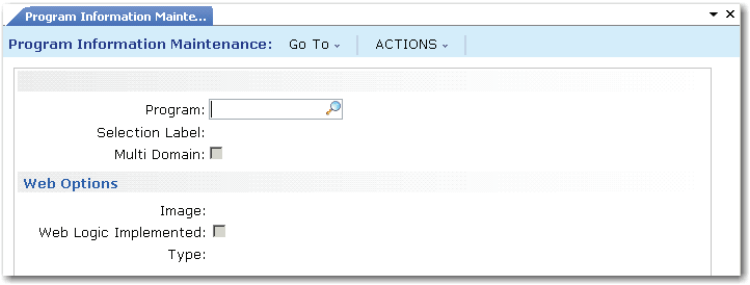


Fig. 3.6 Program Information Maintenance (36.4.4.13)

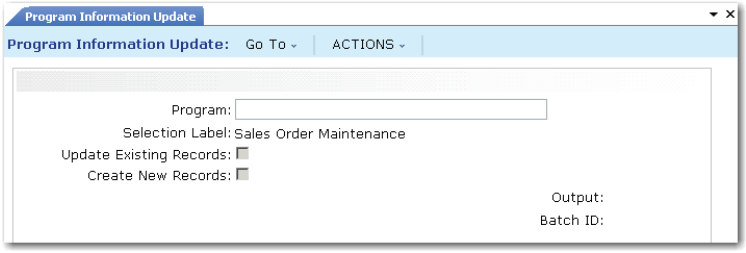
To create program information records:

1. Enter a custom program name.
2. Indicate if this program updates data for all domains in the database.
3. Click Next to continue.
4. Select the Web Logic Implemented check box if this is a browse written according to QAD standards. Clear the Web Logic Implemented check box if this is a maintenance program, report, or inquiry that you want to display in HTML.
5. Leave the Type field blank for a browse. Specify Desktop for an HTML maintenance program, report, or inquiry.

Program Information Update

Use Program Information Update to automatically add records for custom programs to Program Information Maintenance. Use this utility as an alternative to adding records manually. It is especially useful for initially populating records with referenced tables.

Fig. 3.7
Program Information Update (36.4.4.14)



Viewing Business Components

Use Business Component View to view a list of the business components in the system. Not all business components are represented by activities in the system, so this gives you a complete view as well as displaying the technical name of the component. This might be needed in troubleshooting as well as developing API calls to the component.

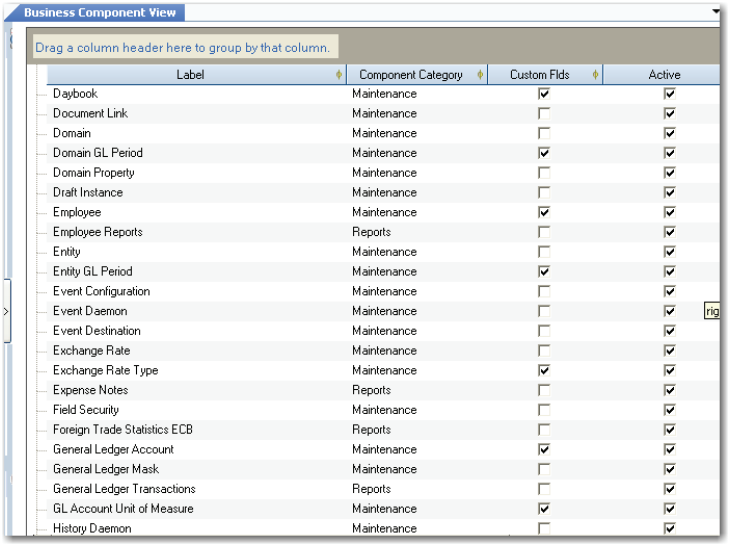


Fig. 3.8
Business Component View (36.4.4.22)

Modifying Messages

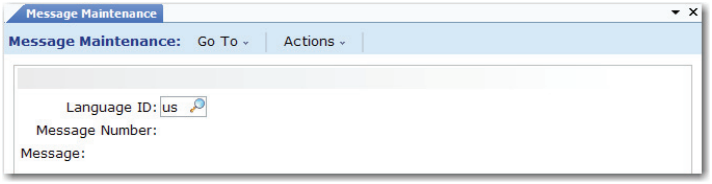
The system has three kinds of messages:

- Validation messages stored in the data dictionary. These display when the contents of the field do not match its specifications.
- Application messages for standard Progress programs stored in the database.
- Application messages for component-based programs stored in external resource files.

Note Numbered Progress error messages sometimes display when a Progress instruction fails. Most of these messages are handled by the system, and a system message is substituted, so this should occur rarely.

You can modify standard messages in Message Maintenance (36.4.6.1). One reason for changing messages is multiple language requirements. If a message seems unclear to some end users, an administrator can clarify its meaning. You can also add your own messages for custom code.

Fig. 3.9
Message Maintenance (36.4.6.1)



Changing messages can create the same version control problems that occur when menus are changed. Be careful to use message numbers not likely to be used in a later product version. Messages in the ranges 9000–9999 and 90000–99999 are reserved for customer use.

Creating and Managing Browses

You can use system features to define browses that display selected data in the form of a table. See *User Guide: QAD User Interfaces* for details on using browse features.

Component-based functions have browses that support extended configuration features. These are described in *User Guide: QAD Financials A*.

Standard browses can display data from one table or from several tables joined into a specific view of the data. Use the functions on the Browse Menu (36.4.8) to manage browses:

- Use Drill-Down/Lookup Maintenance (36.4.8.1) to attach browses to program fields.
- Use User Tool Maintenance (36.4.8.5) to create links from one program to another.
- Use Stored Search Maintenance (36.4.8.7) to manage default browse information for component-based views.
- Use Browse URL Maintenance (36.4.8.9) to create links to other resources that can be accessed from a browse.
- Use Browse Maintenance (36.4.8.13) to create your own browses or modify system supplied browses.
- Use View Maintenance (36.4.8.18) to join information from more than one table to be used in a custom browse.

Maintaining Drill Downs and Lookups

Two types of browses are available in standard Progress programs:

- *Lookup browses* return the value you select to the active field in the calling program.
- *Drill-down browses* are more complex. They include more information and can display, filter, graph, or print data.

The field values in the browse can come from a table or a view. A *view* is a table that has selected values from one table or several joined tables.

Use Drill-Down/Lookup Maintenance (36.4.8.1) to assign drill-downs or lookups to fields that do not have a browse, to replace a browse, or to delete one.

One of the most common uses of this program is to display generalized codes associated with a field. You can also assign lookups to any field that acts as an index to a maintenance screen. You may, however, need to write your own custom browse to find and display the data.

See “Adding a Lookup” on page 19.

Most programs attached to a function with Drill-Down/Lookup Maintenance display values in a database table. But this is simply a convention. You can attach any Progress function to a field, and this program executes when the user selects Help. For example, you can attach the program `calculat.p` to field pt_avg_int to display a calculator.

Before you can use Drill-Down/Lookup Maintenance, you need to know:

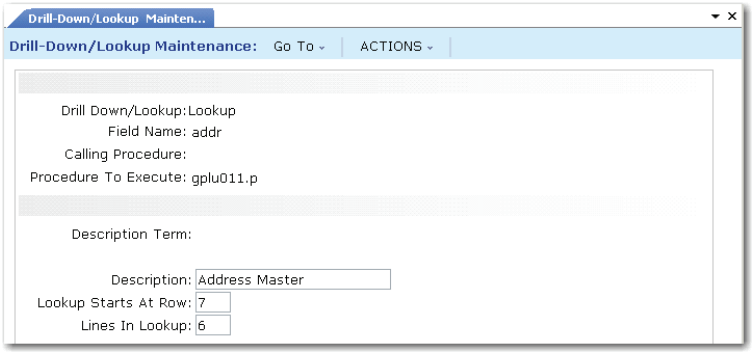
- The name of the field where you want the browse to display.
- The name of the program using the field.
- The program name of the browse to attach. If a lookup is missing for a particular field but exists for a similar one, use Lookup Browse (36.4.8.3) to determine the program that displays appropriate field values. Then use Drill-Down/Lookup Maintenance to specify the same program for the similar field.

See “Creating Browses” on page 74 for details on creating browses.

Determining the name of the program and field depends on the user interface.

- In the character interface, run the program. Note the program name in the upper left corner of the screen. Then place your cursor in the field where you want to attach the browse. Press Ctrl+F and note the field name.
- In the QAD .NET UI, find the program in the Application menu area, right-click on the program name, and select Properties. A screen displays program information, including the program name. To identify the field name, place your cursor over the field where you want to attach the browse on the program screen. The field name displays.

Fig. 3.10 Drill-Down/Lookup Maintenance (36.4.8.1)



You can assign more than one drill-down to the same field. A menu of drill-downs appears when you request the drill-down. Only one lookup can be attached to a given combination of field and program name.

You can attach browses to fields in any program, including another browse. Drill-downs can be nested. A field can call a browse that can call another browse that can call another browse, and so on.

Follow these steps to use Drill-Down/Lookup Maintenance to associate a drill-down with a field or program:

1. Select Drill Down in the Drill Down/Lookup field.
2. Enter a field name to associate with the browse in Field Name. Leave it blank to associate it with all fields.
3. Enter the program containing the field in Calling Procedure. Leave it blank to attach the browse to all programs using the specified field.

4 Enter the browse name in Procedure to Execute.

5 Optionally, enter a label term in Description Term. The long label contained in this term is displayed as the title in the browse. The default is the browse description term defined in Browse Maintenance.

See “Creating Browses” on page 74.

You access drill-downs differently in the two UIs:

- In the character UI, Select Drill Down from the Help menu or use Alt-Tab.
- In the .NET UI, click the underlined link that indicates a drill-down. When more than one drill-down is associated with a field, right-click to see a menu listing drill-down options or URLs.

Follow these steps to associate a lookup with a field:

1 Select Lookup in the Drill Down/Lookup field.

2 Enter a field name to associate with the browse in Field Name.

3 Enter the program containing the field in Calling Procedure. Leave it blank to attach the browse to all programs using the specified field.

4 Enter the browse name in Procedure to Execute.

5 Optionally, enter a description for the lookup. This description is for reference only and is not displayed in the lookup.

6 Enter the starting row and the number of lines to display in the lookup.

Wildcards in Drill-Down/Lookup Maintenance

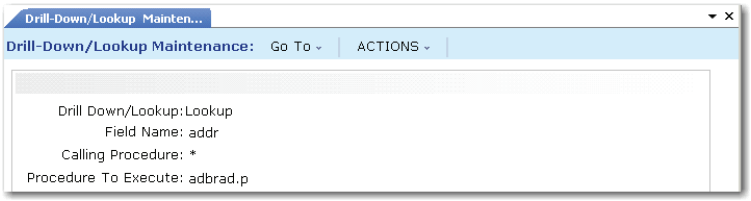
Use wildcards to attach browses to fields in multiple programs. For example, `pp*.p` attaches the drill down to the specified field in all programs starting with `pp` and ending with a `.p` extension.

The table displays the possible entries to Drill-Down/Lookup Maintenance:

Field	`ad_addr`	`ad_addr`	`ad_addr`
Calling Procedure	`*`	`so*`	`soivmt.p`
Procedure to Execute	`adbrad.p`	`adbrcs.p`	`arbrbl.p`

When you drill down on `ad_addr` in `soivmt.p`, a menu shows all three browses: `adbrad.p`, `adbrcs.p`, `arbrbl.p`. When you drill down on `ad_addr` in a program other than `soivmt.p` but beginning with the letters `so`, a menu shows two browses: `adbrad.p` and `adbrcs.p`. When you drill down on `ad_addr` anywhere else, the browse `adbrad.p` opens.

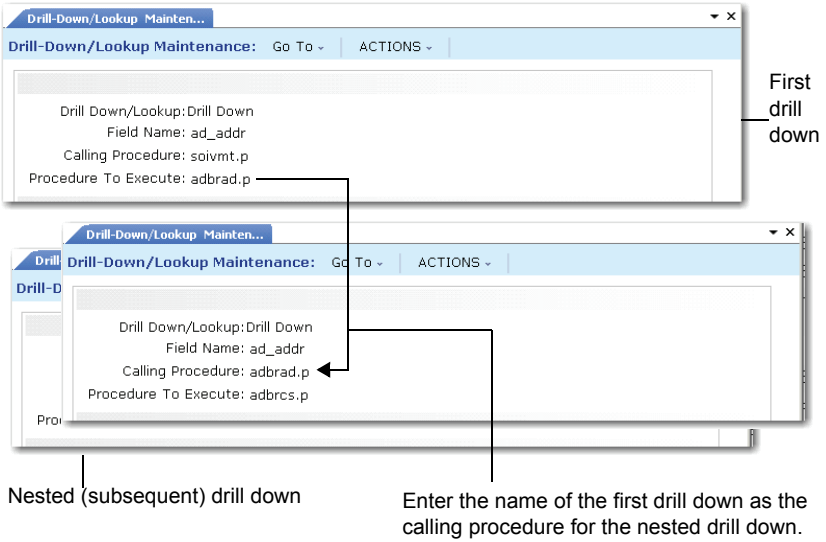
Fig. 3.11
Wildcards in Drill-Down/Lookup Maintenance



Drilling Down on Drill-Downs

You can nest drill-downs. In other words, one drill-down can call another, which can call another, and so on. After creating the first drill-down, you can assign the others to the same field. Enter the name of the first drill-down as the calling procedure for the nested drill-down.

Fig. 3.12
Nested Drill Downs



Planning for Upgrades

When you update your product version, be careful when loading flh_mstr. This table contains the records created by Drill-Down/Lookup Maintenance. If you have customized it, make sure that the new version does not overwrite your customization.

Creating Access to Other Programs

User Tool Maintenance (36.4.8.5) lets you specify programs that can be run from other programs in the .NET User Interface. This makes it easier for you to run frequently used programs.

Note The relationships you define in User Tool Maintenance do not apply to any programs in the character interface and they do not apply to browses.

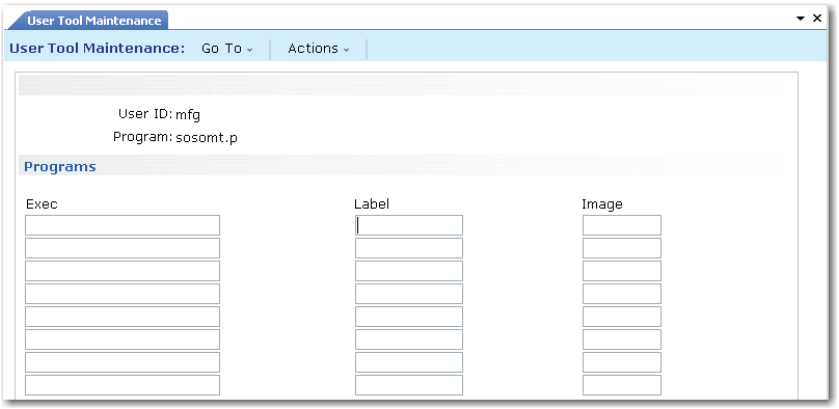
Use the Add Link command on the Go To menu from any HTML program in the .NET UI to access User Tool Maintenance. The links you define also display under the Go To menu. When you click a link, the linked program opens in a new tab.

Note You can also add links by accessing User Tool Maintenance directly from the menu. When you access it from a program in the .NET UI, the User ID and Program fields are filled in already.

User Tool Maintenance

Figure 3.13 illustrates the User Tool Maintenance screen.

Fig. 3.13
User Tool Maintenance
(36.4.8.5)



1 Enter a user ID or leave the field blank to assign the link to all users.

2 Enter the name of the program where you want the link to display. Leave Program blank to add the link to all programs that do not already have a user-specific record. Click Next to continue.

Note You can also use wildcards to specify where the links appear. Specifying pp* places the links in all programs beginning with pp.

3 In the Exec fields, enter the program names (for example, `adbr001`) for the links to launch.

4 In the Label field, specify a text string to appear in the link area. If you leave this field blank, the standard menu description from Menu System Maintenance is used.

5 Leave the Image field blank. Images do not apply in the .NET UI.

Displaying Links

You can assign programs to all users (blank user ID) or a specific user. You can also assign programs to a specific program or using wildcards. However, only one set of records displays when a user accesses a program. The system searches for the appropriate links to display in this order:

1. Specific user ID and specific program name
2. Specific user ID and program name with wildcards
3. Blank user ID and specific program name
4. Blank user ID and program name with wildcards

Managing Stored Searches

Use the Stored Search activities (36.4.8.7) to manage stored searches created by users from various component-based browses Each user can create stored searches so that they can quickly find records they want to view or update. Depending on the user's access, searches can be saved at three levels:

See *User Guide: QAD Financials A*.

- For the individual user
- For all users assigned to a role
- For all users in the system

From the system administrative point of view, you can use the maintenance functions to:

- View stored search records that have been created by users.
- Delete stored search results. This might be necessary if a user leaves the company or a role is made obsolete.

Note Exporting stored search values is not currently implemented.

The system loads search and browse defaults during installation. Factory defaults display as a type of stored search, along with the last used customized settings. Use Export Browse Defaults to export default search and display settings into a `FactoryDefaults.xml` file. The system saves the file in the same folder that contains other QAD financial business logic.

Defining Browse URL Links

Use Browse URL Maintenance (36.4.8.9) to create URL links that users can activate from .NET browses. When a browse cell contains a URL link, double-clicking it launches a new browser window and displays the intranet or Internet resource associated with the URL. You can use these URLs in two ways:

- Create links to external Web sites that users can activate from .NET browses, such as a supplier Web site associated with a supplier ID.
- Create links to other programs and pass specific data values to the programs. This lets you use browses as a means of navigating directly to maintenance programs.

You can access links to other programs only from drill-down browses, not lookups. Drill-down browses are typically available directly on the menu, but can also be associated with program fields in Drill-Down/Lookup Maintenance (36.4.8.1), see “Maintaining Drill Downs and Lookups” on page 61.

Defining URLs to External Web Pages

Use Browse URL Maintenance to create links to external URLs with information that is related to items in the browse, as in the following example.

Example You want to establish a URL link in the Purchase Order Browse from supplier ID GS10100 to the corresponding supplier’s company Web site, located at http://www.generalsupplies.com. To do this, enter the values in listed here in Browse URL Maintenance.

Table 3.3 Sample Field Entries for Browse URL Links

Field Name	Value
Browse	pobr006.p
User ID	*
Field Name	so_vend
Value	gs10100
URL	http://www.generalsupplies.com
Description	General Supplies Web Site
Primary	Selected

URLs can contain special strings that are automatically replaced by field values in the browse. Selecting a link containing this type of string automatically replaces that string with the corresponding field value in the row.

Follow these steps to define this type of special string in a URL:

1. Enter `#b#` to indicate the beginning of the string.
2. After the `#b#`, enter a field name associated with the specified browse.
3. Enter `#e#` to indicate the end of the string.

Example The Web site for one of your primary suppliers contains a catalog of items. Entering an item's identifier at this Web site accesses the catalog entry for that item, containing information such as item cost, quantity available, and ship weight. To create links from the internal supplier item numbers to their corresponding catalog entries at the supplier's Web site, create the following URL:

http://www.generalsupplies.com/catalog/#b#vp_vend_part#e#

Then, associate it with the Supplier Item column in the Supplier Item Browse. After you establish this link, selecting a supplier item number in the Supplier Item Browse automatically inserts the selected field value.

For example, selecting supplier item 10-1005 creates this URL:

http://www.generalsupplies.com/10-1005

The system then launches a Web browser to display the relevant catalog information for that item located at that URL address.

Defining Links to Other Programs

Use Browse URL Maintenance to create links to other system programs, as in the following example.

You can set up links in an item browse to directly access Item Master Maintenance (1.4.1), passing the current item number to the maintenance program, and executing the Next command any number of times. When a user clicks the link, Item Master Maintenance displays in a separate tab.

Multiple columns of data in a browse can contain links so that you can access maintenance programs for any data related to a record. However, data for only one field can be passed to each program.

To support this kind of URL link, use the run_html setting to indicate that you want to build a URL for system programs. The string must include the beginning and ending indicators required for other strings in URLs:

1. Enter `#b#` to indicate the beginning of the string.
2. Enter `#e#` to indicate the end of the string.

Then specify values that determine:

- The name of the program to be executed when a user clicks the link
- The field in the designated program to which you want to supply a value
- The value to be passed to the specified field
- The number of times the Next command should be executed in order to reach the field

To make it easy to build the URL, leave the URL and URL Script fields blank and click Next to display a pop-up that prompts you for the values required to build the URL. In this case, the system builds the URL including the run_html setting using the values you supply.

Table 3.4 Sample Field Entries for Browse Program Links

Field Name	Value
Browse	sobr009.p
User ID	*
Field Name	sod_part
Value	*
URL	Leave Blank
Description	Link to Item Master Maint.
Program Name	ppptmt
Field	pt_part
Value	sod_part
Index	2

The URL that the system builds based on these input values looks like the following example:

#b#run_html#e#?id=ppptmt.p&f=pt_part&v=#b#sod_part#e#&x=2

When the user clicks item 01053 in the `sobr009.p` browse, Item Master Maintenance displays with 01053 entered in the Item Number field and the active cursor focus is in the Name field below it.

Using Browse URL Maintenance

The following figure illustrates Browse URL Maintenance.

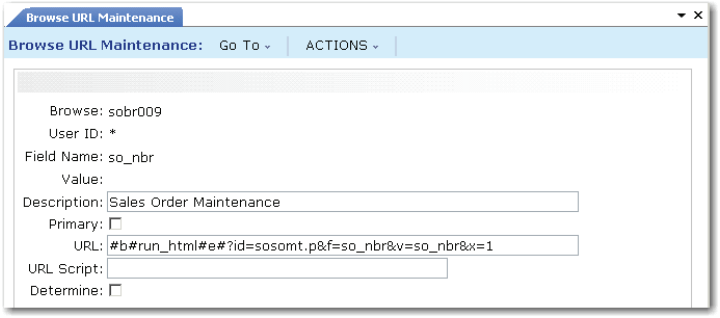


Fig. 3.14 Browse URL Maintenance (36.4.8.9)

Use the following field descriptions to configure Browse URL Maintenance settings for your environment:

Browse. Enter the name of the browse program to contain the specified URL link.

Entering an asterisk (*) in this field allows the specified URL to be associated with any browse in the system.

Example To associate a specific URL with the sales order number column in all .NET browses, enter an asterisk in this field and specify the sales order column's corresponding field name (so_nbr) in Field.

User ID. Enter a user ID to associate with the specified URL link. To associate all users in the system with that link, enter an asterisk (*) in this field.

Field Name. Designate the browse column in which you want to establish a URL link by entering the database field name associated with that column. The lookup associated with this field displays fields from the previously entered browse.

This field cannot be left blank.

Value. Specifying a value in this field associates the designated URL with every browse cell that contains that value and belongs to the browse column indicated in Field Name. Enter an asterisk (*) to associate the URL with every cell in the column.

Description. Optionally enter a description to display when this URL is selected.

Primary. Select to indicate that the specified URL is the primary URL for a cell.

When you right-click a browse cell containing multiple URLs, a list displays for selection. The primary link displays at the top of the list in bold font and is the default link for the cell.

Note The value of Primary has effect only if no drill-downs are also associated with the cell. Drill-downs always take precedence over URLs and are listed first when you right-click.

This value applies to defined (non-scripted) URLs only.

URL. Specify a URL referring to an Internet or intranet location (maximum 132 characters). Leave blank to create a link to another program.

URL Script. To associate the specified user, browse, column, field value, or combination of these with a custom URL script, enter the full path to the directory containing the custom script. You cannot specify both a URL and a URL script. Scripts should be based on the supplied template `urltempl.p`, located in the source code directory, `/src/urltempl.p`.

Determine. Select this check box to have the system run the specified custom URL script upon selection of the associated cell or column to determine whether that cell or column has an associated URL.

When this field is clear, the script is not run and the designated column or cell is defaulted to having a URL.

HTTP Parameters Pop-Up

This pop-up displays when both URL and URL Script are blank and lets you specify a set of values used to access another program.

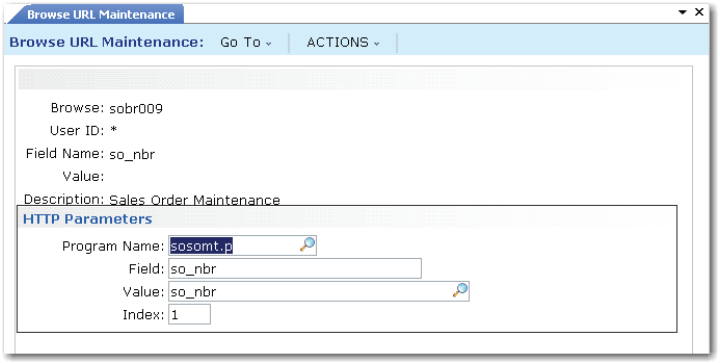


Fig. 3.15
Browse URL Maintenance, HTTP Parameters

Program Name. Enter the name of the program to launch when a user clicks this link. You can include or omit the `.p` extension. For example, specify `adcsmt` or `adcsmt.p`.

Field. Enter the field in the program that you want to pass a value to from the browse; for example, enter cm_addr to pass a value to the Customer Address field in Customer Maintenance. Only one field can be specified.

Value. Enter the field in the browse that contains the value that you want to pass to the named field. Use the beginning and ending delimiters for this value. For example, to pass the value of the sod_part field in the Sales Order Browse (7.1.2) to the pt_part field in Item Master Maintenance, enter `#b#sod_part#e#`.

Index. Enter the number of times you want the system to execute a Next command when it invokes the program specified for Program Name. To access the first frame in a maintenance program, set this value to 1. If you set this to more than 1, all validations are executed before processing the Next command. If appropriate values do not exist for all required fields, an error is generated when you click the link.

Creating Browses

Use Browse Maintenance (36.4.8.13) to create custom browses or modify system-supplied browses.

When you create a browse, it is saved in your working directory as a source-code file. The source-code name is the first two characters of the name you entered, then the letters `br` or `lu` (depending on whether you selected power or look up), then any remaining numbers from the name you specified, then the extension `.p`.

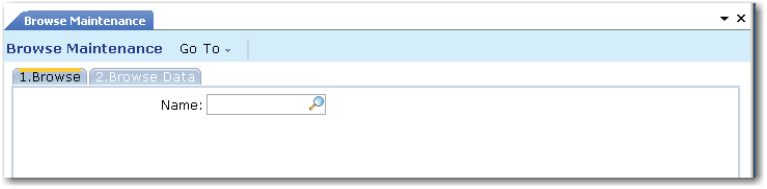
Example You create a power browse and name it `ap010`; the system names the code `apbr010.p`. If you selected both power and lookup browses, the system generates two source-code files: `apbr010.p` and `aplu010.p`.

Although you do not need to compile the source code of the browse, you should for better performance. If other users on your network want to use your browse, you must compile it and move it to the network directory. Use the Progress editor to compile the browse.

Note You can access the Progress editor only if your PROPATH is correctly set up to access source files.

Browse Maintenance has some special features that are not available in the character UI and that are not like other HTML programs. This section illustrates the program in the .NET UI.

Fig. 3.16
Browse Maintenance (36.4.8.13)



To create or modify a browse:

1 Select or enter a name for the browse. To name the browse, enter two letters and click Next. Use the existing QAD module mnemonics or make up your own. The system gives the browse a name that increments by one the number in the file name of the last browse created.

2 Click Next or click the second tab, Browse Data, to continue.

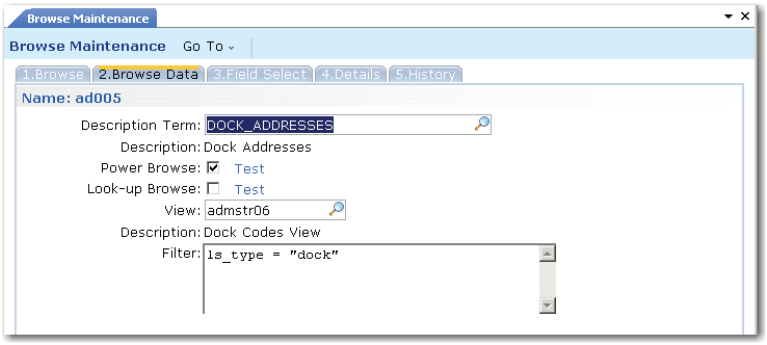


Fig. 3.17 Browse Maintenance, Browse Data

3 Enter a label term in Description Term. The long label contained in this term is displayed as the title in the browse window.

4 Indicate whether this is a power browse, lookup browse, or both.

5 In View, enter the name of an existing view or a primary table whose data the browse displays. You can see only those views you have access to. If a view exists for a table and the view name is the same as the table name, you have access to only those fields that are available in the view.

See “Creating Views” on page 78.

Note The view name you enter in View must already be defined in View Maintenance, or you must enter a primary table name.

6 In the Filter field, type the selection criteria (optional) to limit the browse’s search to records that meet a certain condition. Do not put a period (.) at the end of the criteria, because the system adds a `no-lock no-error` statement to the criteria.

The Preview function, accessed with a button in the character user interface, is accessed by clicking one of the Test links next to the Lookup Browse and Power Browse check boxes. This displays the currently edited browse in a new tab. For existing browses, you can preview the browse before changing it.

Use the Field Select tab to select fields to be included in the browse and specify the display order.

Fig. 3.18
Browse Maintenance, Field Select

7 Fields from the view or primary table entered in the Browse Data frame display in Available Fields. Include up to 20 fields in your new browse.

- In the .NET UI, move fields from one list to another by clicking the field to select it and clicking the arrow that indicates the appropriate direction. Multiple fields can be selected by holding down the Shift key and clicking. Use the Move Up and Move Down buttons to arrange the fields in the Selected Fields list.
- In character mode, select a field to include in your browse by using the Up and Down keys to locate it and then press Enter. Multiple fields can be selected. Use the Tab key to choose the Add, Field Help, or Done buttons or to navigate between the Available Fields list and the Selected Fields list.

8 In the .NET UI, to specify sort columns—for look-up browses only—select one or more fields in the Selected column and click the right arrow to move them to the Sort Order column. You can use the up and down arrows to reorder the fields in the Sort Order list. The look-up browse sorts the records on the first column you enter in the Sort Order field.

In the character UI, use the Sort Columns field to enter the columns you want to have available for sorting. Enter the columns as a comma-delimited list of up to seven numbers. The first field name in the Selected Fields list is column 1, the second is column 2, and so on.

9 When you have arranged the fields in the order you want, click Next to display the Details tab.

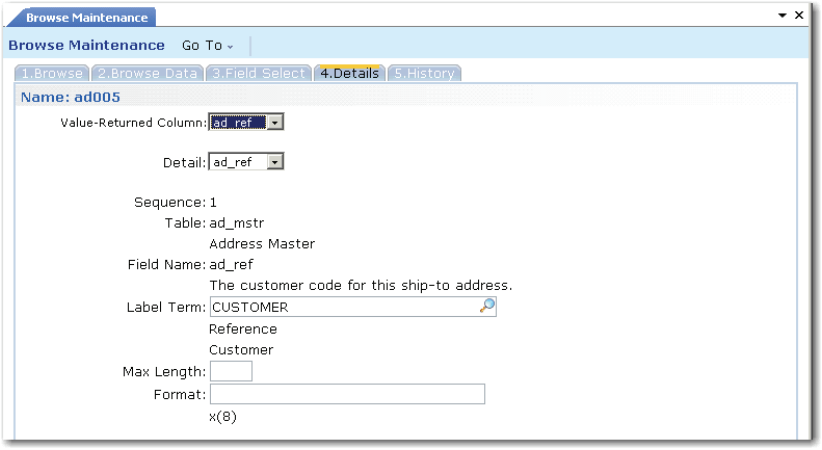


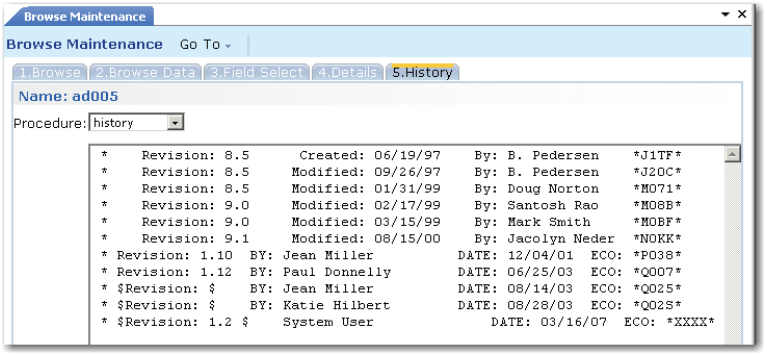
Fig. 3.19 Browse Maintenance, Details

10 Enter the column number to take the field values from in Value-Returned Column (optional). The default is the first column of the browse.

11 To view the details related to a specific field, select it from the drop-down list in the Detail field.

12 If needed, you can change the default field label and format. To control the display length of a label, enter a Max Length value. Click Back if you want to view or modify data for other fields. Otherwise, click Next to display the browse history.

Fig. 3.20
Browse Maintenance, History



13 The program automatically creates a revision history line containing a revision number, the user name (or logon ID), and current date. You can modify this as needed. The revision history is also saved in the source code.

Note The Procedure field offers other choices; only History is currently implemented.

14 Click Next to generate the browse.

Creating Views

A view is a display of some or all of the fields from one or more tables. You join two or more tables for a view by specifying the relationships between fields in different tables and choosing the type of join to use.

Views are used in browses, which display the fields gathered using views. By choosing which fields to include or exclude in a view, you control which fields are available for a browse to display. By putting security on the view, you can allow users to modify browses, knowing that they can access only those fields that you have authorized.

Use View Maintenance (36.4.8.18) to create or modify views.

Using Progress Syntax

You use some Progress syntax in creating or modifying views. You must also understand database table and field relationships.

To create or modify a view:

1. Select the table or tables to include in the view.
2. For sequences after the first, specify the type of join to use: inner or outer.
3. Join the tables using Progress logic.
4. Select fields from the tables.
5. Save the view.

Figure 3.21 illustrates how to create a view of selected fields from one table.

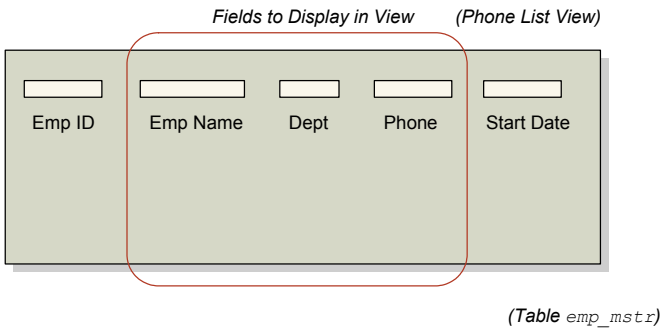
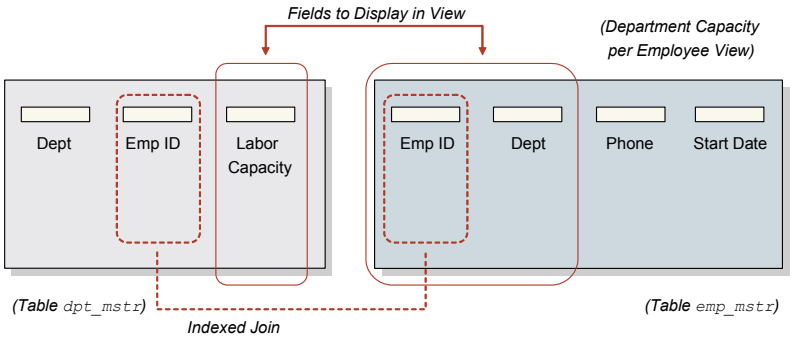


Fig. 3.21
Creating a View from One Table

Figure 3.22 illustrates how to create a view of selected fields from two tables.

Fig. 3.22
Creating a View by Joining Two Tables



Using Join Types

When a view includes data from more than one table, you can choose from two types of joins when creating a view:

An inner join returns the records selected for the first table combined with related records selected from the second table. If a record does not exist in the second table, no records are returned. Only related records selected from both sides of the relationship display in the view.

An outer join returns the records found by an inner join. However, in addition, for each value in the first table, it returns unknown values from the second table when no related record is found. As a result, all matching records from the first table are preserved for unmatched records in the second table.

The default join type is inner. Using the outer join can give you more flexibility in displaying information.

Example An inner join between customers and sales orders displays only customers with sales orders. An outer join includes all customers, even those who do not have orders.

Using View Maintenance

Figure 3.23 illustrates View Maintenance (36.4.8.18).

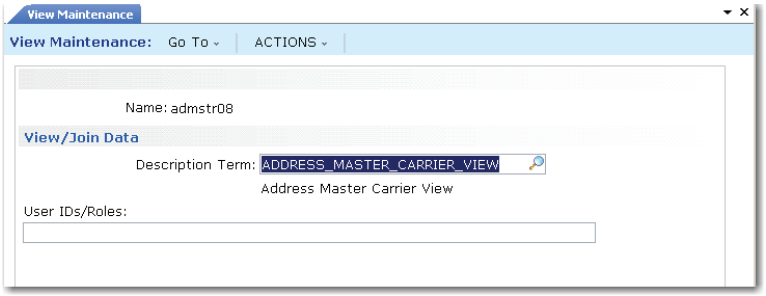


Fig. 3.23
View Maintenance (36.4.8.18)

1 Select or enter a view name.

2 Enter a label term in Description Term. The long label contained in this term is displayed as the view label.

3 In User IDs/Roles, enter one or more user IDs or roles to limit user access to the view (optional). Enter multiple user IDs or roles by separating them with commas.

4 Click Next to continue.

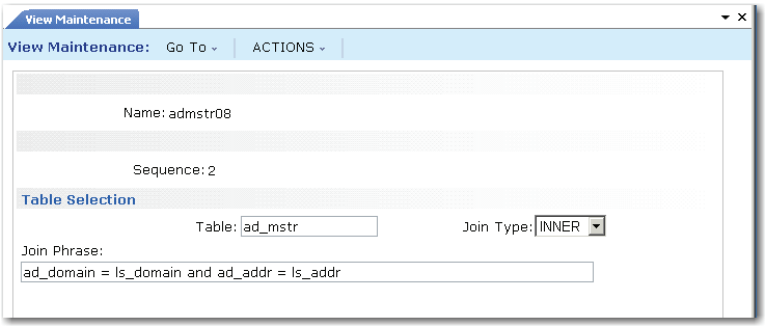


Fig. 3.24
View Maintenance, Table Selection

5 The number you enter in Sequence controls the order in which the table defined in Table is joined to the view.

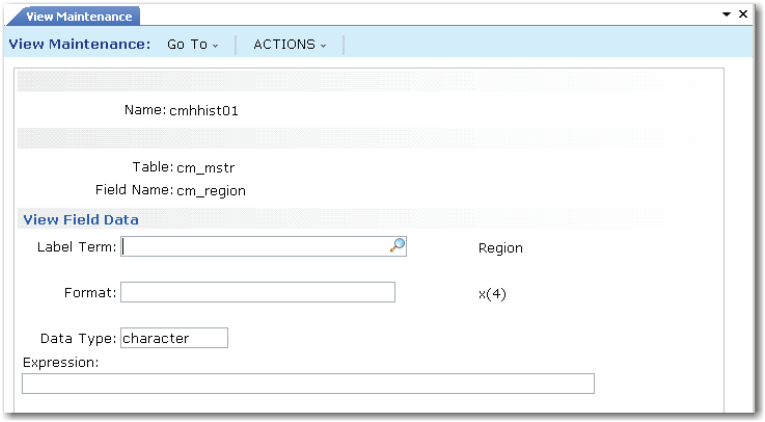
6 Enter a table name.

7 If the sequence is not 1, specify the type of join, either inner or outer. The Join Type field is only enabled when the sequence number is greater than 1.

8 Enter or edit the phrase to join the tables. Use proper Progress syntax. Do not include a Where verb. Join phrases express the field relationships between tables (see Figure 3.22). For a faster display of fields, use indexed fields in the Join Phrase.

9 Click Back.

Fig. 3.25
View Maintenance, View Field Data



10 In Field Name, enter a field from one of the tables in the view or enter a local variable. When entering a local variable, name it `local-varnn`, where `nn` is a number incremented by one from the last defined variable.

For example, you see from the lookup browse that the last local variable was `local-var05`; you name your local variable `local-var06`. Use local variables when you want to return a value resulting from an operation on two fields; for example, the quantity required minus the quantity open. Define the operation in Expression.

11 If you entered a local variable in Field Name, enter its Label Term, Format, and Data Type.

Note Search for a label term by entering a portion of a label, then use Next/Previous to display available records.

12 If Field Name is a local variable, you can enter Progress syntax in Expression to define the local variable. Valid expressions include:

- `field1 + field2` (computation, where `field1` and `field2` are fields within the record)
- >, <, >= (operands that perform comparisons)
- Progress functions, such as `substring (field1,1,4)` or `round (field1,1)`

Note Incorrect syntax terminates your session if you attempt to use the view.

Defining User Menu and Function Keys

User menu and function keys are available in the character UI. In the .NET UI, users set up personalized menus with the Favorites functionality.

Assigning function keys to frequently used menu items is another way to execute programs quickly in the character user interface. Keys can be established for all users or individually customized. Up to 999 function keys can be defined. In addition, you can change the standard label for a menu item to customize menu labels for each user.

Note Function keys apply only to the character interface.

Example A user entering a sales order may need to check on the available-to-promise (ATP) quantities for an item before indicating a due date. By setting up a function key for the Master Schedule Summary Inquiry (22.18), the order clerk can review an item's ATP quantity without leaving Sales Order Maintenance (7.1.1).

Note Do not use function keys or the function menu to access a maintenance screen in the character environment. Progress only completes transactions initiated with function keys after the initial transaction is completed. If, for example, you are in sales orders, you start an order, then perform an inventory transaction using a function key, and then cancel the sales order, the inventory transaction is also canceled.

Access programs associated with a function key by selecting that function key. Function keys F1 through F12 are reserved for system use, so the assigned key must be F13 or higher. Since many keyboards do not handle that number of function keys, this option is used less frequently.

User Menu

Access the User Menu by pressing F6. A list of menu items set up for your user ID appears. Choose the one you want by highlighting it and pressing Enter or F1. Press Tab to sort the list by menu number or function name. Press F4 to display the user menu items defined without a user ID.

Note There is no relationship between the order of items on the User Menu and the function key assigned, and the function key is not shown. Menus sort lexically, so that 13 appears before 2 if you are in the Menu Selection column.

Different environments have different function key uses and limitations. Set up your system according to your environment. For example, if your system is limited to only 12 function keys, do not attempt to use the function keys as a quick method to launch programs. Instead, use the User Menu.

Executing Programs in Sequence

In the character interface, you can make several programs execute in sequence by assigning them to the same function key and giving each a different sequence number. When you press that function key, the first function in the sequence executes. When that function is finished, the next one in sequence is called automatically.

Important All transactions in the sequence must be completed before data is updated in the database.

User Function Maintenance

Set up user menus and function keys in User Function Maintenance. Each selection on the user menu should have a different function key reference, from 13 to 40, and a zero or blank sequence number. The function key reference must be 13 or greater, even if your keyboard supports fewer function keys or you plan to access selections through the User Menu.

Note To set up function keys, terminals must be compatible with the Progress protermcap file.

Using Field and Program Help

The system provides two types of online help: program (also called procedure) and field help. Program help explains what the current function or program you are working within does. Field help describes particular fields.

You can view these help records in either HTML or character format. The content of the HTML and character help files is identical. You can add your own information to the help; it displays under a heading of User Help in both character and .NET UI.

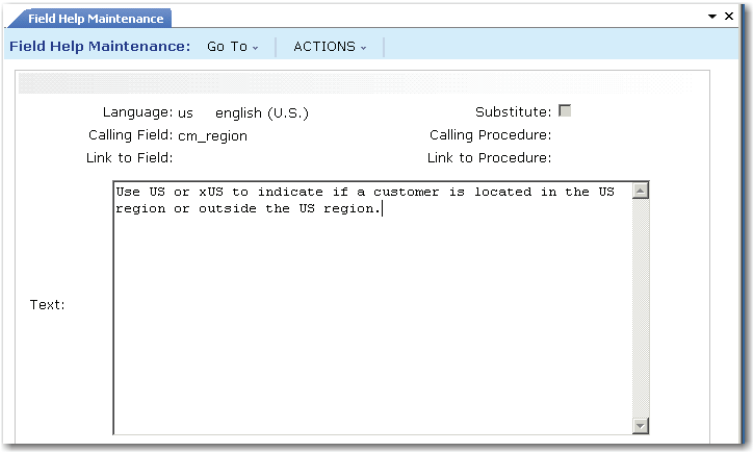
Note QAD .NET UI displays the character help data in an HTML format. Any changes you make to character help are also visible in the QAD .NET UI.

In the QAD .NET UI, view field and program help by pressing F1. In the character interface, view field help by pressing F2 with the cursor in the field. Press F2 a second time and program help displays. No keyword searches or hypertext links are available in character help.

Adding User Help

Use Field Help Maintenance (36.4 13.1) to add to the character-format help delivered with your system.

Fig. 3.26
Field Help Maintenance (36.4.13.1)



Custom text entered in Field Help Maintenance appears at the top of the help record.

Printing Help

You can print out portions of the field and program help to supplement your user guide set. Printed field help is available through Field Help Report (36.4.13.2). The Procedure Help Report (36.4.13.4) prints program help in alphanumeric ranges by program name.

The Field Help Book Report (36.4.13.3) enables you to print a book containing all field help. Choose units as small as one field and as large as an entire module.

Local Vars. Clear this option to exclude local variables. These are field names created within a program, not drawn from the data dictionary. In reports, the From and To fields are often local variables. Usually, help for local variables is not as significant as database fields.

Update Only. Select this option to limit output to fields that can be changed.

Where-Used, Maximum. Disable the Where-Used option to keep the system from printing a where-used list after each help item. Some database fields are used throughout the database, and a complete where-used list can be very long. If selected, limit the length of the where-used list by entering a value in Maximum.

Setting User Telnet Options

As part of system installation, telnet setup scripts are configured for use by the Connection Manager for managing HTML programs running under the .NET UI. Create additional telnet scripts for running terminal sessions within the QAD .NET UI with User Option Telnet Maintenance (36.4.14).

The QAD .NET UI also reads other settings in this program, including the port value defined in this program to determine which port to use to connect to the server for terminal programs.

Configuring the settings in User Option Telnet Maintenance consists of the following tasks:

- Specifying telnet server settings
- Defining the login sequence
- Configuring the telnet connection settings
- Verifying the script login sequence

Configuring Telnet Server Settings

Figure 3.27 depicts the User Option Telnet Maintenance (36.4.14).

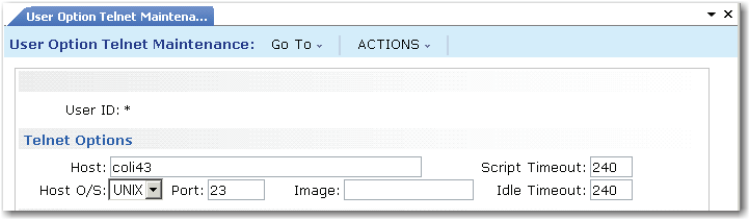


Fig. 3.27
User Option Telnet Maintenance, Telnet Options (36.4.1.4)

Use the following instructions to configure telnet server settings:

1 In User Option Telnet Maintenance, create a record that applies to all users by entering an asterisk (*) in the User ID field; then click Next.

2 Enter values in the following fields

Host. Enter the fully qualified machine name or IP address of the telnet server. The script uses this information to establish the telnet connection.

Host O/S. Enter `UNIX` for UNIX systems. Enter `NT` for Windows systems.

Port. Enter the port number for the telnet server. The default value is 23. This is the value you would normally use, unless you are using SSH under the .NET UI. In this case, the port value is 22.

Image. Leave this field blank; it is not currently used.

Script Timeout. Enter the number of seconds (1–999) the system waits for the telnet login script to execute. If this value is exceeded, a time-out message displays and the session closes.

Idle Timeout. Enter the number of seconds (1–999) the system waits after a telnet session begins for a program to execute.

Note Idle timeout is not used in the QAD .NET UI.

Define the Login Sequence Script Lines

For users to be able to log in to and begin a telnet session on the telnet server, you must provide the sequence of telnet server login prompts and responses. The last value in the sequence specifies the script you created during system installation.

Terminal mode supports either operating-system level or script user credentials based on the setting of `TerminalAuthentication` in *`qaduiConfig`*`.xml`, located in:

`TomcatInstallDir``/webapps/qadhome/client/configs`

- When this is set to ShellUser (the default), the user ID and password used to login to the .NET UI are used for the terminal login, regardless of what is set in this program.

- When this is set to ScriptUser, the user ID and password defined in User Option Telnet Maintenance is used.

When defining paths for scripts used in the .NET UI, avoid using relative paths since each user's access may be different.

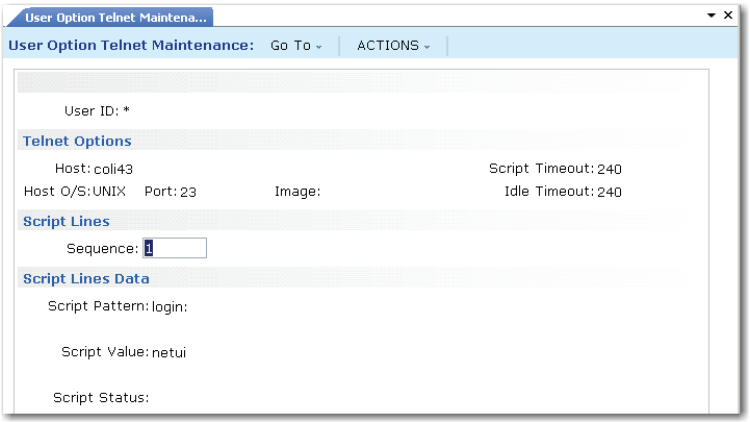


Fig. 3.28
User Option Telnet Maintenance, Login Script Line

1 Specify the telnet login sequence number in the Script Lines frame. For each telnet command, enter a sequence number beginning with 1, and click Next. In the next frame, enter the following:

Script Pattern. Enter the prompt generated by the telnet server when a telnet login occurs. The values in this field must be identical to the prompts the telnet server displays when users log in.

Script Value. Enter the response to the telnet login prompt defined in Script Pattern.

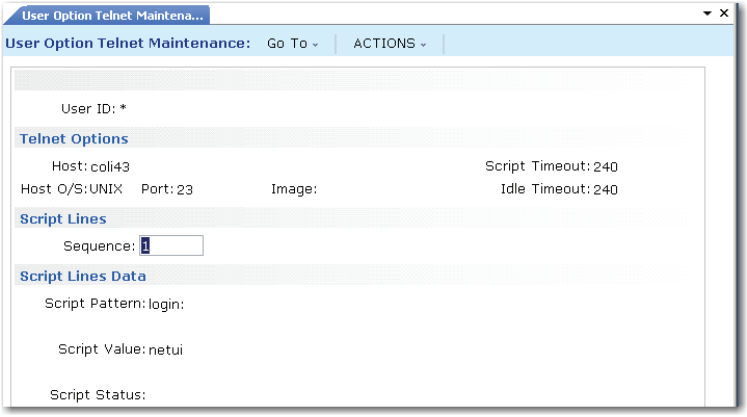
Script Status. Optionally enter a description of the prompt and response; for example, Logging In.

If you have tracing enabled and the Java console is displayed, the description in the Script Status field displays in the Java console on the client when an error occurs during the execution of the prompt and response. You can use these descriptions as an aid in troubleshooting telnet session issues.

Note Script sequence 2 has special validation for suppressing password display. When you enter a password as a script value, only blanks display. When you click Next at the end of the sequence, you are asked to confirm the password.

2 Click Next after entering the sequence values. You return to the Sequence field to enter the next sequence number and values.

Fig. 3.29
User Option Telnet Maintenance, Second Login Script Line



3 After entering the final sequence, click Next to return to the Sequence field. Then click Back to move to the Telnet Connections fields.

Configure Telnet Connection Settings

Once you configure and verify your telnet login sequence, access the Telnet Connections frame and specify telnet connection settings. These settings define the maximum and minimum number of telnet connections available to the associated user.

Recommended settings are 10 or more for Maximum; 1 for Minimum.

Fig. 3.30
User Option Telnet Maintenance, Telnet Connections

Maximum. This value specifies:

- The maximum number of concurrent embedded telnet screen connections this user can have open per session.

- The maximum number of running HTML programs allowed for the user. If a specific record does not exist for a user with this setting defined, that user can continue opening programs until the maximum number of sessions allowed for the entire pool is reached.

Valid values are:

- Unlimited: The associated user can have an unlimited number of concurrent telnet connections open.
- Disabled: The associated user cannot log in through .NET UI. Until you create a login script to initiate telnet sessions for this user, you cannot set this field to any value other than Disabled.
- Any numerical value between 1 and 99.

Note Setting this to 0 is the same as disabled.

This setting applies separately to HTML telnet sessions and terminal sessions in the .NET UI. For example, if Maximum Telnet Settings is 5, a user can have 5 HTML maintenance programs running and 5 telnet programs running in one session before an error displays.

Minimum. Enter a value between 0 and 9 to indicate the minimum number of telnet connections to be available to the associated user at all times.

Set this value to the number of telnet programs the user is likely to run simultaneously. Specifying a value here can dramatically reduce the wait time for these programs to display. However, setting this value too high depletes system resources.

QAD recommends that you set Minimum to 0 (zero) for most users, including the generic user—defined with an asterisk (*). If users access terminal maintenance programs extensively, set Minimum to 2.

Verify the Login Sequence

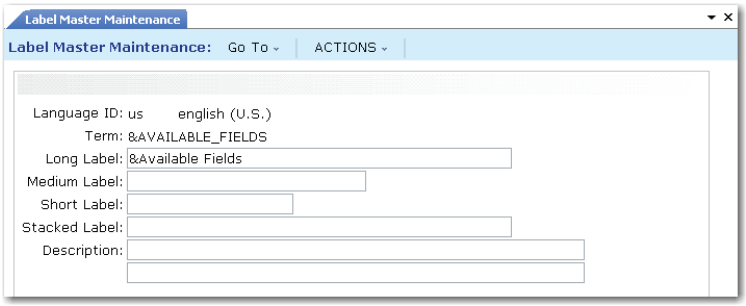
To verify the login sequence, from a remote machine attempt to log in to the telnet server using the login sequence you configured in the system. Login is successful if the system displays a blank telnet screen after the telnet connection script is launched.

Modifying Labels

The system dynamically reads the label master table to determine the appropriate labels to display on screens and reports. For the system to display labels from the label master, Translate Frames must be selected in Label Control (36.4.17.24). Otherwise, screens and reports display field labels statically from the source code.

You can modify how labels display in Label Master Maintenance (36.4.17.1). You may want to modify labels in order to meet specific company needs or to improve definitions of non-English labels.

Fig. 3.31
Label Master Maintenance (36.4.17.1)



The system validates the language code and accesses the *term*. The term is the key that links labels to fields, allowing the system to determine which labels to display. The term remains the same regardless of the language selected.

Terms display in all uppercase with underscores; for example, CALCULATE_DUE_DATE is the term for Calculate Due Date when the language code is US (American English).

Use Label Detail Maintenance (36.4.17.5) to assign terms and labels defined in Label Master Maintenance to fields generically or to fields in specified programs.

Warning Because terms can be assigned to fields accessed by many programs, label modifications and new term assignments should be made with extreme caution.

Building an E-Mail System Interface

Some functions can be configured to send e-mail messages to designated users. For example, optional e-mail messaging is used in System Security, Product Change Control, Supplier Performance, and the Global Requisition System. Additionally, you can have the system send report output in an e-mail message and you can send links to programs using e-mail.

Note Certain component activities also send preconfigured e-mail. This is discussed in “Configuring E-Mail Notification for Components” on page 97.

To take advantage of this feature, the e-mail system must be defined and addresses specified. The e-mail interface is built around an operating-system command that communicates with the user’s e-mail system. This command tells the e-mail system how to construct and address messages.

Setting Up E-Mail System Interfaces

There are two ways to set up e-mail:

- Use text e-mails.
- Convert documents to HTML and e-mail them as mail attachments.

The following topics discuss these methods.

Using Text E-Mail

With text e-mail, you cannot inform e-mail clients, such as MS Outlook or Lotus Notes, in which code page the data resides. So, for Unicode or non-English environments, the e-mail client determines the code page of the text. In most cases, if the text does not display correctly, you can right-click within the e-mail to access the code page; for example, in MS Outlook, you can right click to select Other Actions, then Encoding, then select the character set to access. In Lotus Notes, you can right click to select Encoding, then Other, then select the character set.

E-Mailing Attachments

You can convert the text to HTML, then convert to UTF-8 to make an attachment. When you open the e-mail attachment in your Web browser, the attachment displays correctly.

Note The UTF-8 code page is part of the Unicode standard. See “Using Multiple Languages” on page 45 for details.

To use HTML e-mail, you must use an e-mail command that supports attachments, like the command for attachments for the Mutt e-mail client. Mutt is a free, text-based program for reading electronic mail under the UNIX or Linux operating systems. Mutt provides a `-a` option for attachments.

If you want HTML e-mails as an attachment, you must add the `-a` option to the Mutt configuration file. Once the command is added to the configuration file, you can send text e-mails with an attachment by pressing the less than (<) key.

E-Mail Definition Maintenance

Set up a command line in E-Mail Definition Maintenance (36.4.20) for each system you want to access. Then, in User Maintenance (36.3.1), specify an e-mail definition and address for each user. The system uses the address as the sender of e-mail messages.

Be sure that an output device is defined in Printer Setup Maintenance (36.13.2) that has Destination Type set to Email. This is described in “Setting Up Printers” on page 104. When you select the associated device in the Output field in programs throughout the system, the resulting report is sent to specified e-mail addresses.

Before you implement E-Mail Definition Maintenance (36.4.20), refer to the e-mail application documentation or consult with your e-mail system administrator to determine if the application you are using provides an operating-system command interface. If it does not, various shareware products provide e-mail command-line interfaces.

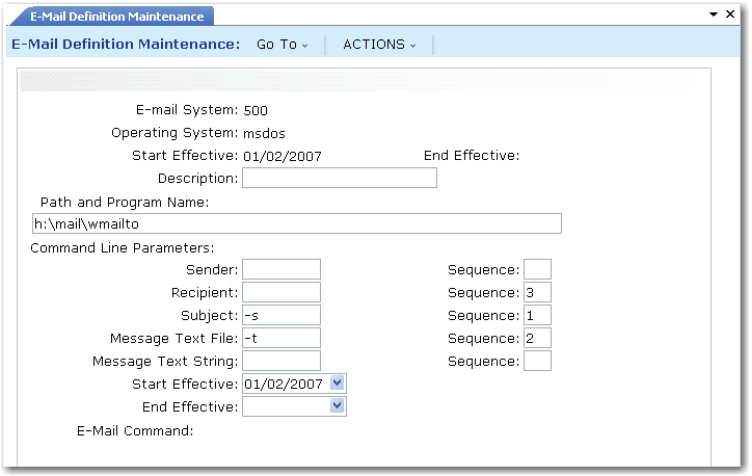


Fig. 3.32
E-Mail Definition Maintenance (36.4.20)

E-Mail System. Enter an alphanumeric code that identifies an e-mail system your company uses. This can be a number or a shortened version of the application name. You can use the same code for more than one record to give users access to multiple systems. For example, you can define both a UNIX system and a Windows system with the same code so that a user can log on to either system with the same user ID.

Operating System. Enter the name of the operating system on the user's computer. This is not necessarily the same operating system as the computer where the QAD databases reside. Valid values are UNIX and WIN32.

Start Effective. Optionally enter the first date this system is available for use.

Description. Enter a brief description of this system.

Path and Program Name. Enter the complete path to the executable e-mail application file; for instance:

```
F:\apps\shared\email\blat.exe
```

End Effective. Enter the last date this system is available for use. This is an optional field.

E-Mail Command Parameters

Command line parameter fields can store parameters or arguments to identify the type of data being passed to the command. The parameter is a prefix, which is followed by the type of data. The UNIX `mailx` command, for instance, requires that the subject of the message have a `-s` prefix, as in the following example:

```
mailx -s "test message"
```

E-Mail Definition Maintenance defines four parameters: Sender, Recipient, Subject, and Message Text File (or Message Text String). Use the message parameters required by your e-mail system. You can use either a text file or string or both, as needed.

The Sequence fields control the order in which the Sender, Recipient, Subject, and Message Text parameters appear in the command line. Some e-mail systems require these parameters in a specific order. If your system does not use one of the parameters, leaving both the Parameter and Sequence fields blank omits that parameter from the command line.

If you enter a parameter without a sequence, the parameter is not included on the command line. If you enter a sequence without a parameter, the system skips this parameter and creates the command.

The E-Mail Command field displays the system-built Path and Program Name, Parameters, and Sequence.

When you complete the setup for your e-mail system, you are prompted to send a test message. The default addressee is your log-on user ID. If you have not yet entered your e-mail address in User Maintenance, the system prompts you for an address.

Customizing Appearance of HTML E-Mail

If you are receiving e-mail in the form of standard MIME attachments, you can customize the appearance of HTML-based messages by modifying the style tag in the `email.css` cascading style sheet. This file is located in the *`InstallDir`*`\src` directory. You should move the file so that it can be found in the PROPATH.

You can modify color and font by updating the following values:

```
<style>
    BODY { margin: 0; COLOR: #000000; FONT-FAMILY: Courier,
    Arial, Geneva, sans-serif; BACKGROUND-COLOR: #FFFFFE }
    PRE { COLOR: #000000; FONT-FAMILY: Courier, Arial,
    Geneva, sans-serif; BACKGROUND-COLOR: #ffffee }
</style>
```

Configuring E-Mail Notification for Components

You can configure the system to send notification e-mails to recipients with the relevant roles when customer, supplier, employee, or end-user records are created.

Note This same e-mail configuration is used for sending work items to recipients if you are using the optional workflow. See “Configuring Workflow” on page 174 for details about enabling workflow.

The system notifies users of an event when they have the appropriate role and belong to the same domain as the item being created. In the case of supplier, customer and end users components, this is extended to include all users for each domain associated with the shared set referenced by the component. A user configured to receive an e-mail notification receives an e-mail for each domain in which the component is replicated.

Users receive separate e-mail notifications for each event for which they have the relevant role. For example, if two employee records are created, users with an employee notification role receive two separate e-mails—one for each of the new records.

Users who are to receive notifications must belong to the roles listed in the following table.

Table 3.5 Events and Corresponding Roles

Event	Role
Creation of new employee record	EmployeeNotify
Creation of new supplier record	SupplierNotify
Creation of new customer record	CustomerNotify
Creation of new end-user record	EndUserNotify

To use e-mail notification, a running SMTP (e-mail) server must exist. The location of the SMTP server is specified in the `server.xml` file. A sample configuration is shown below:

```
[SMTP]
SmtpServer=smtp.qad.com
SmtpPort=25
SmtpSender=financials@company.com
```

It is not necessary to specify the values for SmtpPort and SmtpSender. The SMTP port defaults to 25 (standard port for SMTP service) and the SMTP sender defaults to mfgpro@qad.com (this is primarily for having a default sender for development environments). Generated e-mail notifications are sent as though they originate from the specified SMTP sender. The e-mail address specified for the sender need not be a valid e-mail address.

If the e-mail (SMTP) server used to process the e-mail notifications is not available, the e-mails cannot be sent and the system returns a notification message to the user creating the component.

The system uses the e-mail address specified in User Maintenance (36.3.1) for the users that are members of the role to receive notification. If this address is invalid, the user does not receive any notification e-mails and no error is generated.

If the e-mail server used for notification is not available, the e-mails are not sent and are not queued for later delivery. In this case, a message is returned to the user creating the component to make them aware of the problem.

Using Advanced Reporting Tools

See *User Guide: QAD Business Intelligence* for detailed information on these programs.

Use programs on the Report Setup Menu (36.4.21) to support advanced reports and dashboards designed using the Cognos reporting tool. Additionally, several QAD-designed dashboards are available with QAD Business Intelligence 2.5 and higher.

Dashboards add an interactive element to reports. They let you:

- Drill up and down to see higher and lower levels of detail.
- Include multiple charts derived from different data sources in a single report.

Important Although the setup menu is also available in the character UI, you can only view the resulting reports and dashboards through the QAD .NET UI.

QAD-Provided Dashboards

If you have purchased QAD Business Intelligence 2.5 or higher and the appropriate supporting elements, you can implement several QAD-provided dashboards. See *User Guide: QAD Business Intelligence* for detailed requirements and procedures.

Custom Reports and Dashboards

You can implement custom reports and dashboards without using QAD Business Intelligence, as long as you have installed the following components:

- QAD .NET User Interface
- QAD ReportNet Bundle, delivered with the QAD .NET UI
- Cognos 8.2

Use the following workflow to implement custom reports and dashboards.

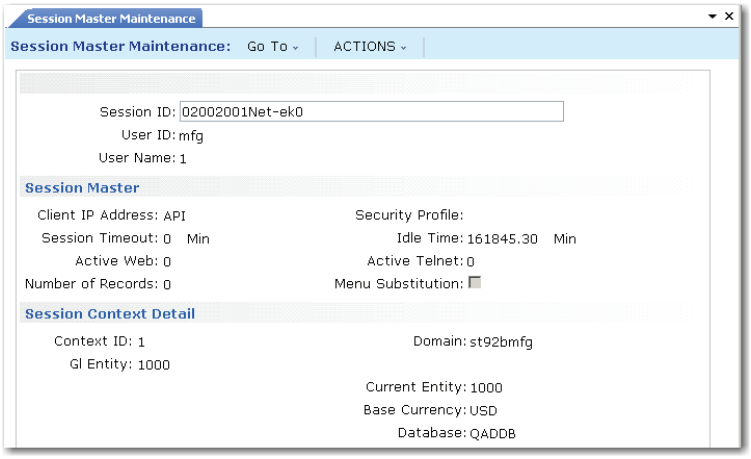
1. Set up the QAD report server after installing Cognos 8.2.
2. Create reports and dashboards using Cognos Report Studio. See *User Guide: Cognos BI 8 Report Studio* for details.
3. Configure report settings and perform report synchronization using programs on the Report Setup Menu:
   a. Use Report Control (36.4.21.24) to configure report server settings and view or modify URL parameters.
   b. Use Report Synchronization (36.4.21.2) to synchronize reports between the system and the report server.
   c. Use Report Parameter Synchronization (36.4.21.4) to synchronize report parameters in the system with the report server.

4 Create menu entries for the new reports using Menu System Maintenance (36.4.4.1).

Monitoring User Sessions

Use Session Master Maintenance (36.4.22) to view information about users who are currently logged in to the system through the .NET UI. This information displays in the form of session records, each identified by a unique session ID that is generated by the system. A session record is automatically created when a user successfully logs in to the system from the .NET UI and is deleted when the user logs out.

Fig. 3.33
Session Master Maintenance (36.4.22)



Only some of the settings displayed on this screen apply to .NET UI sessions. You can ignore the following: Active Web, Number of Records, Security Profile, Active Telnet, Menu Substitution, and Context ID.

The session context detail displays information about the current workspace (domain). One user session can be associated with multiple contexts if programs have been activated in more than one workspace.

You can also monitor the status of sessions for HTML maintenance programs, reports, and inquiries using Connection Manager, which is described in detail in the installation guide for your system.

Chapter 4

Printers

This section describes how to set up and use printers.

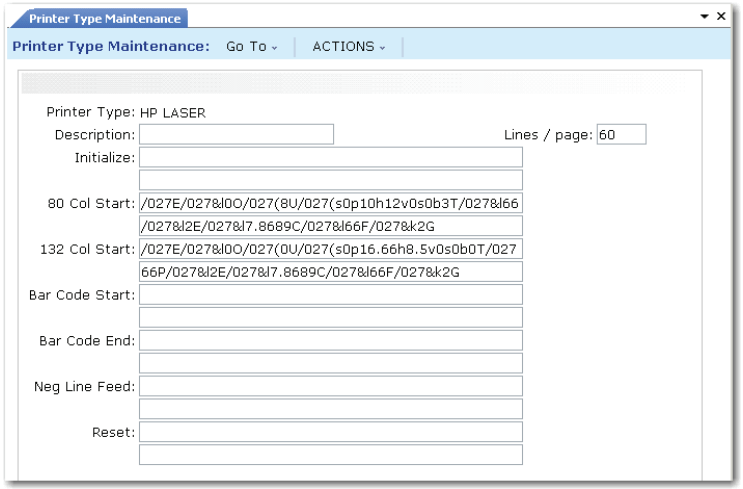
Introduction

You can send reports, inquiries, and browses to a variety of printers—both local and network. The Printer Management menu (36.13) contains programs for setting up system printers and default printers by user or group. Printers apply to all domains in a database.

Defining Printer Types

Before setting up printers, define printer types using Printer Type Maintenance (36.13.1).

Fig. 4.1
Printer Type Maintenance (36.13.1)



Printer Type. Select your printer type from the list of predefined types. If your printer type is not in the list, use a similar printer type or define a new one.

To define a new printer type, you specify a series of programming sequences to control printer characteristics and behavior in the following situations:

- 80-character-width print jobs
- 132-character-width print jobs
- Barcode print jobs
- Hardware initialize and reset

Using control characters, you define how your printer performs such tasks as modifying fonts, changing page orientations, producing multiple copies, and so forth. Your printer manual is the best resource for control code definitions.

Note Without correct control codes, the related aspect of printer control will not work.

Use normal ASCII characters in the control fields. For nonprinting characters, also called control characters, use a slash and the three-digit ASCII number for the character. Table 4.1 lists characters frequently used in control sequences.

Table 4.1
Control Characters

Control Character	ASCII
Backspace	/008
Tab	/009
Linefeed	/010
Form Feed	/012
Carriage Return	/013
Escape	/027

Default system data includes correct control sequences for some commonly used printers.

Note One of the default printers is terminal. Use terminal in a character interface and page in the QAD .NET UI.

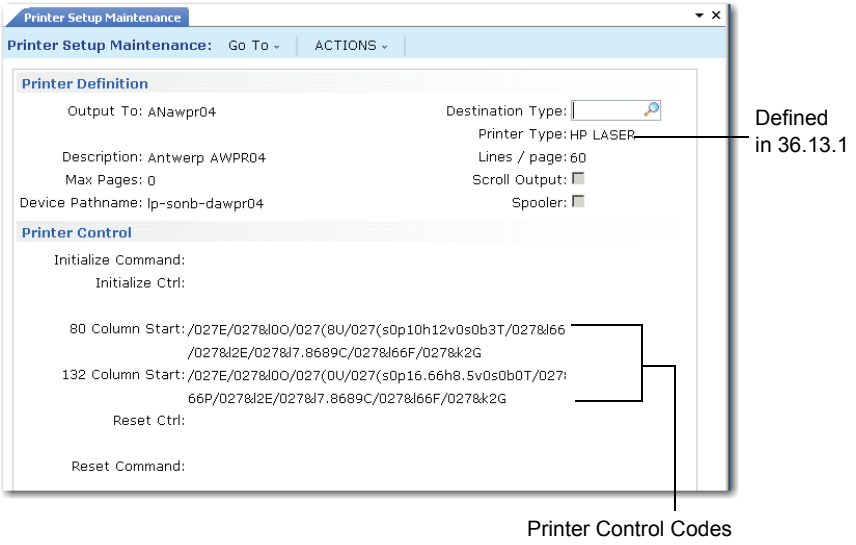
Table 4.2
Sample Printer Control Codes

Code	Function
/X27E	Printer reset
/X27&l3A	Folio paper format
/X27&lXO	Portrait orientation
/X27&l1O	Landscape orientation
/X27&l1S	Long edge binding (prints on both sides)
/X27&l66F	Bottom margin is 66 lines from top
/X27(sXp16.67h8.5vXsXbXT	Pitch 16.67, height 8.5, default style, thickness, font
/X27&l7X89C	Adjusts vertical index in steps of 1/48 inch
/X27(sXp16.67hXs3b4X99T	Pitch 16.67, height default, bold, courier (4X99)

Setting Up Printers

After you have defined printer types, use Printer Setup Maintenance (36.13.2) to set up printers and other output devices.

Fig. 4.2
Printer Setup Maintenance (36.13.2)



Output To. Assign a unique name to each printer or other output device. This name displays in the Output field of reports and inquiries. The QAD demo databases use *printer* and *terminal* for the most commonly used printers. However, you can use any name.

You can set up more than one record for the same printer, as long as you use different names in Output To. For example, this lets you access the same printer from both character and Windows clients.

Destination Type. Enter the type of device represented by this printer definition. Valid values are:

- Default. This is a server printer, a terminal display, a Windows display, or output to page. In Language Detail Maintenance (36.4.2), this mnemonic is assigned to value 0 (zero).
- EMail. This printer definition sends the report output to an e-mail message. For this to work properly, you must have an e-mail system that accepts a command-line interface. The e-mail system must be set up in E-mail Definition Maintenance, and the User

See “Building an E-Mail System Interface” on page 93.

Maintenance record for each user must include an e-mail definition and e-mail address. In Language Detail Maintenance, this mnemonic is assigned to value 1.

Printer Type. Optionally enter a printer type defined in Printer Type Maintenance. If you specify a type, the characteristics assigned to that type are copied into this printer setup record. You can modify them as required.

Description. Enter a description of the output device. Describing the physical location of a printer can be helpful.

Lines/Page. Enter the maximum number of lines to appear on a page. If you set up a printer to accept a maximum of 6 pages at 72 lines to a page, the printer prints only the first 432 lines of output, exclusive of the trailer.

Max Pages. Enter the number of pages a device can accept. If zero, no page limit applies.

Note If you try to print checks, forms, and similar items on a device with a maximum page limit, an error message is displayed.

Scroll Output. Select this option to have the system accept a maximum of 3,000. Otherwise, the Max Pages limit applies.

Device Pathname. Specify the operating system command or path name that enables you to output to this printer. A device path name is normally not required for a terminal. However, if you are setting up a slave printer or a terminal window under X-windows, you may need to enter a path name. Table 4.3 lists examples of device path names.

Table 4.3 Sample Device Path Names

Device Path Name	Operating System	Effect
`//arnt01/supjet1`	Windows	Prints to network printer, shared as `supjet1` off the `arnt01` print server.
`printer`	Windows	Prints to Windows captured default printer.
`lp -d supjet1`	UNIX	Passes UNIX `-lp` command to operating system, causing printing at destination `supjet1`. Spooler must be selected.

Spooler. Indicate if this is a spooled device. This field only applies to UNIX systems.

Initialize Ctrl/Reset Ctrl. A slave printer is one connected to a local PC printer port or the printer port of a dumb terminal. To transfer printer output to the proper port, you may need to specify control codes for these fields. The initialize control string passes output from the terminal to the print device. The last section of the Reset control string returns output to terminal. Set up control strings for each printer. In UNIX, the slave printer device path name is:

```
/device/tty
```

Defining a Printer for Use with QAD .NET UI

If users generate reports from the QAD .NET UI and want to view them immediately, they should choose the Page output device rather than terminal. Output to terminal is not formatted to display correctly in a browser.

The Page output device should be defined with the following settings:

- Max pages is 0.
- Destination type and printer type are blank.
- Lines per page is 66.
- Scroll output is selected.
- Spooler is not selected.

Setting Default Printers

Use Printer Default Maintenance (36.13.4) to assign default output devices to users.This is only the default; you can change it to any valid device when you run the program.You can apply a record to all users by entering an asterisk (*) in the User ID field.

Note Default output devices apply only to reports; the default device for inquiries is always terminal.

You can specify devices for a user ID or a combination of user ID and menu selection. This can be useful for specialized tasks such as sending checks to a check printer; the same user can have different default output devices for different programs.

The default does not necessarily have to be a physical printer; you can also choose to send output to the terminal, page, a window (GUI only), or an e-mail recipient.

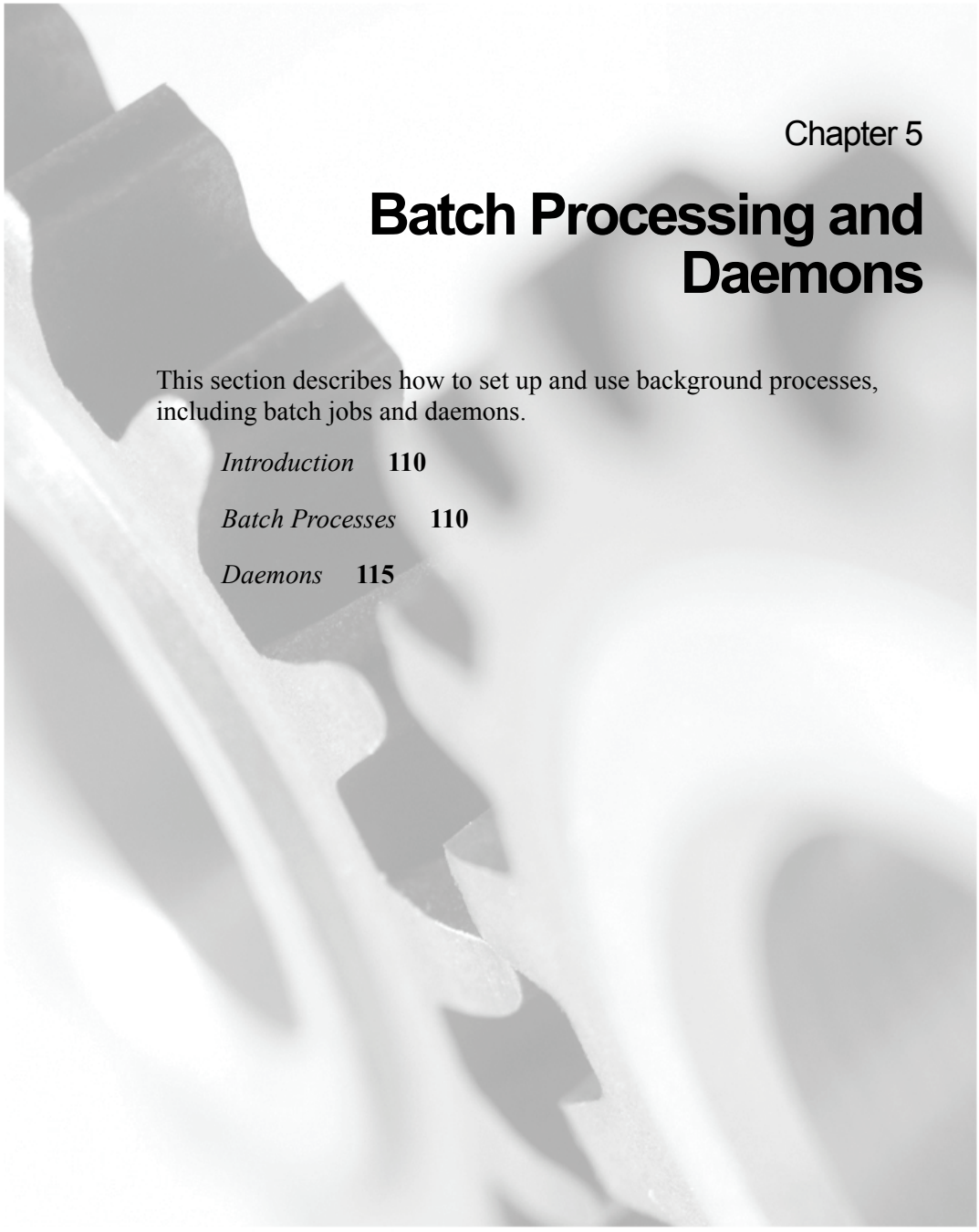
Defining Document Formats

Some programs let you specify alternative formats for printed documents in addition to the system-defined default formats. For example, an Italian customer may require a different sales order layout than a US customer. In that case, you can specify a predefined alternate format in the Form Code field of Sales Order Print (7.1.3).

You do not use a menu-level program to define alternate document formats. Instead, you must create a Progress program to generate them. Use the following steps to do this.

1. Create a Progress program to format the document as required.
2. Name the new program file appropriately so it can be located by the print program. The file name is typically created by removing the first two characters of the print program name and appending a two-character form code.
3. Modify the applicable print function to consider the new form code as valid.

Example You create two new sales order formats, identified with form codes AA and 2. The program name for Sales Order Print is `sosorp05.p` and the default sales order layout is defined by `sorp0501.p`. Use program file `sorp05AA.p` to store sales order form code AA and program file `sorp0502.p` to store form code 2. Be sure to include the zero preceding the 2. Then, modify `sosorp05.p` to define the two new formats as valid.



Chapter 5

Batch Processing and Daemons

This section describes how to set up and use background processes, including batch jobs and daemons.

Introduction

The Batch Processing/Daemon (36.14) menu includes two types of programs:

- Programs for creating and monitoring batch requests. These functions let system administrators edit and process batch jobs from multiple domains without having to switch the current working domain associated with their user IDs.
- Programs for configuring and managing daemon processes that run in the background whenever your database is active.

Batch Processes

A batch process is a group of processes run simultaneously. You can use batch functions to defer processing and report printing for reasons such as the following:

- A printer is busy or broken.
- Users want to be able to continue working without having to wait for lengthy reports to finish.
- Reports need to be run in a sequence, regardless of how they are submitted.
- You want to balance system load by running CPU-intensive programs when system load is low, perhaps at night.

Note Batch reporting for Financial reports is handled by the Report Daemon and .Net Report Service. See “Report Daemon” on page 137.

Define Batch IDs

To set up a batch process, system administrators first create batch IDs in Batch ID Maintenance (36.14.1). Use ID names that are descriptive and easy to remember, such as Paycheck, Monthly, or After5. You also assign the batch a priority that determines when it will run. Requests with the highest priority are run first.

Users then submit reports or programs that can be run in batch mode and specify the batch ID.

Note Batch IDs are domain specific. You must set up a separate set of IDs for each domain. You can, however, manage batch processes from multiple domains in the batch processing programs.

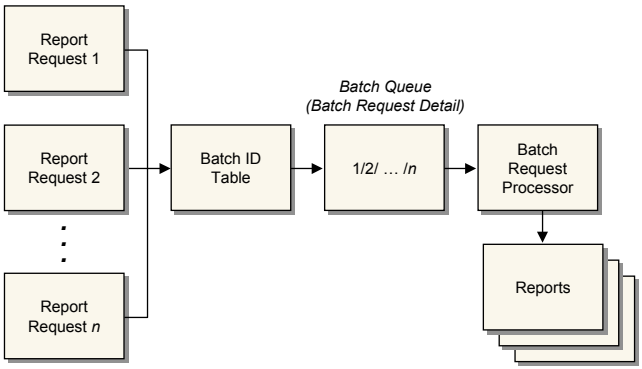


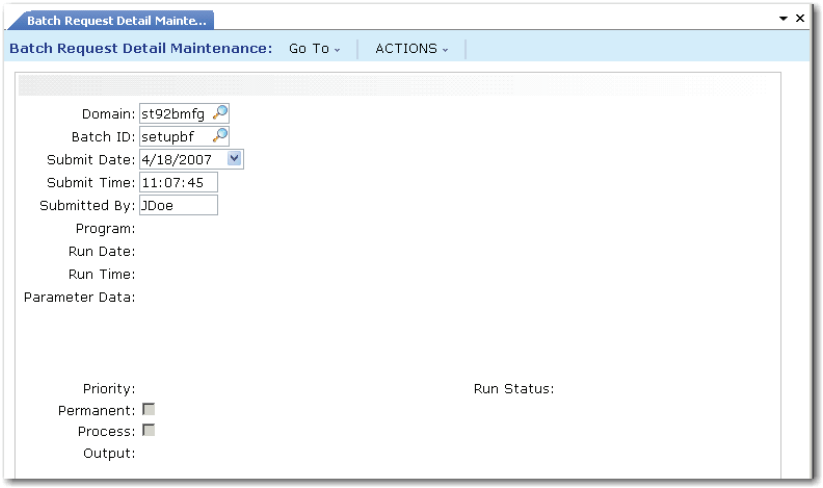
Fig. 5.1
Batch Processes

Review Batch Jobs

Usually the system administrator reviews batch requests prior to batch processing. Use Batch Request Detail Maintenance (36.14.3) to view reports and programs submitted to any batch. You can eliminate duplicate or unnecessary requests, prioritize requests, and redirect output as needed.

You must specify the domain associated with the batch requests you want to modify. You must have access to any domain you specify in User Domain/Entity Access Maintain.

Fig. 5.2
Batch Request Detail Maintenance (36.14.3)



As each request is executed, its status is updated to reflect whether it completed normally. Statuses include:

Failed (incomplete)

Complete

Running

When a batch does not complete normally, use Batch Request Detail Maintenance (36.14.3) to select the Process field and restart Batch Request Processor.

Process Batch Request

Use Batch Request Processor (36.14.13) to run reports and/or programs submitted by users with a batch ID. You can process up to 10 batch IDs in a single run. Each batch ID can be associated with a different domain. This lets you manage batch requests for multiple domains within one database without having to change your current working domain.

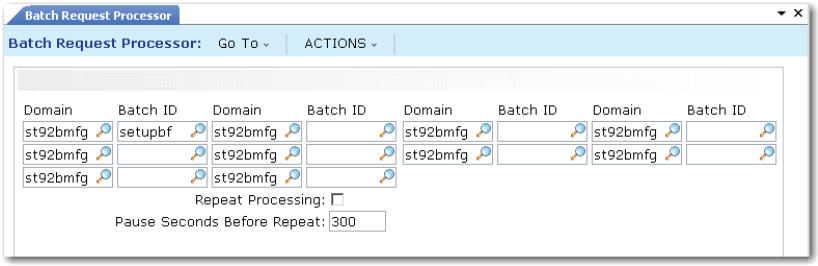


Fig. 5.3
Batch Request Processor (36.14.13)

When you run a batch process, the system executes all items queued for a given batch ID in the requested order. You control the batch order by assigning a priority to each batch ID.

Invoke Batch Processing from CIM

In UNIX or Windows, you can create a batch file that invokes batch processing. You can then schedule when to run the script of the batch file. The scheduling capability of the operating system lets you run the batch processing at a time that is most convenient for you.

Note This is especially important if you have a Unicode database with domains that have incompatible code pages. This lets you run batch jobs that bypass the restrictions on a character client session. See “Multiple Languages and Unicode” on page 46 for details about restrictions in a Unicode database.

To set up a batch script, follow these steps.

1 Prepare a file that anticipates all data entry to Batch Request Processor (36.14.13).

  The file should use CIM format. The first line provides login information:

```
"<User_Name>" "<Password>" "<Login_Domain>"
"mgbatch.p"
"<Domain>" "<Batch_ID>"
- - - - - - - - - - - - - - - - - - -
"<Is_Repeat>"
-
.
.
"Y"
```

In the script, `mgbatch.p` is the program name for Batch Request Processor. *`Domain`* and *`Batch_ID`* identify the batch requests to process. The line *`Is_Repeat`* indicates that requests for multiple domains can be included in the script. A hyphen (`-`) indicates to tab through a field; the two dots are exits, and `Y` confirms the exit from your session.

See Chapter 7, “CIM Interface,” on page 179 for more details on CIM load processing.

2 Create a `.p` file of following format. Replace *`Input_File`* with the path of the file that you prepared in the previous step.

Note If you are working in UNIX instead of Windows, the first statement in the following script is unnecessary.

```
Assign PROPATH = <Propath>.
Input From <Input_File>.
Output To <Output_File>.
Run mf.p.
Input Close.
Output Close.
```

3 Set up a batch file. The batch file is a `.sh` file that can be scheduled using the UNIX `crontab` command or a `.bat` file that, in Windows, you can add to Scheduled Tasks in Control Panel.

To set up the batch file, use the Progress command `mpro` (UNIX) or `prowin32.exe` (Windows) to invoke the `.p` program that you created in step 2.

- In UNIX, the `.sh` file has the following structure:

```
TERM = <Term>;
DLC = <DLC>;
PATH = <Path>;

PROPATH = <Propath>;

mpro <DB_Parameters> -p <Progress_Program>
<Startup_Parameters>
```

- In Windows, the `.bat` file has the following structure:

```
SET DLC = <DLC>

SET PATH = <Path>

prowin32.exe <DB_Parameters> -p <Progress_Program>
<Startup_Parameters>
```

The table describes the variables used in the scripts.

Table 5.1 Variables in Batch File

Parameter	Description
DLC	Specify the value of the *DLC* system variable.
PATH	Specify the value of the *PATH* system variable.
TERM	For UNIX only, specify a terminal type.
PROPATH	Specify the value of the Progress *PROPATH* variable.
DB_Parameters	Specify the parameters to connect to the database. For more information, see Progress help.
Progress Program	Specify the path of the `.p` program that you created in the previous step.
Startup Parameters	Specify other parameters for `mpro` or `prowin32.exe` to start. For more information, see Progress help. If you are using OpenEdge 10.1A or above, you can specify the database connection to support Unicode with the following setting: • `-cpinternal utf-8` • `-cpcoll icu-uca` You need Unicode support if the batch processing you execute will access more than one domain in a database, and the domains store language data that have incompatible code pages.

Daemons

Daemons are server-based processes that let you run background tasks. The user has no direct input with the daemon processes.

Daemon processes apply only to component-based functions. They can be run on the same application server as QAD Financials, or you can specify a different appserver for each daemon. You should define the appservers to be used for daemon processing during system implementation.

Important Some daemon processes must be running to ensure the integrity of your application. You should ensure that these processes are configured to start when the database is started.

Overview of Daemon Processing

In general, tasks for the daemon are stored in a queue in a dedicated database table. The daemon regularly checks its queue for tasks and then processes them. If no more tasks remain in the queue, the daemon enters a period of inactivity known as sleep mode. After a predefined period, the daemon exits sleep mode and check its queue for new requests.

The behavior of each daemon and its request queue are controlled and monitored using specific maintenance programs. You can start multiple instances of a daemon if the workload requires this.

The system has the following daemons:

- Balance daemon
- Budget daemon
- Cross-Company daemon
- Event Daemon
- History daemon
- Replication daemon
- Report daemon
- Scan daemon
- Time Out daemon
- XML daemon

All daemons, except the XML daemon and Report daemon, share the same control activities. The XML daemon differs from the others in that it creates its own requests by importing files from a specific directory on the file system. Otherwise, the XML daemon behaves like the other daemons.

The Event daemon publishes Financials events as XML messages that can be exported to other application instances or external Financial applications using QXtend.

Requests to the Report daemon are handled by the .NET Report Service, which is a separate server installation. You also use the Report Service to stop and start the Report daemon, and can monitor Service activity using the daemon Monitor function.

The Replication daemon is only used during the creation of new domains to propagate shared set data.

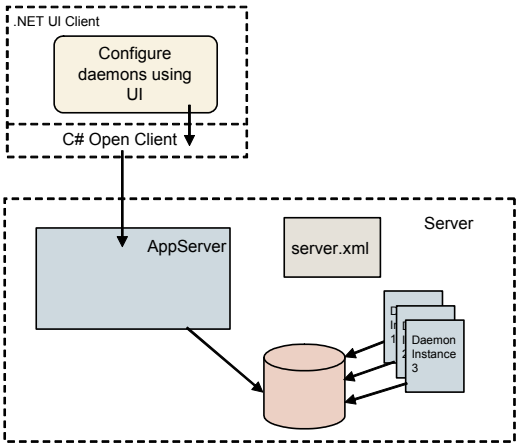


Fig. 5.4
Daemon Architecture

Daemon User

Each daemon must log into the system to perform its updates, and you specify the user login ID and password to associate with each daemon when you configure it. The recommended practice is to use the same user ID for all of the daemons, to make maintenance easier.

The user you specify must have access to all activities in Role Permissions Maintain and all domains and entities.

Important Whenever you create a new domain or entity, you should assign access to this system user. When you create a new domain and confirm it, the replication will fail if the Replication daemon user does not have access to the new domain.

Types of Daemon

Balance Daemon

The Balance daemon operates in a similar manner to the History daemon. It builds the supplier and customer balance and movement data and history files.

The Balance daemon updates the supplier and customer history tables each time a movement is created on an invoice.

Note If queue records are waiting to be processed by the Balance daemon, the supplier and customer balances may be inaccurate.

Budget Daemon

The Budget daemon allocates postings and commitments to the appropriate budget topics. All postings are processed, including those in the transient and management layers.

Budgets are composed of budget topics, and each topic is linked to a level in the chart of accounts (COA) and to specific values for that COA level. The Budget daemon inspects all budget definition tables, and updates the actuals for the relevant budget topics.

The Budget daemon should be active when allocations are executed because the allocation functionality uses the same tables. The Budget daemon ensures that the most current values are available for an allocation run. Allocations are discussed in *User Guide: QAD Financials A*.

Cross-Company Daemon

The Cross-Company daemon processes automatic cross-company postings that cannot be performed manually in the UI. The daemon is used to perform calculations and postings to spread the transaction load on the CPU.

The Cross-Company daemon handles transactions related to invoices and banking entries. These can occur in the following activities:

- Customer invoices and credit notes
- Supplier invoices and credit notes
- Banking entry
- Payment selection
- Payment documents
- Open item adjustment

The Cross-Company daemon processes transactions in the official layer only, and completes the linking fields in both the source and target posting line.

Posting occurs in the source entity using the Cross-Company Control account for the target entity. The system creates a request for the Cross-Company daemon, and posting and request creation occur within a single transaction. The daemon processes the request and creates the posting and the movement in the sub-ledger of the target entity.

Event Daemon

The Event Daemon publishes Financials business events, such as record creation events, as XML messages. The Event daemon publishes the messages in its queue to a preconfigured destination, which can be a directory on a server or a QXtend server. Event publishing lets you export data updates to other Financials instances, without the need to extract the data from the database. See “Event Daemon” on page 133.

History Daemon

The History daemon populates the database with condensed GL transaction data, and updates GL and SAF balances for each period. Detailed transaction data is accumulated in tables to increase performance in, for example, drill downs and reporting. All GL postings, including those in the transient and management layers, are processed.

The History daemon generates historical data grouped by a number of criteria, such as the posting period, the account, sub-account, project, cost center, period mark, daybook, and the entity used in the transactions.

The data fields in the history tables include the period movements, balances, and year-to-date values and are provided in the base, transaction, and management currencies. In addition, the quantity field is included.

Replication Daemon

The Replication daemon makes domain shared set data available to the operational functions, and replicates the data to the appropriate operational domain.

During implementation of the system and setup of the domains, you can continue to modify the data associated with a domain for as long as you require and then confirm the setup when you are satisfied that it meets your business requirements. Until the setup of a domain is confirmed, it is not available to your operational functions.

The availability of domain data to the operational functions is controlled by the Setup Complete field in the Domain function. When you select the Setup Complete field, the Replication daemon creates request queue records for each shared set. After the data has been reproduced, you cannot change the shared sets and base currency of the domain. In addition, you cannot link any of the entities linked to the domain to a different domain; the relationship is now permanent.

See *User Guide: QAD Financials A* for details on the Domain function.

The data made available to the operational functions includes daybooks; GL accounts, sub-accounts, and cost centers; suppliers, customers, and exchange rates.

The system creates a single Replication daemon request for a maximum of 90 records. Each request contains:

- The shared set code
- The ID of the primary entity of the domain
- A list with the IDs of the records that need to be replicated
- A priority that indicates the order in which each request should be processed

If any of records in a replication request fails to be replicated, all other records in the request fail also. Failed replication records remain in the daemon queue and have the status Processed-Error.

The user ID specified for the Replication Daemon login must have access to all domains and entities in the system.

Important You must grant the daemon user ID access to a new domain before confirmation or the replication process will fail.

Report Daemon

Financial reports can be printed to screen or to a printer directly, or can be batch printed from a report queue. You can schedule batch reports, output them to files, or e-mail them to addresses or roles. The Report daemon processes these batch reporting requests. See “Report Daemon” on page 137.

Scan Daemon

The Scan Daemon lets you configure the system to monitor a directory for documents to be attached to new records in the QAD application database. You configure the daemon by instructing it what to do when it finds a document.

Note In order to use the Scan daemon, you must enable workflow, since the scanned documents are associated with draft objects and sent to user inboxes for completion. See “Configuring Workflow” on page 174 for details.

For example, you can set up the daemon to monitor a directory for incoming supplier invoices. The Scan daemon processes documents located in entity-specific scan directories. The daemon creates a draft instance of the supplier invoice, attaches the scanned document, and sends the draft instance to the inbox of any user with access to the relevant domain and entity and a role with access to the Supplier Invoice Create activity. The original scanned documents are renamed in the scan directory to prevent a document from being processed multiple times.

Supplier invoice is the most common type of document to scan, but you can also use this feature to scan documents related to any components that support the save-as-draft feature. For example, you could scan new customer profiles and send them to the inbox of those responsible for creating customer records.

You can attach documents of any file type, but you should ensure that your users have the correct applications to open the file. Typical files to attach are `.pdf`, `.doc`, `.txt`, and `.jpg`.

The Configure activity for the Scan daemon includes an additional grid where you specify the scan directory associated with each entity. See “Configuring the Scan Daemon” on page 131 for details.

Time Out Daemon

You can configure a user time-out setting defined in Security Control (36.3.24). This feature lets system administrators automatically terminate inactive user sessions, therefore reducing system load and improving performance for active users. Time out is defined as a number of minutes a logged in user can be inactive.

See *User Guide: QAD Security and Controls* for details on security settings.

Note If you decide to implement this setting, it can result in loss of data if users have partially completed input in a program.

If you want to use the time-out feature, in addition to the setting in Security Control, you must ensure that the Time Out daemon is also running. This daemon ensures that the system is aware of component-based activities that have being started and resets the time-out period whenever a user initiates an activity.

If the daemon is not running, the system is only aware of the execution of standard programs and may terminate users running component activities.

XML Daemon

The XML daemon processes external data in XML format, by parsing XML document headers and calling the relevant software component to process the document. See “XML Daemon” on page 141.

Daemon Functions

Use the following activities to maintain and control daemons:

- Configure
- Clear Queue
- Monitor
- Start
- Stop
- Unconditional Stop

These activities are the same for all of the daemons, except the XML daemon, which is described on page 133, and the Report daemon, which is described on page 133. In addition, the Scan daemon has an extra grid, described on page 131.

Both the Configure and Monitor activities check the status of the daemon and correct it before displaying. For example, if the daemon status is set to Running but no daemon process is detected on the QAD Financials appServer, the status is reset to Inactive before the Configure or Monitor screens display.

Note You configure, clear queues, and monitor the Report daemon in the same way as for other daemons. However, the Report daemon is stopped and started from within the .Net Report Service.

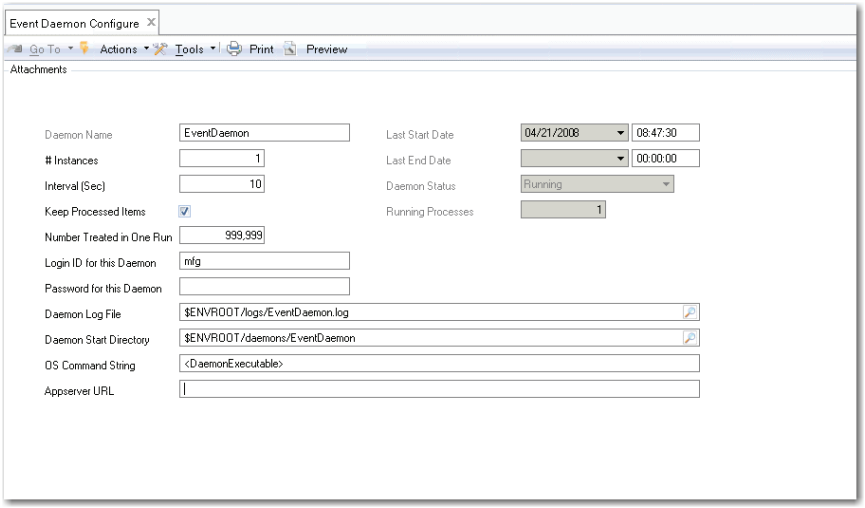
Configure

The Configure activity lets you configure maintenance information for the daemon, such as the login ID and password, in addition to the log file and start directory. For security reasons, you should use a dedicated daemon user name. The user name must have access to all domains and entities have role permissions to all component activities in the database.

Note The activities controlled by the daemon must be included in the list of activities for the role assigned to the daemon user name.

The Configure activity also lets you define the time interval before the daemon checks its queue for new requests, indicate whether to record processed items for audit purposes, and set the number of records that the daemon should process in each run.

Fig. 5.5
Event Daemon, Configure (36.14.16.1.1)



Field Descriptions

Daemon Name. Displays the daemon name.

 # Instances. Enter the maximum number of instances that can be started for this daemon. It is possible to enter 0, meaning that the daemon is not configured for use in the system setup.

Interval (Sec). Specify the time interval in seconds before the daemon checks its queue for new requests.

Keep Processed Items. Select the field to record processed items for audit purposes. When you clear the field, the system deletes all successfully processed requests from the Monitor screen. See “Monitor” on page 127.

Note It is recommended to clear this field for any daemons that process a high number of requests so that you can easily see requests processed with errors.

Number Treated in One Run. Specify the number of requests that the daemon should process in a single run. Note that during a run, the daemon does not react to Stop commands. If you set this number too high, the response time to a Stop command increases accordingly.

Login ID for this Daemon. Specify the ID of a valid active user that is assigned a role that grants permission to all activities and access to all domains and entities. To ensure proper security is enforced, the daemon process accesses the database by logging in with this name.

Note The login details you specify are used by the Report daemon to determine the From address when you e-mail a report.

Password for this Daemon. Specify the password for the user ID in the previous field.

Important If the password of this user is changed in User Maintenance, the change is not automatically reflected here. You must ensure that the passwords are updated in both places, or the Daemon process will fail on login.

Daemon Log File. Specify the full path of the log file in which the system records the daemon's actions.

By default, the system uses the Environment Root variable *$ENVROOT* to specify the location of the daemon log file. The *$ENVROOT* variable takes its value from the `serverXML` file for the application environment, and ensures that the financial database can be copied between environments without the need to reconfigure the daemons.

The log file path by default is:

`$ENVROOT/logs/<daemon name>.log`

You do not normally need to modify this default.

To troubleshoot daemon activity, you can modify how messages are generated to the daemon log file by setting the debug level in the daemon configuration file. Each daemon has a configuration file of the form `<DaemonName>.config` in the PROPATH of the Financials appserver.

Example To increase the message detail for the History daemon, modify the debug level in `HistoryDaemon.config`.

Daemon Start Directory. Specify the directory on the file system where the physical daemon process is started. Progress keeps its internal and temporary files related to the daemon process in this location.

The system also uses the Environment Root variable *$ENVROOT* to specify the location of the daemon start directory.

The start directory path by default is:

`$ENVROOT/daemons/<daemon name>/`

You do not normally need to modify this default.

OS Command String. Enter the physical command used to run an instance of the daemon. A part of this command string is determined by the application itself, and is referenced by using `<DaemonExecutable>`, and differs between operating systems. You can add parameters to the command, but should include the `<DaemonExecutable>` tag to refer to the actual executable.

Appserver URL. Enter the connection URL for the application server on which this daemon is to be run. You can optionally use a different server than the main application server.

Important It is recommended that you define a different appserver connection for the Report daemon. This ensures that the batch processing of report requests does not have an impact on overall server performance.

Start Date of Last Daemon Run. This field displays the last time the daemon started.

End Date of Last Daemon Run. This field displays the last time the daemon stopped.

Daemon Status. This field displays the daemon status. Valid values are Inactive, Running, and Stopping. The Start, Stop, and Unconditional Stop activities control the status changes.

Number of Running Processes for this Date. Displays the number of instances of the daemon that are currently running. This value is zero when the daemon is inactive.

Start

The Start activity starts the daemon. The system displays a message that indicates that the daemon has started successfully.

Stop

The Stop activity stops the daemon.

The daemon stops after it completes its current task. Depending on the type of daemon and the amount of time it takes to complete an operation, some time may pass before the daemon detects the stop request and shuts down.

The system displays a message that indicates that the daemon has stopped successfully.

Unconditional Stop

If you cannot stop the daemon using the Stop activity, the Unconditional Stop activity stops the daemon. This task should only be performed by the system administrator.

Note The Start, Stop and Unconditional Stop activities are not available for the Report daemon, which is managed by the .Net Report Service.

Monitor

The Monitor screen lets you view the start and stop dates for the daemons and the requests that are processed, waiting, in progress, or incorrectly processed.

The screen is composed of a header area and two grids.

The header area contains fields that display the daemon name, the start and end times, and the daemon status. The header also contains four fields that let you choose the type of requests to include in the view.

The center grid of the Monitor displays:

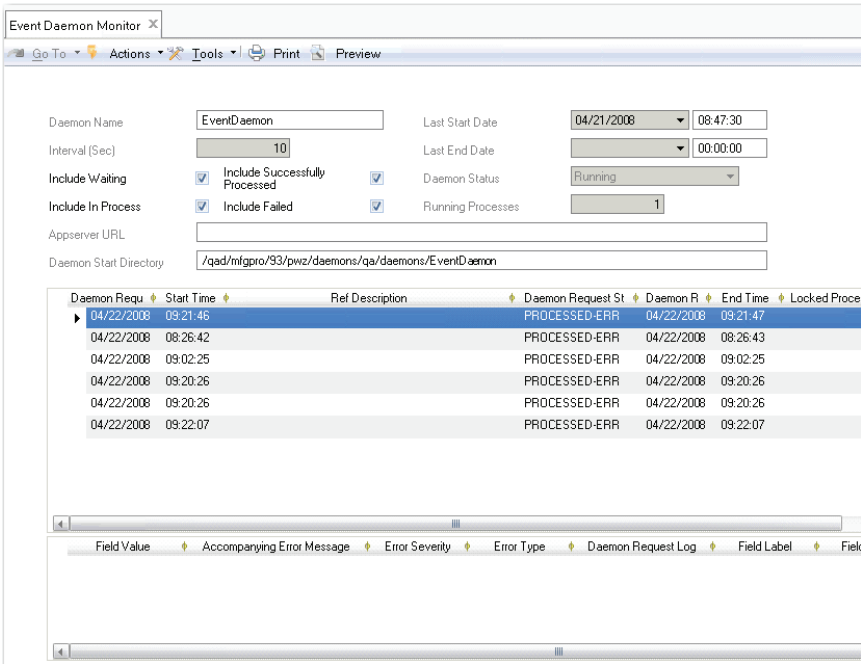
- The start and end time of the request currently processing.
- A description of the request.
- The status of the request. The valid values are PROCESSED-OK, PROCESSED-ERR, WAITING, IN-PROCESS.

Additionally, if you set a specific start date and time for the request, the center grid also displays this data. It is only possible to set a specific request date and time for the XML daemon.

The lower grid displays the error information for each request line, if any.

The view does not store correctly processed records after the next refresh unless the Keep Processed Items field is selected in the Maintain screen.

Fig. 5.6
Daemon Monitor
(36.14.16.1.3)



Field Descriptions

Daemon Name. Displays the daemon name.

Interval (Sec). Displays the time interval in seconds before the daemon checks its queue for new requests. You can update this value using the Configure screen.

Include Waiting. Select the field to include waiting requests in the view.

Include Successfully Processed. Select the field to include correctly processed requests in the view.

Include in Processing. Select the field to include requests that are in progress in the view.

Include Failed. Select this field to include incorrectly processed requests in the view.

Number of Waiting Queues. Displays the number of records in the daemon queue that have the status Waiting.

Last Start Date. This field displays the last time the daemon started.

Last End Date. This field displays the last time the daemon stopped.

Daemon Status. This field displays the daemon status. The valid values are Inactive, Running, and Stopping. The Start, Stop, and Unconditional Stop activities control the status changes.

Number of Running Processes for this Daemon. This field displays the number of instances of the daemon that are currently running. This value is zero when the daemon is inactive.

Appserver URL. This field displays the URL of the appserver on which this daemon is running. The field is blank when the daemon is running on the main appserver.

Daemon Start Directory. This field displays the location of the daemon start directory, which the system retrieves from the daemon configuration.

Status Grid

Daemon Request Start Date. This field displays the date on which the daemon request was submitted.

Daemon Request Start Time. This field displays the time at which the daemon request was submitted.

Ref Description. This field displays an abbreviated description of the request type.

Daemon Request Status. This field displays the status of the request. The possible statuses are: Waiting, In Process, Processed OK, and Processed Failed.

End Date. This field displays the date on which the request was processed.

End Time. This field displays the time at which the daemon finished processing the request.

Locked Process. This field displays the ID of the process that is running the request. Some daemons can be run several times at once. Every time a daemon is started, a new physical Progress process is started.

Daemon Request Log. This field displays the full history of the daemon queue record.

Sequence. This field displays the order in which the requests should be handled.

Requested Start Date. This field displays the requested start date for the daemon request.

If no start date is requested, the request will be processed as soon as possible.

Requested Start Time. This field displays the requested start time for the daemon request.

If no requested start time or start date is specified, the daemon processes the request immediately. If no start time is specified, but the start date is, the request will be processed as early as possible on the requested date.

Error Grid

Field Value. Displays the value in the field that initiated the error.

Accompanying Error Message. Displays the message associated with the error.

Severity of the Error. Displays the severity level of the error.

Error Type. Displays E for error, I for informational, and W for warning.

Freeze/Reactivate. Stops the screen updates temporarily and reactivates them.

Clear Queue. Clears the daemon request queue, based on the request types you have selected.

Important Only qualified and trained personnel should clear queues because this action can potentially destroy valuable information or interfere with critical application processes.

Report Daemon Options

The Monitor screen for the Report daemon has two additional fields:

Socket Port. This field displays the server port on which the .Net Report Service is listening for requests.

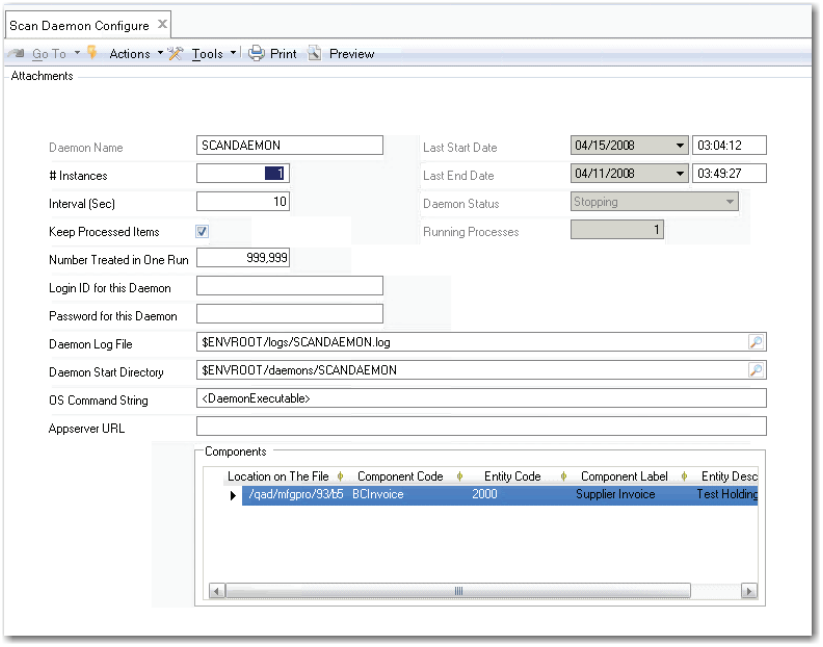
Socket Server. This field displays the server on which the .Net Report Service is running.

Note These fields display values when the .Net report Service is running successfully. If the .Net Report Service is interrupted, these fields are blank. Errors in the .Net Report Service are displayed as Windows log events on the server on which the service is installed.

Configuring the Scan Daemon

The Scan Daemon Configure activity contains an additional grid for specifying the business component to be associated with a scanned document, the directory the daemon should monitor for documents, and the entity the scanned documents should be associated with.

Fig. 5.7
Scan Daemon, Configure (36.14.16.4.1)



Field Descriptions

Component Code. Choose the name of the business component that the scanned document should be associated with. You can select only components that support the Save as Draft feature. Currently, this includes the following components:

- Banking Entry
- Business Relation
- Customer
- Customer Finance Charge
- Customer Invoice
- Journal Entry
- Petty Cash
- Supplier Invoice
- Supplier

Example You specify BCInvoice, the technical name for a supplier invoice. The Scan daemon runs the Supplier Invoice Create activity when it detects a document to scan.

Documents for each scanned component must be located in a separate directory. The component description displays in the Component Label column.

Location on The File System. Enter the full path to the directory on the Financials appserver where documents to be scanned and associated with the specified component are located.

Important Make sure the directory is on the correct server or it will not be found.

Entity Code. Specify the entity associated with the scanned documents.

Event Daemon

The Event daemon processes Financials business events and publishes them to a defined destination. The system publishes events to a directory on the application server or to the QXtend Server for replication to other Financials instances or to third-party financial applications.

A business event in this case can be any modification to a Financials business object, such as an update to customer or supplier base data. By publishing the event, the system ensures that the update is replicated to other instances and so eliminates the need to extract the data directly from the database. The event consists of complete object data with context information that is published in XML format.

The Financials application uses a buffer queue within the running transaction to prepare messages before they are sent out. This ensures that no events are lost during system down-time and that no disruption of service occurs. Events are then published in XML format using a daemon process which routes the message to the specified destination.

Event publishing has three components:

- Event Destination

  Create an Event Destination to define the name of the event and the type and details of the server connections.

- Event Configuration

  Use Event Configuration to activate the publishing mechanism, configure which components and status transitions will trigger events—for example, supplier invoice creation—and indicate the destination queue of the event.

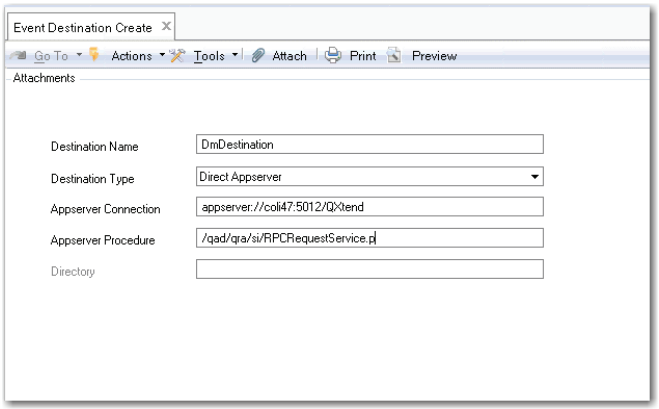
- Event Daemon

  The Event daemon processes the events in its queue and displays the event message status.

Event Destination

Use Event Destination Create (36.14.16.14.1) to define the name of the event and the type of messages it will contain. When you are using QXtend to export the events, you specify the connection to the appserver instance of QXtend Out (QXO) as the destination.

Fig. 5.8
Event Destination Create



Field Descriptions

Destination Name. Specify a destination name (maximum of 40 characters).

Destination Type. Select a destination type:

Direct Appserver. The events are published to an application server. When you select this type, the Directory field becomes unavailable.

To Directory. The events are published to a server directory.

Appserver Connection. Specify an application server connection. The protocol for the server path is:

```
appserver://<hostname>:<nameserver port>/<application service
name>
```

The default nameserver port for a daemon running on the main application server is 5162.

Appserver Procedure. Specify a procedure to be applied to the event.

This is the procedure that is called on the appserver. For QXO (QXtend Out) this typically is `qad/qra/si/RPCRequestService.p`

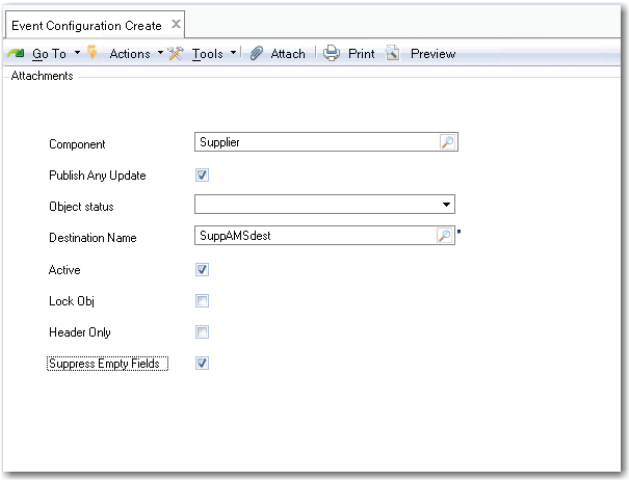
Directory. Specify a server directory. This field is available when you select a To Directory destination type.

Event Configuration

Use Event Configuration Create (36.14.16.15.1) to perform the following activities:

- Activate the publishing mechanism.
- Configure which components and status transitions will trigger events; for example, an update to Supplier base data.
- Select the destination of the event.

Fig. 5.9
Event Configuration Create



Field Descriptions

Component. Choose the business component that will trigger an event.

Publish Any Update. Select this field if any type of update to the business component is to trigger an event. Selecting this field makes the Object Status field unavailable.

Object Status. Choose the object status of the business component upon which the system will trigger an event.

Destination Name. Choose the name of a queue to which the event will be sent.

Lock Object. Select the field to lock the object for update after it has been published.

Active. Indicate if this is an active record.

Header Only. Select this field to publish event header details only.

Suppress Empty Fields. Select this field to exclude empty database fields from event publishing.

Event Daemon

The Event daemon reads event messages in its queue, and processes them according to the event configuration. You configure and monitor the Event daemon in the same way as other daemons.

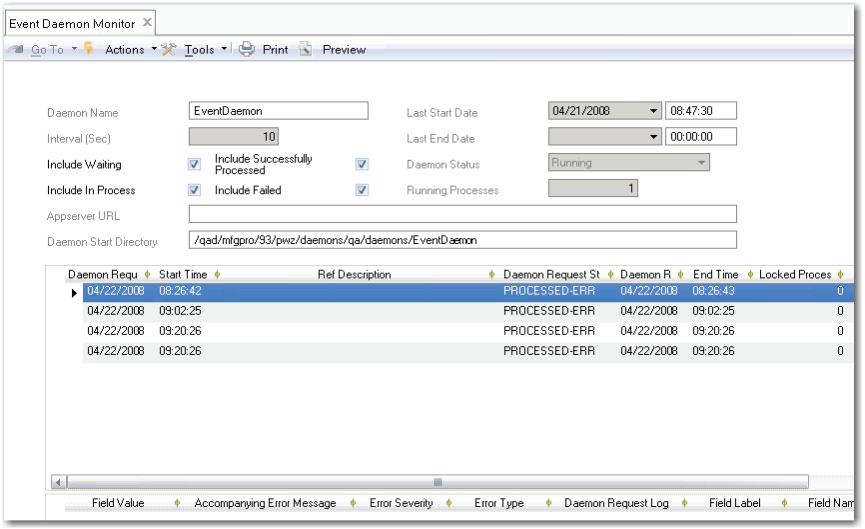


Fig. 5.10
Event Daemon Monitor

Report Daemon

You can output a report directly to screen, in which case the system retrieves the report template and processes the report on the application client.

You can also:

- Schedule the report to be processed (printed, mailed or saved) at a later date.
- Schedule the report to be pre-processed at predefined intervals (daily, weekly, monthly, or yearly).
- Save the report in file format.
- E-mail the report to systems users or roles.
- Combine some or all of these options.

Note You select these options in the server output processing section of the report screen. See *User Guide: Financials A* for details on generating reports.

When you select one or all of these options, the report request is sent to the report request queue and is then handled as a server-side activity by the Report daemon.

The Report daemon handles all batch report requests for the application, including scheduled reports, e-mailed reports, and reports to be saved to file format or sent to a printer. The Report daemon is managed by the .Net Report Service, which is installed as an additional service on the application server.

The Report daemon:

- Monitors the report request queue
- Calls the .Net Report Service and passes the report details
- Receives the result of the report and issues a confirmation that the report has been successfully generated

.Net Report Service

The .Net Report Service is responsible for starting, monitoring, and when necessary, restarting the Report daemon.

The .Net Report Service communicates with the Report daemon through the server socket port defined during the service configuration. Once the daemon is started, the report service:

- Runs the report business logic and resolves the e-mail addresses from roles and user names
- Retrieves a list of printers installed on the server and passes this list to the appserver for report printing
- Checks that the report templates stored on the server are up-to-date
- Updates the server socket port and socket host information on the Report Daemon monitor screen

When a report request is received from the Report daemon, the Report Service then:

- Passes the request details and report template to Crystal Reports

- Executes the report options by generating the report at the scheduled time, saving the report as a file, printing the report on a local printer, or e-mailing the report to a user or role
- Sends a confirmation request or error message to the daemon
- Receives a confirmation from the daemon and updates the daemon report log accordingly

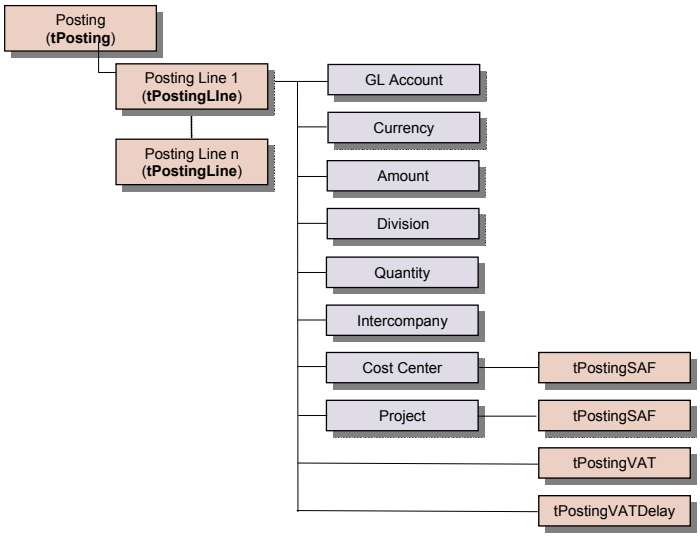


Fig. 5.11
Report Daemon Architecture

Configuring and Monitoring the .Net Report Service

The .Net Report Service is configured by defining service settings in the `QAD.CBFReportingService.exe.config` file, which is located in the service installation directory.

Define the following settings in the `<applicationSettings>` section of the file:

- The SMTP Server for e-mailing reports
- The folder in which to save reports in file format
- The check interval in seconds for the report daemon
- The TCP socket port on which the service listens for report requests
- The user login ID and password for the service—normally your application login and password

Example

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <configSections>
      <section name="qad.appserver" type=
"QAD.AppServer.ASConfigHandler, QAD.AppServer" />
        <sectionGroup name="applicationSettings" type=
"System.Configuration.ApplicationSettingsGroup, System, Version=
2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" >
            <section name=
"QAD.CBFReportingService.Properties.Settings" type=
"System.Configuration.ClientSettingsSection, System, Version=
2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
requirePermission="false" />
        </sectionGroup>
    </configSections>
  <qad.appserver url="appserver://localhost:5162/devcbf93ui">
  </qad.appserver>
  <applicationSettings>
    <QAD.CBFReportingService.Properties.Settings>
      <setting name="SMTPServer" serializeAs="String">
        <value>server01.company.com</value>
      </setting>
      <setting name="ReportFolder" serializeAs="String">
        <value>reports</value>
      </setting>
      <setting name="DaemonCheckInterval" serializeAs="String">
        <value>5000</value>
      </setting>
      <setting name="TCPPort" serializeAs="String">
        <value>4331</value>
      </setting>
      <setting name="LoginUsr" serializeAs="String">
        <value>user</value>
      </setting>
      <setting name="LoginPasswd" serializeAs="String">
       <value>password</value>
      </setting>
    </QAD.CBFReportingService.Properties.Settings>
  </applicationSettings>
</configuration>
```

The .Net Report Service is automatically started when you log in to the application.

The service can also be started with an alternative configuration file location, using the `-configfile configfilename` startup option.

Errors in Report Processing

The Report Daemon monitor lists the report requests and status, and indicates if a report has not been processed correctly.

Processing error details are displayed on the .Net Report Service monitor screen. This monitor is launched using the `QAD.CBFReportingServiceMonitor.exe` file, which is contained in the .Net Report service installation folder.

XML Daemon

The XML daemon processes external data in the form of XML files; for example, invoices are received in electronic form from a supplier. You store the files in a specific directory on the server and the daemon reads and processes them.

The XML daemon differs from the other daemons in that it creates its own requests.

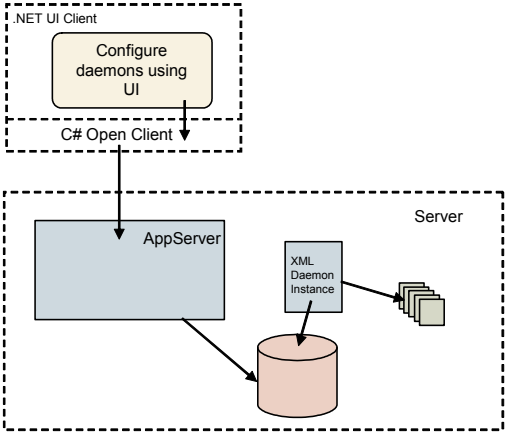


Fig. 5.12
XML Daemon Architecture

The daemon parses the header of an XML document and decides which software component to call to process the data. It then creates the appropriate request. The daemon reads each XML file once.

The XML daemon can process XML files only. You must convert other text formats to XML according to the QAD-defined XSDs before importing them.

You can generate well-formed schema as a starting point for entering your own data by accessing the program that creates a component and selecting Properties from the component menu. The Properties dialog includes a Dump XML button that lets you generate schema to a designated directory.

XML Daemon Actions

The XML daemon reads each XML file to determine what action it must take and how it should handle invalid data.

The XML daemon supports the following activities:

Save. The daemon validates the object and tries to save it. If the validation returns an error, the daemon does not save the object. You can use the Monitor screen to view the error message.

Save/Create Draft. This activity is similar to Save; however, if the validation returns an error, the XML daemon saves the object as a draft, without validation. You can use the Monitor screen to view the status.

Create Draft. The XML daemon does not validate data, and saves the objects as drafts.

Validate. The XML daemon validates the objects, and displays the results in the Monitor screen. It does not save data.

Note For the Create Draft feature to work, the object being created must support Save as Draft. Only a subset of business components can be saved as drafts: Banking Entry, Business Relation, Customer, Customer Finance Charge, Customer Invoice, Journal Entry, Petty Cash, Supplier, Supplier Invoice. Also the use of drafts must be enabled in System Settings.

Use the Action to be Performed field in the Daemon Configure screen to specify the next action for the daemon to perform.

The action applies to all XML files and overrules the action defined in the file if you select the Overrule XML Action field in the XML Daemon, Configure screen.

Configure

The XML Daemon Configure screen lets you configure the XML daemon and differs from the Maintain screen of the other daemons.

The settings in XML Daemon Configure instruct the daemon to convert an XML file into a daemon request and how to process it. Only the settings that are unique to the XML Daemon are described here; the others are the same as the general settings discussed with the History Daemon Configure screen on page 124.

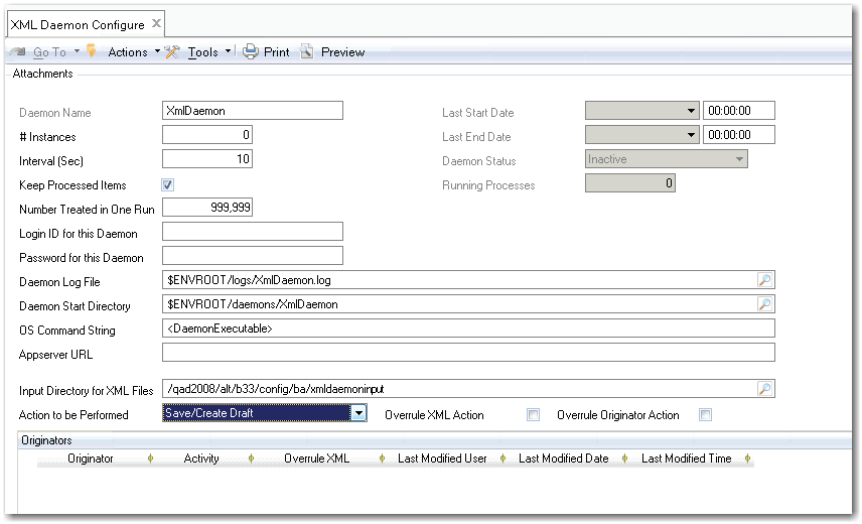


Fig. 5.13 XML Daemon, Configure (36.14.16.6.1)

Field Descriptions

Input Directory for XML Files. Specify the directory that the daemon scans for XML files.

Note You must specify an input directory to enable the daemon process.

Action to be Performed. Optionally, choose the action that the XML daemon must perform on the data in the XML file. If not specified, the action is read from the XML file.

Overrule XML Action. Select this field to overrule the action specified in the XML file with the action defined in the Action to be Performed field.

Overrule Originator Action. Select the field to overrule the action based on the originator and according to the rules defined in the grid.

The Originator field is part of the header section of the XML file and indicates the origin of the file; for example, a source application or external party.

Originator. Specify the originator for which to perform the overrule action.

Activity. Choose the activity to perform. The options are:

- Save
- Save/Create Draft
- Create Draft
- Validate

Overrule XML. Select the field to overrule the action specified in the XML file with the action defined in the Activity field.

XML Daemon Queue

The XML Daemon Queue screen lets you:

- View the results of the daemon activity.
- Correct object errors.
- Relaunch corrected objects to the daemon queue.

Running Daemons on the Command Line

The application control program, `applicationcontrol.p`, can be used to start and stop system daemons on the command line, without launching the application interface. Use daemon-specific parameters to start and stop daemons, and display limited daemon status.

Integrating XML Documents

Data can be imported into QAD Enterprise Edition from an external customer system in a number of ways:

- Using Datasync. DataSync synchronizes static data such as item master data among multiple, distributed databases. The data fields and the specific records to be updated are specified in a synchronization profile. DataSync does not synchronize transaction data or entire databases.

  The data, as documents, is exported from and imported to each database using Q/LinQ. See *External Interface Guide: Q/LinQ* for a complete discussion of importing and exporting documents with Q/LinQ.

- In Excel files, using the Excel integration function available for many components. This function is mainly used for static components, such as GL accounts or cost centers (but can also be used for journal entries).

  The Excel integration function lets you export existing static components to an Excel spreadsheet. You use the spreadsheet as a template in which to enter new data, and re-import the data as new data objects. This function is described in *User Guide: QAD Financials A*.

- As XML files, using the XML Daemon or QXtend Inbound functions. This technique is mainly used for dynamic components, such as transactions.

The Excel and XML-based functions convert the imported file to an Enterprise Application business object, such as a business relation or customer invoice, which can then be processed by standard Enterprise Financials applications. The imported file must contain essential data and conform to a set file structure in order to be processed successfully as a Financials business object.

Each component has an XML schema file, which describes the structure of its data. The structure of XML documents must conform to the schema file of the business component to which they will be converted. You must therefore prepare your document carefully before beginning the integration process.

Planning the Integration

There are a number of scenarios in which a customer uses XML to migrate their data into the Financials application. For example:

- The customer is using an add-on product (for example, a payroll system) and needs to process payroll transactions using Financials functions.
- The customer is using an add-on product (for example, a separate order entry system) and needs to process orders as Customer Invoices.

In these situations, the original data must be converted to XML in order to be integrated into the Financials system.

Before you begin the integration process, you must identify key information:

- Which Financials business component will be used to process the data?
- How is the component structured, and what are its dependant components?
- Which tables are updated by this component and what are its mandatory fields?
- Which component schema files and sample files do you require?
- What are the editing guidelines for creating an XML file from your data?
- Which Enterprise Edition tools do you use to complete the integration?

Identifying Business Components

When you have decided on the purpose of the integration (to create a customer or supplier opening balance, to complete banking entries for posted transactions, or to complete AR or AP payment cycles), you then identify the business components you want to be use to process the imported data.

For example, you may want to process payroll transactions for wages already paid as banking entries. You need to import both the transaction data and the employee data, and may want to import the employee data to

be used as Financials system Employees. The components you will need are Banking Entry and Employee, and the data you import must comply with the schema files for these components.

To identify the name of the business component you need:

1 Right-click on the menu item.

2 Choose Properties.

3 Check the URI field.

For Financials objects, the name is typically in the format

```
urn:cbf:<businessComponent>.<Activity>.
```

You can now view the HTML documentation for this component. See “Tables and Mandatory Fields” on page 148.

You normally use the Business Relations component when importing customer or supplier address and contact information. Business relations contain location and contact information for all addresses defined in the system. They also contain tax details that are directly referenced or used as defaults for customers and suppliers. Therefore, when converting your data to a dataset that can be processed as a Financials component, you should be aware of the component structure.

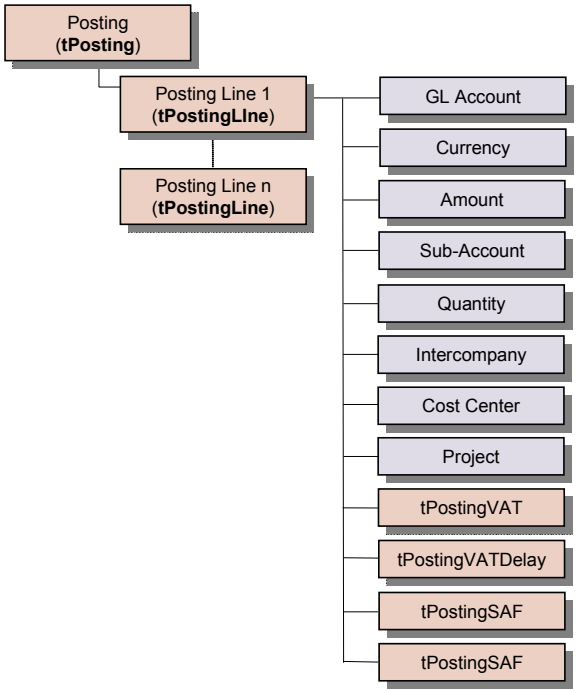
Business Component Structures

Each business component dataset contains data tables. Static components, such as SAF concepts, contain master or parent data tables only, whereas complex components such as transactions contain both parent and child data tables.

For example, the `BJournalEntry` component contains the `tPosting` table, a `tPostingLine` table for each posting line in the transaction, the tax tables `tPostingVAT` for tax postings, and optionally, the SAF posting table `tPostingSAF`.

The component tables also contain fields. In the case of `tPostingLine`, GL account, currency, amount, sub-account, and other accounting fields are defined for each posting line. Each posting line can also optionally have tax tables and SAF tables (when SAFs have been defined for cost centers or projects).

Fig. 5.14
Journal Entry Component Structure



When creating an XML file to be processed as a Financials journal entry, therefore, it is important to identify the fields, tables, and table relations of the `BJournalEntry` component. The XML file you import into Financials must contain the same fields and tables as the component you intend to use. This information is defined in the component schema file.

The schema files for each Financials business component are located in the `<installdir>\qadfin\components\progress\xml\*.xsd` directory.

Tables and Mandatory Fields

Each component contains mandatory fields, which must be given values in order for the component to be validated. Information on these fields, and on the component tables, is contained in the component HTML documentation.

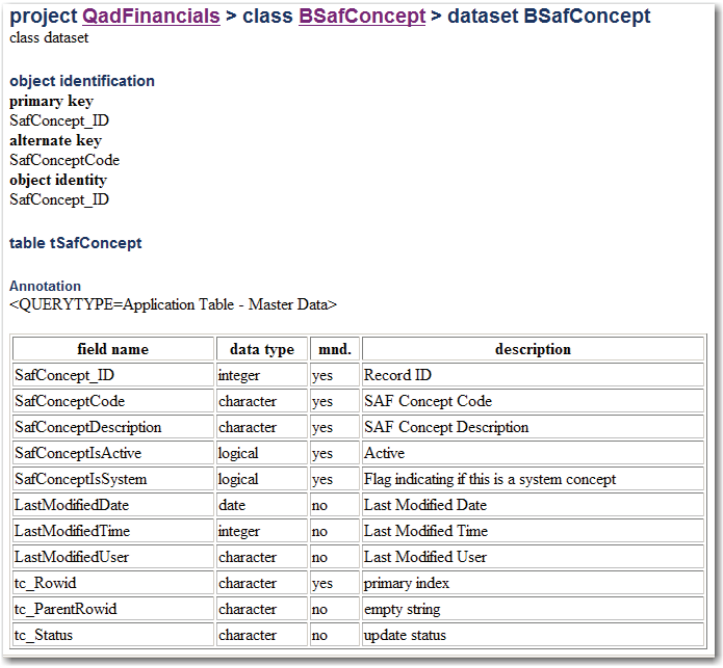
The HTML documentation provided for each Financials business component consists of HTML pages with information about Financials projects, and the components used in projects. The main `index.htm` file for the project lists each component by business area, and you click the link for the specific component to display its own index page.

The component index page is divided into the following areas:

- Public data items
- API queries
- API methods
- Other methods
- Activities

Click the `class dataset` link in the public data items area to display the component dataset attributes.

Figure 5.15 displays the class dataset for `BSafConcept`.



project QadFinancials > class BSafConcept > dataset BSafConcept
class dataset

object identification
primary key
SafConcept_ID
alternate key
SafConceptCode
object identity
SafConcept_ID

table tSafConcept

Annotation
<QUERYTYPE=Application Table - Master Data>

field name	data type	mnd.	description
SafConcept_ID	integer	yes	Record ID
SafConceptCode	character	yes	SAF Concept Code
SafConceptDescription	character	yes	SAF Concept Description
SafConceptIsActive	logical	yes	Active
SafConceptIsSystem	logical	yes	Flag indicating if this is a system concept
LastModifiedDate	date	no	Last Modified Date
LastModifiedTime	integer	no	Last Modified Time
LastModifiedUser	character	no	Last Modified User
tc_Rowid	character	yes	primary index
tc_ParentRowid	character	no	empty string
tc_Status	character	no	update status

Fig. 5.15
BSafConcept Dataset

This component contains the table `tSafConcept`, and contains the following mandatory fields:

`SafConcept_ID`

`SafConceptCode`

`SafConceptDescription`

`SafConceptIsActive`

`SafConceptIsSystem`

`tc_Rowid`

The SAF Concept component is a static component, containing a single table. Complex components, such as BJournalEntry, contain multiple tables, which are linked by parent-child relations. The HTML component documentation lists the component fields and describes them by name, description, datatype and mandatory status. The documentation also lists the component tables, but not the parent-child table relations, which are contained in the `tc_rowid` and `tc_parentrowid` fields in the component schema file, and described in detail in the component HTML documentation.

Component Schema and Sample Files

Each QAD Enterprise Edition calling program is associated with an XML schema file. The schema file provides the definition of the component structure.

The XML schema for a calling program such as Supplier Invoice (`BCInvoice.XSD`) contains all possible data entry fields in the component, and details the component tables. It also includes the information required to start data iterations and information on the calling procedure.

The `tc_Rowid` and `tc_Parentrowid` fields in each table section indicate the parent or child table relations.

Sample values for these fields are displayed in the component sample file.

Supported Components

The following table lists the components for which schema files are currently available:

Table 5.2 Component Schema File

bbudgetgroup	bcashgroup	bcdocument	bcinvoicejournalentry
bcurrency	bddocument	bdebtorcreditrating	bdebtorenduser
bdebtorshipto	bdebtortype	bdinvoice	bdinvoicejournalentry
bdomain	bemployee	bexchangerate.xsd	bexchangeratetype
bgl	bglcalendar	bglmask	bjournalentry
blanguage	blayer	bmirroringgl	bmirroringjournal
bpaymentcondition	bpaymentformat	bpaymentgroup	bperiod
bprofile	bproject	bprojectgroup	bprojectstatus
bpurchasetype	breason	broundingmethod	brole.xsd
bsaf	bsafconcept	bsafstructure	bsafstructurelink
bsettings.xsd	bsharedset	bstate	buser.xsd
buserrole	bvat	bvatbox	bvatgroup
bvatperiod	byearclosing		

Sample XML Files

You can create a sample XML file for all components using the right-click Dump XML option available in browse results screens. The XML dump file is stored on the application server.

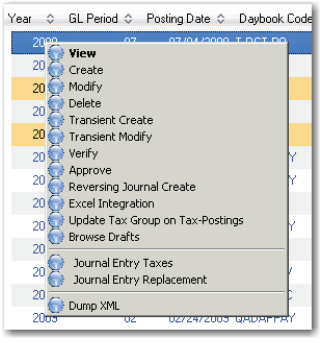


Fig. 5.16 Journal Entry View, Dump XML

The XML files created with this option conform with the schema file for the component, and provide sample values for all fields and table relations for that record. The data contained in the dump XML file is specific to the records you selected.

Creating the XML File

The following section describes rules and editing guidelines for XML file content.

tContextInfo Table

Each component schema file contains a `tContextInfo` table, which must be included in the XML file to be processed. This table is necessary for the system to process the XML file correctly.

`tContextInfo` contains the following fields:

- `tcCompanyCode`

  The `tcCompanyCode` field is mandatory, and indicates the code for the entity in which the data will be processed.

- `tcAction`

  The `tcAction` field is optional. The value of `tcAction` will only be taken into account if the XML daemon is configured so that the default action specified in the daemon configuration can be overridden by the `tcAction` specified in the XML file.

  The possible values for `tcAction` are:

  - `save`

    The data is passed to the back end, and the system attempts to validate and save it. If this is not successful, the load of the XML document fails.

  - `store`

    The data is passed to the back end. The system does not validate it, but saves the data as a draft object.

  - `validate`

    The data is passed to the back end and the system tries to validate it without saving to the database.

- savestore

  The data is passed to the back end, and the system tries to validate it. If the data validates, it is saved to the database. If it does not validate, the data is saved as a draft object. In Create mode it means that the object is not available as a real object in the system yet, and needs to be completed. In Modify mode the modification of the object is incomplete, and it can be completed only by completing the draft object.

- tcOriginator

  The tcOriginator field is optional. (XML daemon only). It can be used by the external party to specify the origin of the data. This can refer to an originator defined within the configuration of the XML daemon, and can indicate if the default action for the originator can be overridden by the value in the XML file.

- tiPriority

  The tiPriority field is optional. (XML daemon only). It can be used to indicate the priority of the XML document. The XML documents in the queue are processed in order of the lowest value set in tiPriority.

- ttRequestStartDate

  The ttRequestStartDate field is optional. (XML daemon only). It can be used to indicate that the object in the XML document can only be processed after a certain date.

- tiRequestStartDate

  The tiRequestStartDate field is optional. (XML daemon only). It is useful in combination with ttRequestStartDate to specify an integer value for the time before which the XML document will not be processed.

- tcComment

  The tcComment field can be used to specify extra comments for the XML document in the queue. (XML daemon only). It can be viewed in the XML daemon monitor.

  Note In QAD 2008.1 Enterprise Edition, the tcComment field can also include:

  ```
  PartialUpdate=<true/yes>
  PartialUpdateExceptionList=t<table>.<field>
  ```

These fields exist as separate configurable fields in later releases and are described in “tlPartialUpdate” on page 154.

- tcCBFVersion

  The tcCBFVersion field is mandatory and should contain the expected Component Builder version number. This field defines the way in which the XML document is constructed. (XML daemon only). The current version is 9.2.

- tcActivityCode

  The tcActivityCode field is not mandatory, but is used to indicate the activity that needs to be executed for the data inside the XML document. If the tcActivityCode field is not specified, or left blank, the system decides the activity. If it cannot find the object based on the alternate key, it assumes the object must be created, and assumes that the activitycode is Create. If it finds the object, it assumes the activitycode is Modify. Possible values are:

  - Modify

    The object is taken in Modify mode. If this is specified, the validation will return an error if the object does not exist (based on the alternate key field search).

  - Create

    The object is taken in Create mode. If this is specified, the validation will return an error if the object already exists (based on the alternate key field search).

  - Delete

    The object is taken in Delete mode. The system tries to find the object, and deletes it.

- tlPartialUpdate

  The tlPartialUpdate field is optional. It can be used to indicate how the system should treat the data that is passed to it. By default the value of this field is false; this means that the data supplied in the XML document is assumed to be complete. This is typical for a Create activity.

If `tlPartialUpdate` is set to true, the data supplied in the XML document is assumed to be incomplete. This allows for the situation in which there may be missing tables or fields. The system then performs a partial update, and does not try to assign default empty values to these fields, or delete child records.

However, you also have the option to identify fields to be deleted. By assigning the `tc_Status` field of a child record to `D`, you indicate to the system that this record is to be deleted.

In this way, you can use the `PartialUpdate` option to supply only relevant field data for integration instead having to supply the complete dataset.

- `tcPartialUpdateExceptionList`

  The `tcPartialUpdateExceptionList` field is not mandatory, and is only read by the system when the `tlPartialUpdate` field is set to true.

  When a field has an unknown value, the system assumes that this field does not need to be updated. Use the `tcPartialUpdateExceptionList` field to specify a comma-separated list of fields that must be updated in the database, even if their value is unknown.

  The syntax for the exception list field names is `t<table-name>.<field-name>`.

  Example `tBusinessRelation.BusinessRelationName1`

Rowid Fields

The values of the `rowid` fields in the XML document are used to map the parent-child relationships between tables. You can assign any values to these fields, while ensuring that the `tc_parentrowid` of the child record matches the `tc_rowid` of the parent. When the system processes the XML files, it uses this matching to map the tables, then assigns its own values to the `rowids` in the database.

Example An XML file containing business relation data contains the tables `tBusinessRelation` and two `tAddress` tables, one of which contains a `tContact` table.

To map these tables, you set the value of the `tc_rowid` of the `tBusinessRelation` record to -1.

The `tc_parentrowid` values of the `tAddress` child records are therefore also set to -1.

The `tc_rowid` values of the `tAddress` tables are set to -2 and -3.

The `tContact` table is the child of the `tAddress` table whose `tc_rowid` is -3. Its `tc_parentrowid` is therefore set to -3, and its `tc_rowid` to -4.

Fig. 5.17
Parent/Child Relations

You must specify the `rowid` fields if you have multiple record requests in your XML request.

Custom Fields

Most tables contain custom fields (for example, `<CustomShort0/>`, `<CustomShort1/>`). Unless you have used these fields for customization, you can remove them from your document.

Last Modified Fields

These fields are generated by the system and can be removed from your document. For example:

```
<LastModifiedDate>2009-01-15</LastModifiedDate>
<LastModifiedTime>9874</LastModifiedTime>
```

```
<LastModifiedUser>mfg</LastModifiedUser>
```

ID Fields

ID fields can be removed from the XML file. These fields are generated by the Financials component when the file is processed. For example, in the following section of a supplier invoice sample XML file, you remove the `CInvoice_ID`, `Company_ID`, `Period_ID`, `Creditor_ID`, `Division_ID`, and `Journal_ID` fields, but retain the `CInvoiceType` and `CInvoiceVoucher` fields:

```
<CInvoice_ID>738175</CInvoice_ID>
<Company_ID>9144</Company_ID>
<Period_ID>16683</Period_ID>
<Creditor_ID>173655</Creditor_ID>
<Division_ID>11438</Division_ID>
<Journal_ID>271904</Journal_ID>
<CInvoiceType>INVOICE</CInvoiceType>
<CInvoiceVoucher>0</CInvoiceVoucher>
```

Primary Key Fields

Primary key fields are a combination of fields that combine to uniquely identify the object in the system.

For example, the primary keys for Supplier Invoice and Journal Entry are a combination of Entity, Posting Year, Daybook and Voucher. The Entity, Posting Year, and Daybook require values in an XML file to be integrated. The Voucher field is generated by the system.

Integrating Multiple Records in an XML Document

You must apply the XML document guidelines described in this chapter to all the records you want to integrate, and you include all records of the same type (for example, all Employee records) in a single XML document.

Processing the XML File

The completed XML file can be processed in one of two ways:

- Using the XML daemon to collect the file and process it.

  The XML daemon processes external XML data. You store the XML file to be processed in a specified directory on the server. The daemon then reads and processes the file. See “XML Daemon” on page 141.

- Using QXtend to convert the XML file to a QDoc to be processed by the QXInbound engine.

  QXtend Inbound delivers inbound data to business component programs in QAD Enterprise Edition. The inbound data is in the form of XML data documents, called QDocs, delivered in either a proprietary XML format or, for a select few programs, as direct code APIs. QXtend Inbound operates over SOAP (Simple Object Access Protocol), and XML documents must be contained in a SOAP envelope in order to be processed.

  For more information on QXtend Inbound and QDocs, see *Technical Reference: QAD QXtend*.

Chapter 6

System Control

This chapter describes essential configuration tasks and administrative options.

Overview

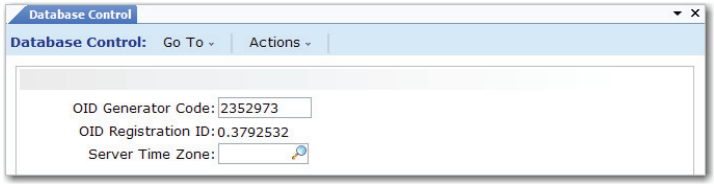
The functions on the System Control Menu (36.24) let you configure and manage settings at the database level. The System Administration function lets you perform the administration tasks required to configure and monitor QAD Financials. These include tasks to:

- Define database control settings.
- Configure system and user settings.
- Set up workflow.

Database Control

Use Database Control (36.24.1) to set the time zone of the database server and to register a code for defining unique record identifiers. You should do this before defining users since the time zone specified here defaults when new user records are created.

Fig. 6.1
Database Control (36.24.1)



OID Generator Code. The OID Generator Code in Database Control is used to assign unique object identifiers (OIDs) to database records for auditing purposes. The code is assigned during system implementation.

Based on the OID generator code, the OID fields in the database are populated using an algorithm that ensures uniqueness across all records, tables, and QAD databases within the company. The value stored in the OID field for each record has the following decimal format:

<date><seq_value>.<registration_id>

Where:

<date> is the server date with format yyymmdd.

<seq_value> is obtained from a Progress database sequence.

<registration_id> identifies the origin of the OID value.

The registration ID is derived from the OID generator code by reversing the digits of the generator code value and placing the decimal point in front of the result.

Server Time Zone. Enter the time zone associated with the server machine for the current QAD database. The system verifies that this is a valid time zone defined in Multiple Time Zones Maintenance (36.16.22.1).

When a new user is created in User Maintenance (36.3.1), the user time zone defaults from the server time zone.

If you are using the optional Service/Support Management module and the Multiple Time Zone option is activated in Service Management Control (11.24) for any domain in the database, this field cannot be modified here. Instead, you must use the Server Time Zone Change Utility (11.21.22.22).

See *User Guide: QAD Service/Support Management*.

System Maintenance

You can use the following activities to manage system settings:

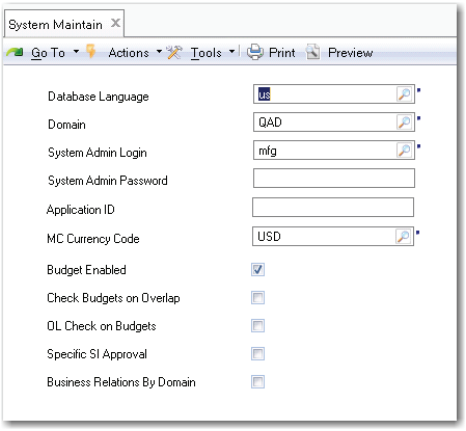
- Use Maintain to configure the main system setup definitions.
- Use Synchronize during initial system installation and to enable new features after a software update.
- Use View System Codes to display the list of values for fields that display valid values in drop-down lists.

Maintain

The System Maintain activity lets you configure the system setup, and describes the system-level values that are applied to all domains and entities.

Note The system settings must be defined before you create domains and entities.

Fig. 6.2
System Maintain
(36.24.3.1)



Field Descriptions

Database Language. Choose the system language from the drop-down list. The system language is used to determine the language of UI elements if a user logs in and no UI strings are found in the language associated with their user record.

Domain. Enter the code that identifies the system domain. By default this is set to QAD. The system domain includes default data that is copied to new domains that are created, including control program settings, default accounts, and generalized codes. The system domain is used as a template for new domains.

Since the system domain is used as a template, you can add data to it or tailor defaults before creating new domains based on it. See *User Guide: QAD Financials A* for details on the system domain.

System Admin Login. Specify a user ID to be used for system startup activities that can be run from a shortcut or as a scheduled task in Windows. These activities include system integrity checking, synchronize, starting the application, and starting the daemons. A valid user ID is required to ensure that a session can be created. By default this is set to the initial user, mfg.

System Admin Password. Specify the password for the system administration user ID.

Application ID. Specify an ID for this instance of the application (maximum 24 characters).

Budget Enabled. Select the field to enable budgeting. If you clear the field, the Budgeting function is disabled, and you cannot create new budgets or modify existing ones.

If your organization does not use budgeting, you should deactivate it to improve system performance.

MC Currency Code. Choose the management currency to use for this database. The management currency is used for management reporting and displaying a consolidated view of activities across domains and entities.

Important Make sure you specify a management currency before you initiate any transactions in the database; it cannot be modified once GL transactions exist.

Specific SI Approval. Select the field if the task of approving supplier invoices is assigned to a specific user role.

After you select the Specific SI Approval field, create a role that corresponds to the approver's user name; for example, SIA–xxxxxx, where xxxxxx is the user name of the approver. Link the new role to the approver's user name.

Choose the SIA–xxxxxx role as approver when you create supplier invoices. When you save the supplier invoices, the system sends them to the Workflow Inbox of user xxxxxx for approval. See "Configuring Workflow" on page 174 for information on setting up the Workflow Inbox.

Note The Role field displays in supplier invoice activities only if you customize the screen to add it. See the chapter on customizing the UI in *User Guide: QAD Financials A* for details.

Business Relations By Domain. Select this field to enable the restriction of business relations to domains. When you select this field, the Domain Restricted field is selected by default on each new business relation you create. When a business relation is restricted to a domain, it can only be viewed and accessed within the domain in which it was created. You can over-ride this setting by clearing the Domain Restricted field in Business Relation Create. See *User Guide: Financials A*.

Synchronize

The Synchronize activity is used to initialize a new system and to activate new features following a software upgrade. It can also be used to update the data in the system such as loading translated strings.

The system uses a number of database tables that must be current for the software version. Some components contain a method that updates the system tables. The Synchronize activity calls and runs these update methods.

Synchronize is typically run directly by the QAD Deployment Tool. You might need to run it for specific updates, such as installing a new language.

Note If you install a new language, you must synchronize both Translations and Areas, Classes, and Activities.

See the release documentation to determine if you must synchronize after you install a new software version. The synchronization must be performed once only.

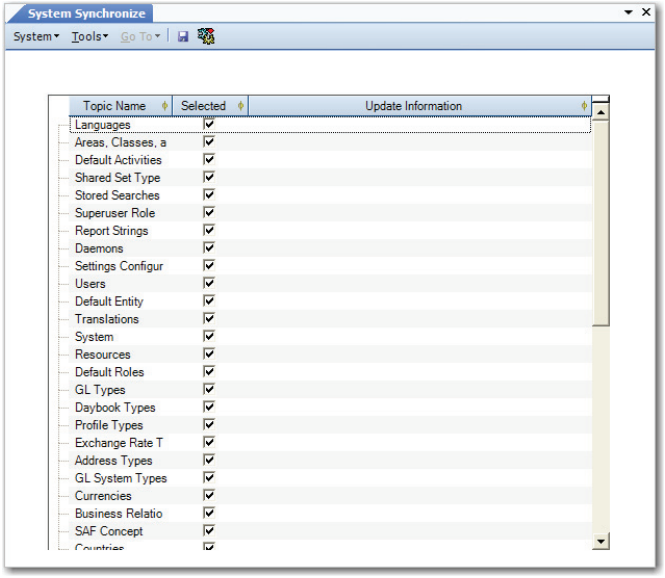


Fig. 6.3 System Synchronize (36.24.3.2)

Field Descriptions

Selected. Select the areas to synchronize.

Selecting specific areas removes the need to synchronize the entire system, and decreases the synchronization time.

Update Information. When the synchronization is complete, this column displays the update status of the area.

View System Codes

The View System Codes activity lets you view the list of values for fields that are validated using a drop-down list. When an integration is set up with an external system, consult this list to ensure that the data sent over by that system is valid.

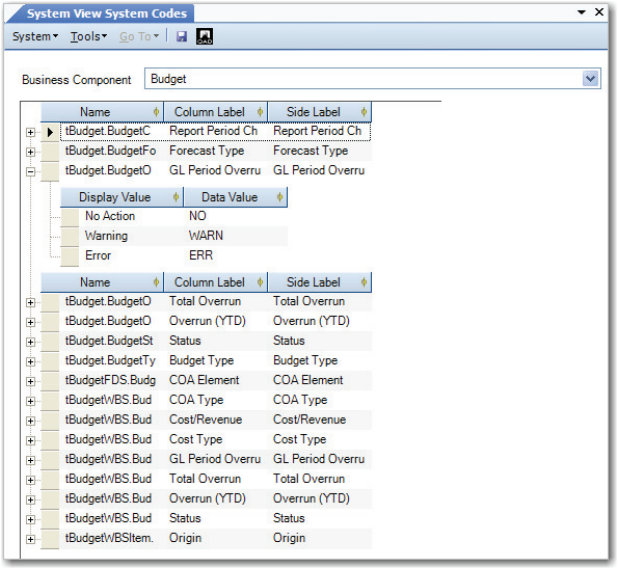


Fig. 6.4
System View System Codes (36.24.3.3)

Field Descriptions

Business Component. Choose the business component for which you want to view the list of values for fields. The name of the business area is displayed in parentheses beside the business component in the drop-down list.

Name. Displays the object name in the format:

t<table name>.<field name>

t indicates that the data type is text.

Column Label. Displays the field name as it is used in reporting lookups.

Side Label. Displays the field name as it appears in the UI.

Display Value. Displays the value that appears in the drop-down list in the UI.

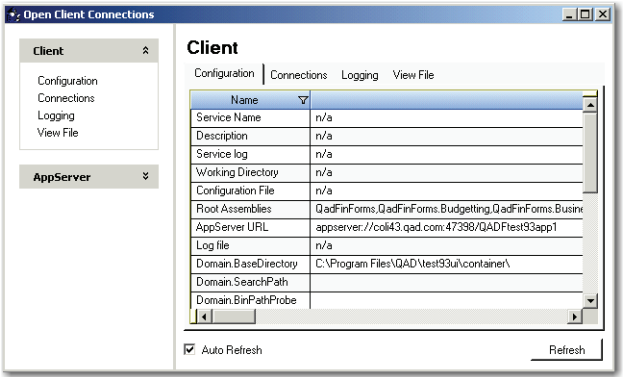
Data Value. Displays the system code as it is stored in the database. Use this value when interfacing from external systems.

System Monitor

Use the System Monitor function to view information about the application. Two sets of information can be viewed: client settings and application server settings. Click on the various tabs to see additional details.

Use this information to monitor activity and help in troubleshooting.

Fig. 6.5
System Monitor, Client Settings (36.24.3.4)



Set Debug Level

The Set Debug Level (36.24.3.5) option lets you set the level of detail displayed in the `ctlog` file. You can display full business code, including parameter values and database queries. You can use the `ctlog` file to run a full trace on component methods and procedures.

Note To run the system log, you must first close all other application screens.

When logging is enabled, you can run the function you want to customize, and then display the execution flow in `ctlog`. The log displays the business methods used in the function and how they are implemented.

The path and filename for the `ctlog` file are set in the `server.xml` file and you can view the filename in the CTLog tab of the Appserver section of System Monitor (36.24.3.4).

Set the debug parameters and select Save to enable the logging.

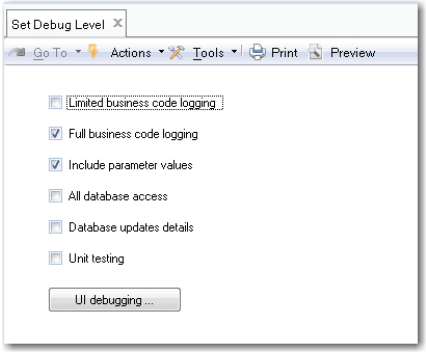


Fig. 6.6
Set Debugging Level

Field Descriptions

Limited Business Code Logging. Select this field to log the start and end of all entry-level business methods. Entry-level methods are called from outside the business logic.

Full Business Code Logging. Select this field to log the start and end of all methods and procedures, internal and external.

Include Parameter Values. Select this field to include parameter values. This field is only available when you select Limited or Full Business Code Logging.

All Database Access. Select this field to include all read or update database queries. The read access logs the executed query statement and the number of records read. The update access logs the action and the `rowid` of updated records.

Database Update Details. Select this field to include details of all database updates (create, modify, or delete). The log entry contains the complete updated record.

Unit Testing. Select this field to log possible memory leaks. This logging checks for:

- Instances that have not been deleted
- Programs that have multiple instances running persistently
- Dynamic objects (datasets + temp-tables + queries + buffers) that have not been deleted
- Connected appservers
- Connected sockets

This logging also lists all persistent procedures and a total count of all database read access events.

UI Debugging. Click to display the View Performance Results screen and enable UI debugging.

UI Debugging

The UI debugging option lets you log UI methods and procedures. Select the Enable UI Logging field to begin logging, and click Refresh to view UI activity. Click Export to export the currently displayed information to a text file for analysis.

Server-side calls are indicated in red, and UI calls are indicated in blue. Darker red and blue lines indicate calls that take longer than 500 milliseconds. Each running thread is represented by a node in the tree view, and every browse or lookup activated creates a new node. You can add user comments (displayed in orange) to the log.

The Export button allow you to export the log details to a text file

Note UI debugging is activated in system memory, and for performance reasons, you should not enable the option permanently. Enable UI debugging for specific UI activities, export the thread for analysis, and then disable the option.

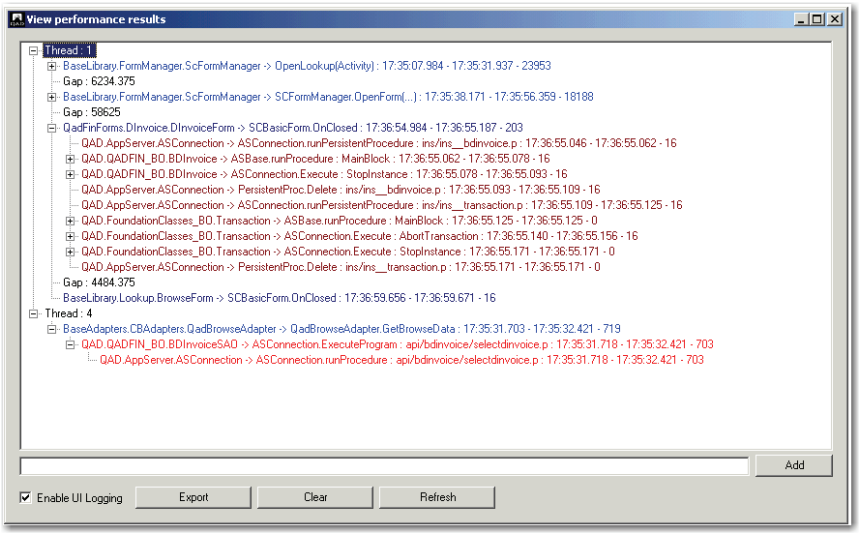


Fig. 6.7
View Performance Results

Configuring System and User Settings

The system and user settings defined in these functions apply only to component-based programs; they do not affect standard Progress programs.

System Settings

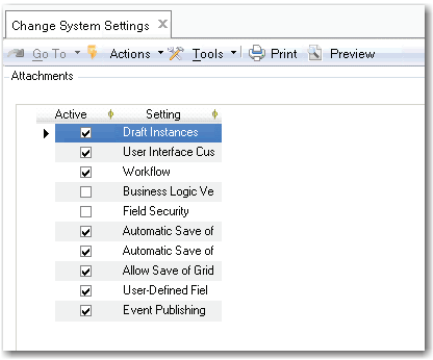
The System Settings function lets you modify system-wide settings that affect the areas of security, customization, and workflows related to component-based functions.

A subset of the settings can also be modified by users, but only if they are enabled system wide. If the setting is disabled, users cannot enable it.

The following settings are enabled by default: UI Customization, Automatic Save of Grid Settings, Automatic Save of Last Used Browse Settings. The others are disabled by default.

Enabling some of the features controlled here may affect system performance.

Fig. 6.8
Change System Settings, (36.24.5.1)



Field Descriptions

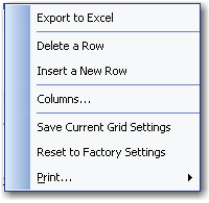
Allow Save of Grid Settings. Disable this if you do not want users to be able to choose to save grid settings. When this is disabled, the Save Current Grid Settings option does not display when users right-click in a grid. When this is selected, the option does display.

Enable this if you want grid settings saved only when a user specifies.

This setting interacts with the Automatic Save of Grid Settings. When grid settings are automatically saved, the Allow Save of Grid Settings has no effect.

The following figure shows the right-click grid menu when Allow Save of Grid Settings is enabled.

Fig. 6.9
Change System Settings, Right-Click Grid Menu



When this option is enabled, individual users can disable it in User Settings if they have permissions. When this option is disabled system-wide, it applies to all users.

Automatic Save of Grid Settings. Select if you want the system to automatically save grid settings and reload them the next time users access that function.

Automatic Save of Last Used Browse Settings. Select if you want the system to automatically save each user's last used browse and lookup settings and reload them the next time the user accesses that function.

When this option is enabled, individual users can disable it in User Settings if they have permissions. When this option is disabled system-wide, it applies to all users.

Business Logic Version Checking. Select to ensure that all business components register their versions in the application database. When a component is activated, the system checks the client version against the server-side version registered in the database. Any inconsistencies are reported.

Enabling this setting can be useful in ensuring system integrity. For example, if you have multiple application servers, a patch may be installed on one server and not another. When the client runs the updated component, its version number is registered. Later, if the other server is used where the patch is not installed, the system detects that an older version of the component exists and displays an error preventing the component from being run.

Draft Instances. Select to enable the support of draft instances—records saved without validation—in all functions that support Save as Draft. If you clear the field, the Save as Draft and Browse Draft activities are disabled.

Currently, the Save as Draft feature applies only to the following functions:

- Banking Entry
- Business Relation
- Customer
- Customer Finance Charge
- Customer Invoice

- Journal Entry
- Petty Cash
- Supplier Invoice
- Supplier

Field Security. Select if you want to activate the ability to set permissions and access to individual fields in the UI. When this setting is selected, the Field Security Maintain activity displays on the menu; otherwise, it does not. *User Guide: QAD Security and Controls* describes how to set up field security.

User Interface Customization. Select to enable users with the relevant roles to customize the UI. Customizing the UI is described in *User Guide: QAD Financials A*.

This option is enabled by default. When this option is enabled, individual users can disable it in User Settings if they have permissions. When this option is disabled system-wide, it applies to all users.

Important This option also controls whether browse and grid savings can be saved. When this field is disabled, the browse and grid settings have no effect, since they are considered to be a form of UI customization.

User-Defined Fields. Select to enable users with access to customizing the UI to add user-defined fields (UDF) to customized activities. UDFs can only be added to the UI when the User Interface Customization field is selected.

Workflow. Select to enable ad-hoc workflow. When the field is cleared, the Workflow choice on the Tools menu of component activities is disabled.

Selecting this field is typically combined with selecting the Workflow Inbox. However, you can also notify the recipients of work items through e-mail rather an application Inbox.

See the chapter on the user interface in *User Guide: QAD Financials A* for details on creating and receiving workflow items. See “Configuring Workflow” on page 174 for more details on setting up workflow.

Enabling workflow adds load to the system, which can degrade performance since the system must continually poll for new Inbox objects. To reduce this load, have users that will not be receiving objects in their Inbox disable the feature in User Settings.

Workflow Inbox. Select to activate the Workflow Inbox. When activated, an Inbox area is displayed on the right-hand side of the .NET UI screen. The system polls for new work items sent to a logged-in user's role and displays them in the Inbox. The polling frequency is defined in a client configuration file.

When this option is enabled, individual users can disable it in User Settings if they have permissions. When this option is disabled system-wide, it applies to all users.

Automatic Save of Last Used Stored Search. Select if you want the system to automatically remember the stored search that was last used in a browse and display it with this search loaded the next time the user accesses that function.

When this option is enabled, individual users can disable it in User Settings if they have permissions. When this option is disabled system-wide, it applies to all users.

Enable Persistent Cache. This option caches menu and browse settings from the current session and re-loads them in the next session. Caching stores static data on the client and improves performance.

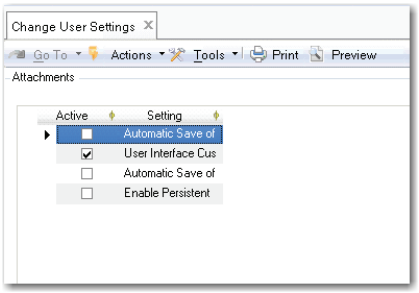
Note Use the Reset Persistent Cache menu option to clear the persistent cache.

User Settings

The User Settings function lets each user configure a subset of settings that are defined at the system level. The settings are effective for the current user only. Users can only modify these settings if they have been enabled system wide.

Disabling Workflow Inbox and UI Customization can enhance performance for a user.

Fig. 6.10
Change User Settings, (36.24.5.2)



Field Descriptions

Each user can modify a subset of the system settings when they are enabled:

Automatic Save of Last Used Browse Settings. See “Automatic Save of Last Used Browse Settings” on page 171.

Automatic Save of Grid Settings. See “Automatic Save of Grid Settings” on page 171.

Workflow Inbox. See “Workflow Inbox” on page 173.

User Interface Customization. See “User Interface Customization” on page 172.

Configuring Workflow

The embedded internal workflow lets users forward or route work items to another role or person in the organization for completion or validation. It serves as a built-in automated approval process that can be applied to any record or document created in the system by forwarding work to the appropriate role. It must be specifically enabled before it can be used.

The workflow is a generic function that can be started from the Tools menu of an individual screen. The user must have a role that has been given access to create workflows in Role Permissions Maintain (36.3.6.5) in order to initiate a workflow.

Workflow is also used with the Scan daemon. See for details.

The record attached to the workflow is assigned to a role. All users that belong to that role receive a notification as an entry in their Inbox. Optionally, when the E-mail Notification check box is selected, users also receive an e-mail notification. The e-mail address is taken from the user record in User Maintenance (36.3.1).

You can display the Workflow Inbox as follows:

1. In Change System Settings, select the Workflow and Workflow Inbox fields. See “System Settings” on page 169.
2. In Change User Settings, select the Workflow Inbox field. See “Set Debug Level” on page 167.
3. Log off from the QAD application; shut down the AppServer.
4. Restart the AppServer.
5. Log in to your QAD application.

   An Inbox icon should now display in the upper-right area of the screen. Roll your cursor over the icon to display the contents; click the icon to open the Inbox.

A system administrator can monitor workflow objects that have been created and never acted on. In some circumstances these objects may need to be removed from the system, for example, when an employee has left the company.

Managing Draft Objects

Enabling the Save as Draft and Browse Drafts options in Change System Settings lets users create draft instances of a subset of components.

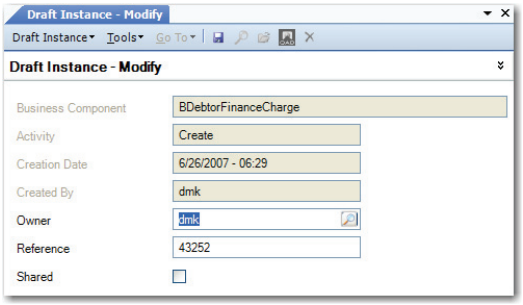
System administrators may need to monitor these draft records to ensure that none are left unattended in the system.

You can use the Draft Object activities (36.24.9) to view, modify, delete, or complete draft objects.

When you modify a draft object, you can assign it to a new owner, change the name by which it is referenced, or modify whether the object can be viewed by others.

After selecting an object in the view, choose Modify from the right-click menu.

Fig. 6.11
Draft Object Modify (36.24.9.3)



Owner. Specify a new owner for this draft instance if needed. The new owner can modify and complete the draft instance.

Reference. Modify the name of the draft instance if needed.

Shared. Select this field if you want this draft instance to be visible to others.

Section 2

Database Administration

This section includes information on system administration functions you can use for regular administrative tasks and to generate reports in your database.

Chapter 7

CIM Interface

This chapter describes how to use database utilities to manage the movement and storage of data in a database.

Introduction

Transferring data can save disk space, increase disk access speeds by compacting fragmented data, and integrate legacy or otherwise noncompatible data with QAD data. There are three basic ways to transfer data into and out of your QAD database:

- Dump or load data files.
- Archive and delete or reload data files.
- CIM load data files.

See page 193.

The first two options are discussed in Chapter 8. This chapter discusses CIM data load, which lets you load data into your QAD database from any source, as long as the data is formatted to match the QAD database schema.

See “Deleting Records through CIM” on page 189.

CIM is typically used to add or modify records in a database. In certain cases, it can also be used to delete records. Only some functions support this feature.

Unlike direct data loads, CIM checks load data for errors and saves unloaded records in an error file for correction and reloading. CIM loads can be run in either batch or continuous mode.

Note CIM must be executed against character-based programs; it is not supported in the QAD .NET UI. Additionally, component-based functions in the .NET UI offer different integration approaches, such as Excel integration, along with the XML and Event daemons.

Using the CIM Interface

The CIM interface loads data through online maintenance programs. All data validation used in these programs during normal data entry is available during a CIM load. Imported data is then made available to other programs.

Most of the data loaded through CIM is loaded into a specific domain. The domain used is the one the user executing the CIM function is currently logged into. If you have access to multiple domains, make sure you are logged into the correct one before beginning the load.

In UNIX, you can use an external load program to load data continuously. These programs can accept input from devices such as barcode readers.

If data is loaded directly into tables using dump/load programs or Progress loads, some tables may not be updated correctly.

Load data into your QAD database using functions on the CIM Interface Menu (36.15). Imported data can come from:

- Any ASCII file that follows the correct conventions.
- The output of programs that run in multiprocessing environments such as UNIX.

See “CIM Data Format” on page 183.

To load a product structure, for example, construct a file that matches the record structure in the product structure master (ps_mstr), then load data into that table. The CIM interface enables you to construct a file of input values for Product Structure Maintenance (13.5), and then validates all the data.

Internally, the CIM Interface operates in two stages:

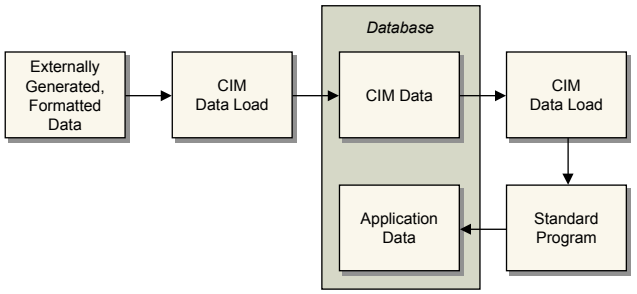
1. CIM Data Load (36.15.1) loads the external data you specify into batch load data tables (bdl_mstr and bdld_det) in the database.
2. CIM Data Load Processor (36.15.2) sends data stored in CIM database tables through the appropriate input screen.

Both the data load and the data processor can be executed as a Progress background session.

Use other functions on the CIM menu to:

- Use CIM Data Load Process Monitor to monitor the load process, as needed.
- Use CIM Data Load Report/Delete to review processing errors and delete processed data, as needed.

Fig. 7.1
CIM Data Load

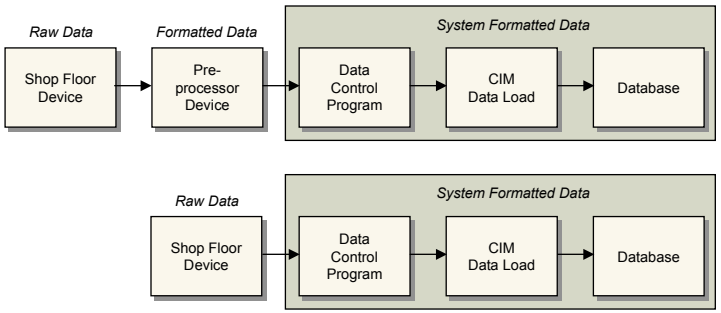


See "CIM Data Format" on page 183 for details.

When CIM Data Load reads a data load group, it creates a record in the batch data load master table and assigns it a unique group ID. This integer record contains the name of the program to receive the data, and the date and time when the record was added. CIM Data Load then creates a record in the batch load detail table for each line of input data from the data load group.

Input from a file can be from either a disk file or a device-character file such as a serial port. If Input File/Continuous Process is selected, CIM Data Load executes the external program named in the Continuous Process Name field. The program controls and formats incoming data and sends its output back to CIM Data Load.

Fig. 7.2
Continuous Data Input



Warning When acquiring external data in real time, run CIM Data Load at the highest possible dispatch priority to ensure that data loss does not occur as a result of competition with other system processes.

CIM Data Format

Each program takes in data in groups. A group typically consists of input fields within a frame. When using a program interactively, you must click Next to move from one group to another.

See "Determining Data for the Input File" on page 185.

Data going into the CIM load must use the rules described in this section.

The `@@BATCHLOAD` key word signals the beginning of the data-load group, consisting of one or more lines. Program name is the system program that will process the input data. For example, if item data is being loaded, the program name would be `ppptmt04.p` (Item Data Maintenance, 1.4.3).

See "CIM Data Input File Example" on page 186.

All input data contained between each `@@BATCHLOAD` and `@@END` is one group, regardless of how many transactions are specified in the data section.

Limit the number of transactions to 50. Each transaction entry can involve the creation of many records. The more transactions in a transaction group, the more system resources are required for processing, and the greater the likelihood of errors.

An error in one transaction can put all transactions in a group out of sequence and prevent the system from processing that group. In cases where maintaining data integrity is vital and re-creating data difficult, you might limit the number of transactions to one.

Input File Formatting Rules

When creating your CIM input file, follow these formatting rules:

- Use a single line of data for each input request.
- To treat two consecutive input lines as a single line, place a tilde (~) at the end of the first line. The tilde (~) is not required if you create the CIM file in an editor. Place no characters, including spaces, after the tilde.
- Surround character fields with quotation marks.
- At the end of each input group, use a line feed. The end of an input line performs the same function as clicking Next (F1 in character UI). Fields for which there are no data and that come at the end of an input sequence do not require hyphens.

- Type all characters in lowercase, taking care to spell correctly.
- Use a hyphen (-) to Tab through a field, retaining the default or existing value. For example, to accept default data for fields 1, 2, 3, 5, and 7, and enter Yes, 12, and 01/01/02 for fields 4, 6, and 8, enter the following:

  ```
  --- "yes" - "12" - "01/01/02"
  ```

- Format data as it is entered.
- Use a period on a line by itself to indicate End or End-Error.

  For repeated input (that is, multilevel), use the period to go back one level. This executes the End command.

- Use slashes (/) where needed. These are not required.
- Make sure the date format in the CIM file matches the date format specified in the Progress session startup parameters (`-d` parameter).
- Use a caret (^) to indicate a null value.

Input Data Types

Input data is information that you would normally enter from your terminal. The manner in which you enter information in an input file depends on the type of information the field is set up to handle. There are four types of input data:

- Character fields can be alphabetic or numeric but have no mathematical operations applied to them. Descriptions (alphabetic) and customer codes (numeric) are examples of character fields. Surround descriptions with double quotation marks (“ ”). The description is accepted without quotation marks, but may be interpreted as more than one input. If there is a space in the description, you must use quotation marks.
- Fields used in mathematical operations are numeric values. They can contain a decimal point (.) or a negative sign (–), but no other symbols, including commas (,) and dollar signs ($) are allowed. Do not use quotation marks for numeric values.
- Logical fields use Yes/No values and do not require quotation marks.
- Format date fields the way they are formatted in the source field.

Determining Data for the Input File

Each program contains one or more entry groups. Each entry group consists of one or more data entry fields in which data can be entered before clicking Next.

Example In Site Maintenance (1.1.13) there are four entry groups, corresponding to the number of times you must click Next. Although direct correspondence between entry groups and frames is normal, it is not required. The four entry groups are:

- Key field group—site code
- Site data
- Selection option
- Domain data

Each entry group corresponds to one line in a CIM file.

While navigating a program to determine field groupings, use the Tab key to move from field to field, rather than the Return key. The Return key works like the Tab key in all fields except the last field in an entry group, where it executes the Next command. This can be misleading in determining which fields belong to an entry group.

CIM Data Input File Example

```
/* wocimp.p */
/* Program to create CIM input data file for Work Order Receipt Backflush */
DEFINE VARIABLE wonbr LIKE wo_nbr.
DEFINE VARIABLE wolot LIKE wo_lot.
DEFINE VARIABLE woqty LIKE wo_qty_comp.
DEFINE VARIABLE woyes AS LOGICAL INITIAL yes.
DEFINE VARIABLE wono AS LOGICAL INITIAL no.
DEFINE STREAM bf.
OUTPUT STREAM bf TO batchloa.d.
REPEAT:
         PROMPT FOR wonbr wolot woqty.
         wonbr = INPUT wonbr.
         wolot = INPUT wolot.
         woqty = INPUT woqty.
         /* See if work order exists in system. */
         FIND FIRST wo_mstr WHERE wo_nbr=wonbr AND wo_lot= wolot NO-LOCK NO-ERROR.
         IF AVAILABLE wo_mstr THEN DO:
                  /*Identify beginning of record & program used.*/
                  PUT STREAM bf "@@batchload wowoisrc.p" SKIP.
                  /*The work order number and ID.*/
                  EXPORT STREAM bf wonbr wolot.
                  /*qty comp., issue alloc-yes, issue pick-yes*/
         EXPORT STREAM bf woqty woyes woyes.
         /*Component issue - yes.*/
         PUT STREAM bf "." SKIP.
         /*Display items being issued - no.*/
         PUT STREAM bf ".".
         /*Is all information correct - yes. */
         EXPORT STREAM bf woyes.
         /* Qty complete. */
         EXPORT STREAM bf woqty.
         /* Remarks - no. */
         PUT STREAM bf "-" SKIP.
         /*Display item and lot/serial detail - no. */
         EXPORT STREAM bf wono.
         /*Is all information correct - yes. */
         EXPORT STREAM bf woyes.
         /* Please confirm update - yes. */
         EXPORT STREAM bf woyes.
         /* Identify end of record. */
         PUT STREAM bf "@@end" SKIP.
         END.
END.
OUTPUT STREAM bf CLOSE.
```

Creating a CIM Input File

To create a data input file, first determine the program to be used and fields to be updated. The basic steps are as follows:

1. Run the program that is to receive the data and determine the program name. You can also run Menu System Report (36.4.4.2).

   a. In the character interface, use the Ctrl+F key combination to display the program context, including name.

   b. In the .NET UI, right-click on the option in the menus and choose Properties to display program details.

2. Determine the program's key fields. These are typically the first fields, and always let you advance to the next field by pressing Next.

   A good test is to position the cursor in a field, and click Next. Note where the cursor goes. Reposition the cursor in the field, and press Return. If the cursor moves to the same place as it did when using Next, embed Next (Carriage Return) in your CIM file. If the cursor went elsewhere, embed a Return. You could still embed Next if this new cursor position did not lead to any field you want to populate.

   An input file must contain values for key fields, each on a line by itself. This allows the Next command to apply to the appropriate field.

   Note which fields are validated or secured. Do this by typing any character (for example, x) and pressing Enter. If a warning displays, the field is validated or otherwise constrained. Your input file must conform to valid choices for the field. Use the lookup browse for a list of valid entries.

3. Choose non-key fields you want to populate and in what sequence. Note whether Next or Return is required after each entry.

   Not all fields have labels. For example, a two-line description can consist of two separate fields. To determine which lines correspond to which fields, place the cursor in each line and press Ctrl+F to display their field names. You must populate each field with a separate entry in a CIM file.

   Note In the QAD .NET UI, field names display as field tips.

4. Record a template of the CIM input file entries for the first frame.

The following is an example template for Item Master Maintenance (1.4.1):

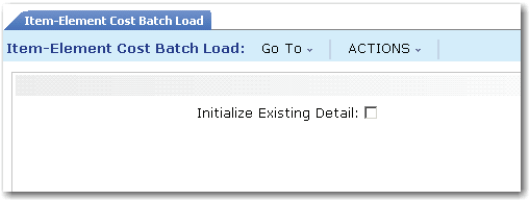
```
@@BATCHLOAD ppptmt04.p
"10-10000"
"EA" "Oasis Cooling System" "Home/Indust Model"
```

Remember, all CIM files start with `@@BATCHLOAD <Program Name>`. The Item Number (10-10000) is a key field and is required. It must be on its own line. The second line represents the next three fields in the entry group.

Follow Item Number with Next. The next line fills in the UM and Description fields. Note that Description is shown as two entries, one populating the first line, one populating the second.

Note There are a few cases where CIM load does not work, such as costing data in Item Master Maintenance (1.4.1). In this case, costing data has to be CIM loaded through Item Element Cost Batch Load (1.4.15); see Figure 7.3.

Fig. 7.3
Item Element Cost Batch Load (1.4.15)



Use the following code to load this data.

```
@@batchload ppptmt04.p
"10-10000"
"Ea" "Oasis(TM) Cooling System" "Home/Indust Model"
"1000" "5/28/1992" "Config" "AC" "DISCRETE" "10-10000" "AB"
.
@@end
```

Error Handling

When the CIM load is completed, CIM Data Load Processor (36.15.2) creates a report showing the groups successfully processed and any processing errors. Groups containing an error are not processed. Troubleshoot errors using the following guidelines:

- Are the values appropriate?

- Is there a line reading: `@@batchload`?
- Is there a line reading: `@@end`?
- Are the data in the correct order?
- Are there any blank lines?
- Are there any misplaced spaces?
- Is there an end-of-line for each data set?
- Does it complete the record?
- Did the first error cause all the others?

Deleting Records through CIM

You can use CIM to delete records created with any of the programs listed in Table 7.1. In each of these programs, an updateable, single-character field, `batchdelete`, exists at the end of the header- and detail-record key frames. This field can be updated only when the program is accessed through a batch process, that is, when `batchrun = true`.

Note In the character UI, when you press Ctrl+F in a field of a program with batchdelete enabled, a pop-up window appears, indicating that you can use batch delete.

Table 7.1 Programs with `batchdelete` Functionality

Menu Label	Program Name
Customer Item Maintenance	ppcpmt.p
Generalized Codes Maintenance	mgcodemt.p
Site Maintenance	icsimt.p
Price List Maintenance	pppimt.p
Price List Maintenance	pppcmt.p
Item Master Maintenance	ppptmt.p
Installed Base Item Maintenance	fsisbmt.p

Because the `batchdelete` value exists at the end of key frames, it does not affect existing CIM input files and can be omitted from these files when not used. Since it is only one character, unlabeled, and hidden, the field also does not change the visible interface.

Creating Input Files to Delete Records

Use these guidelines when creating input files that include deletes:

1. To determine if `batchdelete` is enabled in a particular program, check the list in Table 7.1.
2. To invoke the batch delete functionality, place an `x` at the end of the header- or detail-record key frame line in the input file.
3. Follow the key frame with a blank line consisting of a single hyphen so that the program executes the code that would be executed if an F5 or Ctrl+D has been pressed in the first frame after the key frame.
4. Enter a subsequent line containing the string `yes` as an answer to the Please Confirm Delete prompt displayed for online deletes.

Example of CIM Delete

This sample CIM input file deletes a Site Maintenance record:

```
@@BATCHLOAD icsimt.p
mysite
--
x
-
yes
@@END
```

Running Multiple CIM Sessions

Any number of CIM sessions can be run at one time. However, two load sessions cannot be opened for a single file. To run two sessions, divide the file.

When running multiple sessions, use CIM Data Load Process Monitor (36.15.4). The monitor shows the state of all existing CIM sessions. Type and Process Session are indexes to the sessions. Enter Process in Type and use (/) to first see all the Process sessions, followed by the Load sessions. If you select Next at the Session field, the current status of the processes displays continuously. The display shows startup time, last transaction time, and selection criteria used when the session was started.

Killing CIM Sessions

Although a CIM session runs under the operating system and can be stopped using operating system commands, this is not advised. When the operating system kills a session, your QAD application is not notified and a record of the session may still display in the CIM Data Load Process Monitor (36.15.4).

The best way to kill a CIM session is to use the Process Monitor. To kill a session, identify the session using the Type and Session fields then press the F5 key in the Session field. A prompt asks you to confirm that you want to delete this record.

If the session was invoked with a low-dispatch priority, your monitor may still display a session after it has been stopped, with a status of Killed. To erase the session from the system, delete it again by putting the cursor on the Session field and pressing F5.

Chapter 8

Database Management

This material covers utilities for monitoring database size, performing dumps and loads, reloading archive files, managing database sequences, registering applications, and monitoring license compliance.

Managing Database Size

You can use system utilities for managing the size of your database.

Determining Disk Usage

Use Database Table Size Inquiry (36.16.1) to dump selected tables and review their sizes. The program requires adequate free disk space to run. Reported table sizes may be understated since indexing overhead is not taken into account.

Use Disk Space Inquiry (36.22.13) to display free space for each available disk, in blocks. For most UNIX environments, a block is typically 1024 bytes. For Windows environments, blocks range from 1024 to 8192 bytes. Consult your hardware manuals for exact specifications.

Note These programs must be run from a character user interface.

Fig. 8.1
Disk Space Inquiry (36.22.13)

```
/                (/dev/vx/dsk/rootvol):  956656 blocks   464665 files
/proc            (/proc              ):       0 blocks     4453 files
/dev/fd          (fd                 ):       0 blocks        0 files
/tmp             (swap               ): 7823264 blocks   381700 files
/opt2           (/dev/vx/dsk/crsu03_dg/vol04): 2757000 blocks   948168 files
/dr01           (/dev/vx/dsk/crsu03_dg/vol01):46291736 blocks 12355240 files
/dr02           (/dev/vx/dsk/crsu03_dg/vol02):48571390 blocks 12427225 files
/dr03           (/dev/vx/dsk/crsu03_dg/vol05): 9841572 blocks  2461436 files
/opt.new        (/dev/vx/dsk/crsu03_dg/vol03): 8622328 blocks  2448537 files
/users/cmb      (qcrhp01:/disks/drive2/d7/users/cmb):  654480 blocks      -1
 files
/users/dzn       (qcrhp06:/dr4/users/dzn):  422860 blocks       -1 files
/users/svc       (ohhp04:/home/u3/svc): 1401846 blocks       -1 files
/users/fxd       (ohhp04:/home/u3/fxd): 1401846 blocks       -1 files
/users/pzd       (ohhp04:/home/u3/pzd): 1401846 blocks       -1 files
/users/byd      (qcrhp01:/disks/drive2/d7/users/byd):  654480 blocks      -1
 files
/users/rbe      (qcrhp01:/disks/drive2/d7/users/rbe):  654480 blocks      -1
 files
/qad/mfgpro/85db/etfdb(ohhp40:/dr01/85db/etfdb): 9285970 blocks       -1 files
/users/svb       (ohhp04:/home/u3/svb): 1401846 blocks       -1 files
/users/ncr       (ohhp04:/home/u3/ncr): 1401846 blocks       -1 files
/users/scq       (qcrhp06:/dr5/users/scq): 3373932 blocks       -1 files
```

Freeing Disk Space

There are three ways to reduce the size of a Progress database:

- Use dump/load programs to compact your data. Compacting data can increase disk access speeds significantly. To do this, dump all data from your database, and reload it into an empty database. You need free disk space amounting to about 70% of the total size of your data (`.d`) files. Progress recommends that you dump/load once a year.
- Use delete/archive programs to create free database space. Typically, the largest tables in a database contain history, sales order, and purchase order data. The amount of disk space may decrease if you store the archived data on the same disk.
- Use both dump/load and archive/delete programs. To do this, remove records from the database, dump the remaining data, and reload it into an empty database. You need plenty of free disk space to do this.

See “Deleting and Archiving Data” on page 197.

Dumping and Loading Data

Dump/load programs move the contents of database tables into or out of ASCII files. The dump procedure reads a database table, puts quotation marks around the data value of each field, and places those values in an ASCII file.

Example A record in the user master table (usr_mstr) consists of the following entries:

usr_lang	FR
usr_site	1000
usr_user1	
usr_user2	
usr_user	pxr

One line in the dump file would read:

```
"FR" "1000" """"pxr"
```

You can use dump files as input to other programs after converting the files to CIM input-file format. You can also take output from other programs, convert it to CIM input-file format, and load it into the

See “Using the CIM Interface” on page 180 for details.

database. This assumes the data has the correct form, based on the screen flow and format the CIM input is duplicating. The *Database Definitions* book contains details on specific table formats.

Dump/load procedures are located at 36.16.4 for Windows clients and at 36.16.3 for UNIX environments. Load procedures do not overwrite existing records. You must delete the old data first.

Note Progress and Oracle each provide dump/load and import/export programs, but these programs do not maintain the integrity of data in the QAD database. See the Progress user manuals.

Dump/Load Procedures

To dump/load data:

1. Back up the existing database.
2. Check available disk space. A full dump/load requires free space equaling approximately 70% of existing database size.

See "Determining Disk Usage" on page 194.

3. Log in to your QAD system in single-user mode. You can speed up the dump/load by running multiple sessions of Database Table Dump/Load from multiple terminals.
4. Execute Database Table Dump/Load for the correct range of tables.

   If there is enough free space, select all tables. If there is not, archive the dumped files to a tape, then erase them from the database. Repeat this step as needed.
5. When the dump is finished, copy the standard, empty QAD database (mfg) onto your old database.
6. Load the dumped files back into the database using Database Table Dump/Load.

Data files (`.d` files) reloaded into databases containing data do not overwrite existing records. Files to be loaded must be in a directory specified in your PROPATH. A Progress bulk load is usually faster than a dump/load, but can require an index rebuild.

The system lists load errors in a `.e` file located in the directory you ran the process from.

Deleting and Archiving Data

Delete/archive programs remove selected records from the database, letting you archive them to tape or other media. Each delete/archive screen looks similar to a report criteria input screen. You choose records based on selection criteria. Criteria can include date ranges, document numbers, employee names, and so on.

Table 8.1 lists data that can be deleted and archived.

Table 8.1 Transactions that Can Be Deleted/Archived

Audit Detail	Bulk Pick History	Call/Quote History
Closed Cumulative Orders	Closed Intersite Demand	Closed PO Shippers
Closed Projects	Closed Purchase Orders	Closed Purchase Receipts
Closed Purchase Requisitions	Closed Service Requests	Comment Cross-References
Containers	Contract Revenue	Contracts
Correction Invoices	Cumulative Orders	Customer Schedules
Deferred/Accrued Revenue	Detail Forecasts	Distribution Order Shippers
Electronic Sig Failures	Electronic Signatures	Empty Bulk Picks
Engine History	Expired Call Quotes	Expired Sales Quotes
Expired Warehouse Holds	Family Hierarchies	Field Notifications
Flow Schedules	GL Transactions	Inbound EDI Documents
Inspection History	Installed Base History	Intersite Requests
Intrastat History	Inventory Consumption	Inventory Transactions
Inventory Transactions	Invoice History	Kanban Transactions
Logistics Charges	Logon History	Lot Masters
Master Bills of Lading	Operation History	Operation Plan Simulations
Operation Plans	Outbound EDI Documents	Pending Calls
Performance Data	Physical Inventory Tags	Product Change Orders
Product Change Requests	Product Structures	Purchase Order Shippers
Q/LinQ Documents	Quality Orders	Quality Test Results
Requisitions	Retired Fixed Assets	Routings
Sales Analysis	Sales Order Shippers	Self Bills
Sequence (NRM)	Service Contracts	Service Requests
Service/Repair Orders	Shippers	Subcontract Shippers

Supplier Performance Data	Supplier Schedules	Tracked Documents
Transaction History	Turnaround Data	Uninvoiced Receipts
Verbosity Data	WIP Lots	Work Orders
Zero Inventory Balances		

Audit Detail Delete/Archive

Use Audit Detail Delete/Archive (36.23.1) to delete/archive audit detail information. Unlike other delete/archive programs, this program does not delete each record specified. Instead, for each unique combination of user ID, table, and field, it keeps the latest record and deletes/archives the rest.

To delete and/or archive tables:

1. Back up your database and `.df` files.
1. To safeguard against data archived from a previous product version that has different schema, back up the current database definitions (`.df`) file with each archive/delete run. This lets you reconstruct a corresponding database for data retrieval.
2. Verify record selection.

   Run the delete/archive program without deleting or archiving records. This generates a report showing selected records. Review the report and if records selected for deletion are correct, proceed with the actual archive/delete.
3. Run appropriate historical reports such as Invoice History Delete/Archive (7.13.23).
4. Determine selection criteria for the records being deleted, and run the delete/archive program, selecting both Delete and Archive.

   The program creates a `xxyymmdd.hst` file in the default directory where xx is the record identifier, such as iv for invoices and yymmdd is the archive date.
5. Verify deletion of records from the database.
6. Verify the contents of the `.hst` file using the appropriate operating system command.
7. Back up the `.hst` file to storage media and delete from system.

The delete/archive program does not reduce database size. To reduce database size, use a dump/load program.

Restoring Archive Files

Use Archive File Reload (36.16.5) to reload an archive file after restoring the file from backup media to the system disk.

Fig. 8.2
Archive File Reload (36.16.5)

The reload process puts data from the archive file back into the database exactly as it was when you deleted it. However, if base data has changed, you may encounter errors.

Example You are reloading accounts receivable history for a customer that has been deleted.

Select Allow Errors to continue processing when errors occur. The system lists load errors in a `.e` file located in the directory you ran the process from.

Important Date and time in the stored data are formatted based on the country code associated with the user who archived the data. If a user with a different date and time format reloads the data, load errors and corrupted data can occur.

To avoid these problems, use the same user settings when archiving and reloading the data. Before loading data, use User Maintenance (36.3.1) to temporarily change your country code to match that of the user who archived the data.

See *User Guide: QAD Security and Controls*.

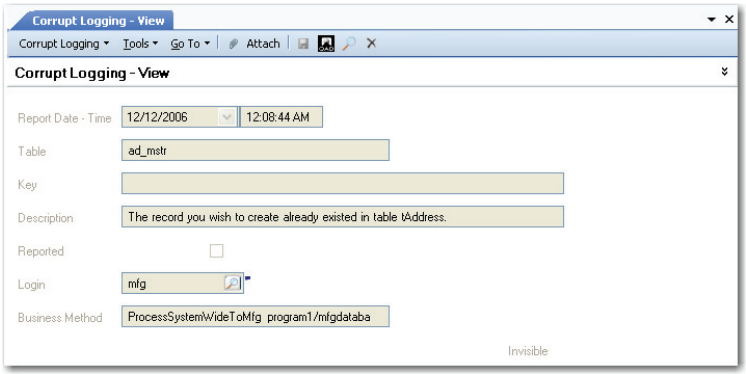
Integrity Logging

The Corrupt Logging function identifies referential integrity problems in the database tables related to component activities, and stores results in a file.

Corrupt Logging Maintenance (36.16.7) has the following activities:

- Mark as Reported: Marks the list of current logs as reported. Previously reported logs are not displayed the next time.
- View: Displays the current list of corrupt logs. Select a row to display detailed information on the log.
- Delete: Clears the logs.

Fig. 8.3
Corrupt Logging, View (36.16.7.1)



Field Descriptions

Report Date-Time. Displays the date and time at which the error was detected.

Table. Displays the name of the table in which the error occurred.

Key. Displays the key field of the record that must be corrected.

Description. Displays a description of the error.

Reported. Indicates whether the error or issue has previously been marked as reported.

Login. Displays the ID of the user who encountered or generated the error.

Business Method. Displays the business component method in which the error occurred.

Registering Licenses

When you receive your QAD software, you also receive license codes for the foundation product and other separately licensed applications.

The license codes identify the license type, version, expiration date and number of days remaining, and number of users, sessions, or transactions for which your site is licensed. Before you can use the system, you must register the license codes.

License registration programs are provided under the License Registration menu (36.16.10). Use the license registration programs to:

- Register newly installed software.
- Upgrade software to add new users or sessions.
- Maintain and report historical license data.
- Report detailed and summary license violations.
- Report license usage and user activity for QAD-conducted audits.

Licensing Overview

QAD licenses software to its customers for use by a predetermined number of users, sessions, or transactions.

The following sections describe concepts associated with license types, license violations, violation types, violation messages, and registration interaction with other QAD modules.

You can use User Monitor Inquiry (36.16.12) or other license-related reports to monitor user activities and application use.

License Types

Two license types apply to users:

Named User. Each unique user ID defined in User Maintenance (36.3.1) is counted as a user. There is no limit on the number of sessions each defined user can run simultaneously. Multiple sessions for the same user ID are counted as one user.

See "Violation Types" on page 203.

Concurrent Session. Each concurrent login is counted as a session. If a single user logs into multiple sessions simultaneously, each login is counted.

User Counts

License usage is monitored regardless of your user interface type, database type (Progress or Oracle), or license type.

For concurrent session license types, the system counts the number of active sessions when you log in and compares the count to the number of licensed sessions stipulated by the license agreement.

If you change to a domain in another database, this process is repeated because changing databases is exiting the current database and starting a new session. Whenever you switch databases, the system stores the logout date and time.

Note If you use the QAD .NET UI, each time you run a program and detach it in a separate window, each window counts as an individual session.

See "Violation Messages" on page 204.

For named user license types, the software counts users when system administrators create new users in User Maintenance (36.3.1) or activate user access to applications in License Registration (36.16.10.1).

License Violations

When the number of users or sessions exceeds the number stipulated by your license agreement, license violations occur.

See "License Reporting" on page 208.

The system stores all license violation occurrences in the database. System administrators and QAD auditors can run reports to view the violation data.

The system responds to license violations with either violation errors or violation warnings. With errors, messages display and the system prevents additional users or sessions. With warnings, messages display, but additional users or sessions can exist and users can still log in to QAD applications.

System administrators can implement enforcement of license agreements by selecting the Enforce Licensed User Count field in Security Control (36.3.24). Setting this field determines whether errors or warnings display and what action the system takes.

See *User Guide: QAD Security and Controls*.

Important The first time a warning displays, you can continue log in to complete transactions or other processing. If you receive repeated warnings, contact your QAD sales representative or distributor to upgrade your license.

The system prevents users from logging in if the license registration record does not exist for the foundation product in License Registration (36.16.10.1).

Violation Types

The system records the violation types listed in Table 8.2.

Table 8.2 License Violation Types

Violation Type	Description
Date Expiry	Displays information about violations that occur when an application's license registration expires. Only evaluation, demo, or temporary licenses have expiration dates.
Application Usage	Displays information about violations that occur when users do not have access to an application.
License Count	Displays information about violations that occur when the number of users or sessions exceeds the amount stipulated by the license agreement.
Non-Licensed Product	Displays information about violations that occur when users attempt to run applications that are not registered.

Violation Messages

Table 8.3 lists error messages that display when license violations occur.

Table 8.3
License Violation Error Messages

Message	Explanation and Solution
Expired license code	The license code expiration date for this application has passed. Contact your QAD sales representative or distributor to obtain a new license code. Register new code in License Registration (36.16.10.1).
Product registration is not valid	The licence code data in your environment has been corrupted or is missing. Contact your QAD sales representative or distributor to obtain the correct license code; register correct code in License Registration.
Application not available in licensed application master	Your environment license data has been corrupted or is missing. Contact customer support to reload valid license data.
Licensed user limit exceed	This message displays in User Maintenance and License Registration when the number of users exceeds the number specified by the license. System administrators can deactive some users; otherwise, contact your QAD representative or distributor to upgrade your license agreement.
Customer is not licensed to execute this module/product: #	You selected a menu item that is not covered by registered license codes. Contact your system administrator to determine the correct menu items for you to access. System administrators should contact their QAD representative or distributor if the license code is not correct or if they wish to purchase this additional module.
User not authorized to run this application: #	You have not been authorized to run this product. System administrators authorize users to use products in User Maintenance or License Registration.

Message	Explanation and Solution
This product expires in # days on #	The license code for this application expires in the number of days indicated. Contact your QAD sales representative or distributor to obtain a new license code; register correct code in License Registration.
Concurrent session limit exceeded	The application you are attempting to access has a concurrent session license type and the maximum number of active sessions for this application has been reached. If this error displays during login, you cannot log in unless another currently logged-in user logs out.

Interaction with Other System Data

The license registration programs use data from other programs to process, maintain, and report license data.

System administrators maintain defined named users and a list of registered software applications that users are authorized to access in User Maintenance (36.3.1). License registration software uses this information to prevent more active users than the license allows.

User Maintenance also includes information that more clearly defines the user. The system ships with a default set of user types predefined in Language Detail Maintenance (36.4.2). The set includes the employee, customer, and QAD user types. It is important for user count and system monitoring purposes that users are correctly identified in User Maintenance before complete license registration functionality can be used.

See “Language Detail Maintenance” on page 48.

License Registration

Use License Registration (36.16.10.1) to:

- Add a new license code for the foundation QAD software or separately licensed QAD modules.
- Upgrade license codes to add sessions or users.
- Remove license codes.

Fig. 8.4
License Registration (36.16.10.1)

The system requests licensing information after you install your base QAD application or separately sold modules and when you attempt to log in with an expired license.

Use the Tab key to select a license code task:

Add. The Add Product frame displays. Enter the license code for the foundation product or a separately licensed module; then choose OK.

The application name, description, version, license type, and number of licensed users display.

See “Granting Users Access to Registered Software” on page 207.

When you add a license code, you are prompted to enter the IDs of users who can access the application. A list of users who can access the product displays once you enter a user ID.

If you try to add an application that is already registered, the following message displays:

```
Product already installed
```

Edit. The Edit Product frame displays. Use this frame to upgrade your license to increase users or sessions.You must obtain the new number from your QAD representative or distributor.

After you enter the code and choose OK, you are prompted to enter the IDs of users who can access the application.

Remove. The Remove Product frame displays. Enter the license code for the application you want to remove from registration. A prompt displays, asking you to confirm the license removal. If you select Yes, the system records the removal date and time. The application is no longer registered, and users cannot execute any programs that are a part of it. If you remove the MFG/PRO license code, you will be logged out of the system, and users cannot log in.

Granting Users Access to Registered Software

You must grant users access to registered software. If a user who does not have access tries to start an application, either an error or warning message displays depending on the value of Enforce Licensed User Count in Security Control (36.3.24).

Access to applications is granted in one of two ways:

1 Assign access to individual users by selecting registered applications in the Application List frame in User Maintenance (36.3.1).

See *User Guide: QAD Security and Controls*.

2 Activate users for a newly registered application in License Registration (36.16.10.1).

After you successfully enter a license code in the Add Product or Edit Product frames, the system displays the Add Authorized Users frame. You can specify a user ID to display a list of users starting with this ID or enter the word All. When you enter All, the list displays with all users selected, as indicted by an asterisk. You can deselect any that you do not want to include.

Note If the total number of users exceeds the number allowed by the application license, the system makes the first users in the list active. For example, if there are 100 user IDs displayed, but the license agreement for the application is for 50 users, the first 50 users are made active for the application.

If you need to authorize more users than your license allows, system administrators can add users through User Maintenance (36.3.1); however, the software records a violation of your license when you add more users.

License Reporting

Various reports let you monitor application use, the number of logged-in users and sessions, the programs in use, and license violations. You can use the application usage and user count reports to be informed about potential license violations.

In addition to license reporting, you can use User Access by Application Inquiry (36.3.22) to display a list of applications, user access status (active or inactive), and access activation date.

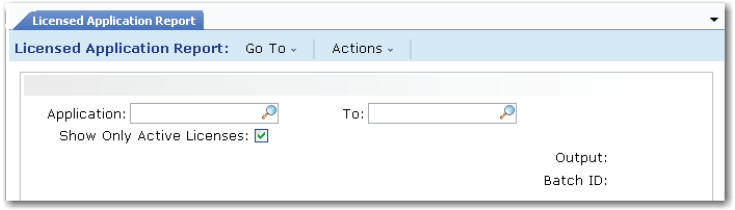
Licensed Application Report

Use Licensed Application Report (36.16.10.3) to display a list of software applications registered in the database.

You can select a range of applications to display. Selecting Show Only Active Licenses displays the current license code for an application. Clearing this field displays information on current and expired license codes for applications. Records display in descending order of the registration date. If there are multiple records for one application, the record with the latest registration date displays first.

The report includes the application description and version, license code and type, number of licensed users, registration and expiration date, user ID of the person who registered the application, audit date information, and any changes to license information.

Fig. 8.5
Licensed Application Report (36.16.10.3)



Application Usage Profile Report

After you install and register an application, the software keeps statistics about your application use. The statistics include:

- Licensed application name

- Menu item executable program name
- Number of times the menu item is accessed
- Percentage of the application in use at the time of reporting

You can use Application Usage Profile Report (36.16.10.8) to display the recorded information for each licensed application.

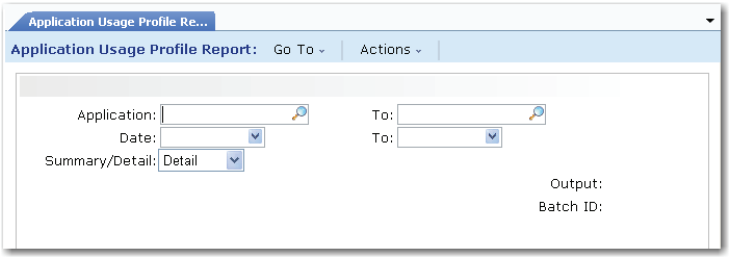


Fig. 8.6
Application Usage Profile Report (36.16.10.8)

You can generate the report in summary or detail format. Summary reports display only the module, access count, and percentage of application use. Detail reports display all recorded information about application usage.

Detailed License Violation Report

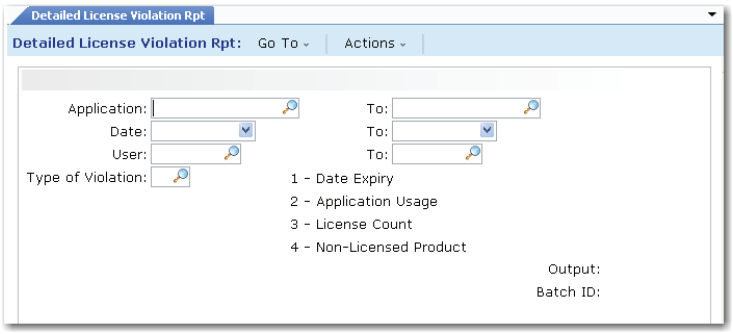
Use Detailed License Violation Report (36.16.10.13) to display information about license violations, including:

- Violation date, time, and error message
- User ID and name of the person who is in violation
- Violation type (for example, application usage or license count exceeded)
- The total number of sessions and users logged in at the time of violation
- Session ID at the time of violation
- Percentage of the application in use at the time of violation

See “Violation Types” on page 203.

Detailed license violation reports let you select a range of registered applications, dates, user IDs, or violation types on which to report.

Fig. 8.7
Detailed License Violation Report (36.16.10.13)



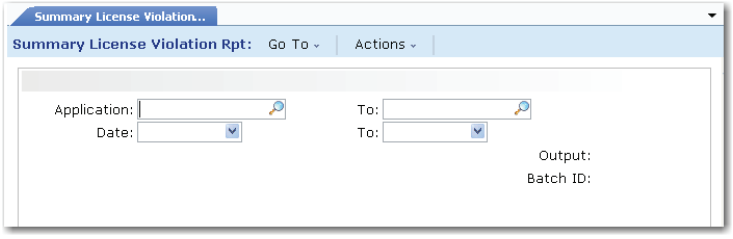
Summary License Violation Report

Use Summary License Violation Report (36.16.10.14) to display:

- Application name, version, and license type
- Violation date
- Total number of violations
- Total number of violations by violation type
- Maximum number of licensed users logged on during a period or the *high water mark*
- Total number of licensed users

Summary license violation reports let you specify the application and the period you want the report to cover.

Fig. 8.8
Summary License Violation Report (36.16.10.14)



If you do not specify an application, all violations for all applications display. If you specify an application, but no dates, all violations for that application display.

If you run either report and there are no violations to report, the following message displays:

```
No violation observed.
```

Audit Reporting

QAD provides programs for QAD auditors to use when the auditors gather statistical information on customer use. The programs are not accessible to users. The statistical information is for QAD auditing purposes only.

User Monitor Inquiry

User Monitor Inquiry (36.16.12) displays users currently logged in, along with the:

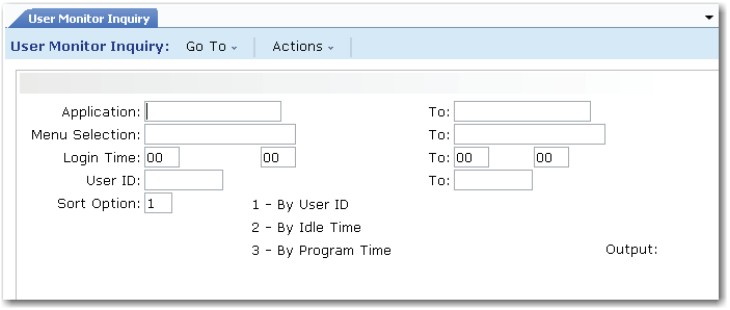
- License type and count for the application
- Program names and menu numbers they are currently executing
- Session ID and user interface type for the session
- Time since they started the current program or menu
- Amount of time they have been idle if no program is selected

This inquiry represents a single point in time, not a continuous system record or audit trail.

By monitoring user and program activity, the system administrator can identify users in violation of license agreements and minimize unnecessary overhead during peak system usage.

You can enter a combination of login time and users, applications, or menu selections to view details of a specific login scenario.

Fig. 8.9
User Monitor Inquiry (36.16.12)



Application. Enter the application name for which you want information to display. You can enter a range of applications by specifying the first application to display in this field and the last application to display in the To field.

Menu Selection. Enter the menu selection for which you want details to display. Leave blank to begin with the first menu matching the other selection criteria.

Login Time. Enter the login time for which you want details to display.

Enter the time based on a 24-hour clock in HH:MM format. For example, enter 1:30 pm as 13:30.

User ID. Enter the ID of the user for whom you want details to display. Leave blank to begin with the first user ID matching the other selection criteria.

Sort Option. Enter the number that corresponds to the way you want to arrange information in the User Monitor Inquiry. You can sort by:

- User ID, which sorts the data in alphabetical order by user ID.
- Idle Time, which sorts the data by the length of time a user has remained on a menu. The user with the longest idle time displays first.
- Program time, which sorts the data by the length of time a user has remained in a program. The user with the longest program time displays first.

Managing Database Sequences

When a unique identifier is needed by a system function program, the system often uses a control field to store the last number used. The system also supports the use of a special schema element called a sequence.

A *sequence* is a database element used to generate a stream of sequential values for assigning unique identifiers to records. Sequences allow fast, accurate numbering, and reduce the amount of time the system spends validating uniqueness.

Note Because the sequence is generated at the database level, records viewed from within a domain may appear to have gaps.

Use Sequence Report (36.16.15) to display a list of sequences defined in the database. The sequence description indicates the database table and field that is updated by the sequence. For example, the description of sequence cmt_sq01 is cmt_det.cmt_indx.

Sequences have the important advantage of speed and reducing the possibility of record locking and contention. However, each sequence is a separate database element, distinct from the table to which it applies. This means that sequences must be initialized correctly whenever you use Database Table Dump/Load.

If sequences are not initialized correctly, Duplicate Unique Key errors may occur when users attempt to create transactions.

If dumping and loading are done as part of installing a software upgrade, sequence initialization is automatically performed by the installation utilities. However, if you perform a dump/load to consolidate tables or increase database size, you must initialize sequences yourself. This is true also if you consolidate data from two different databases.

- Use Database Sequence Initialization (36.16.17) to reset sequences to the highest value plus 1 after loading data.This program works with both Progress and Oracle databases.
- Use Sequence Maintenance (36.16.13) to manually reset a sequence number to a specific value in a Progress database.
- Use Sequence Inquiry (36.16.14) or Sequence Report (36.16.15) to view sequence information.

To guarantee database integrity, perform sequence maintenance:

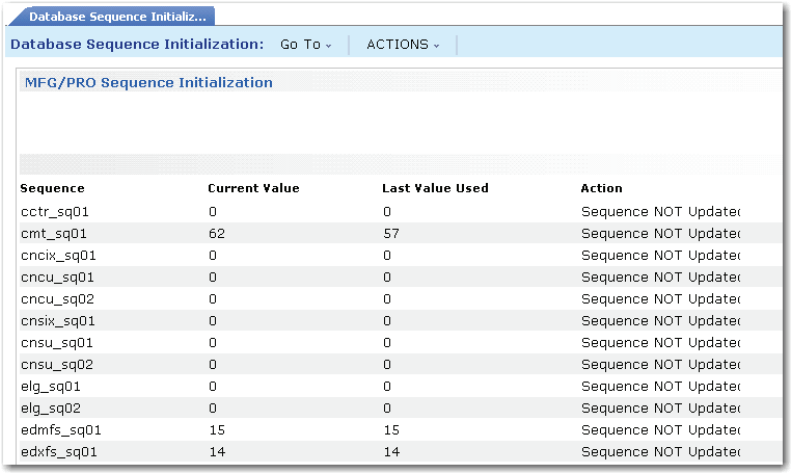
- In single-user mode sessions only
- As a required part of your standard database maintenance

Note To avoid accidental update to sequence structures, use menu security to protect sequence maintenance functions.

Initializing Sequences

Database Sequence Initialization reads each table that uses sequences and sets the sequence number value to the highest number plus 1. This ensures that each new record created has a unique number. This utility initializes sequences correctly in both Progress and Oracle databases.

Fig. 8.10
Database Sequence Initialization (36.16.17)



Maintaining Sequences Manually

See “Maintaining Sequences in Oracle” on page 218.

Maintain sequences manually or through the CIM interface. Maintenance includes:

- Dumping—outputting the current sequence value to a file
- Loading—reading a sequence value from a file
- Updating—manually updating a single sequence

Maintain sequences in Sequence Maintenance (36.16.13). Sequence Maintenance works with Progress Relational Database Management System (RDBMS) only. Oracle dataservers are not currently supported.

Log File Name. The name of the error log file.

Directory. The operating system (OS) directory where you want to store the file.

After you enter the log file name and directory and click Next, a second Sequence Maintenance frame displays. Enter data for the following fields in this frame:

Sequence Name. Specify the sequence or set of sequences to be maintained. Leave blank to specify all sequences.

Note A time stamp is added to the log at the beginning of each session, so session history can accumulate. After a maintenance session, check the log for errors.

Maintenance Activity. Specify the maintenance activity to be applied to the specified sequence sets. Valid values are:

- 1 to dump. Outputs the current sequence value to an OS file.
- 2 to load. Reads the sequence value from the OS file.
- 3 to manually update. This activity can only be performed when a single sequence is specified. When a set of sequences is to be manually updated, the manual update activity is called once for each.

Activity Directory. For a dump or load, specify the OS directory where the sequence files are located. The direction of the data flow is determined by the activity.

Files are named using the name of the sequence with the file extension `.d`. For example, the sequence `tr_sq01` is dumped to a file named `tr_sq01.d`.

When you specify manual update, the system displays the Manual Update frame. Specify the following in the frame:

Original Sequence Value. This field displays the value of the sequence before the user's update was applied.

Current Sequence Value. This field displays the current sequence value.

Note Sequence Maintenance generates a report listing current values of all sequences in the database. It can be run at any time and does not impact the content of sequence structures.

User Input. Enter any sequence value within the valid range. The valid range is determined by the system and is part of the schema. An error displays when the value entered is not within the valid range.

Maintaining Sequences Using CIM

See "Using the CIM Interface" on page 180.

Sequences can be maintained using the CIM interface. The content of a sequence represents the last value applied to the sequence by a call from a a system function. This value is not available for processing, since it was consumed by another process.

Values used to update a sequence are validated against a range of acceptable values for the sequence, as established in the database schema. The value of the sequence can be within and including the boundary values. You receive an error message when the range is exceeded.

Limitations of CIM

Some limitations to maintaining sequences through the CIM interface are:

- Sequence maintenance must be performed in a single-user mode Progress session. The integrity of the sequence value is not guaranteed if maintenance is done in multiple-user mode.
- Destructive updates are not permitted. A CIM update cannot overwrite previously created files. Data dumping does not proceed if any elements in the set of sequences conflict with an existing OS file.
- You cannot manually update from CIM. CIM is an automatic process.
- Any error causes the sequence maintenance to fail. When you suspect a sequence maintenance activity failed while processing, you must repeat the entire process. This guarantees that the sequence values are valid.

Sample CIM File Format

A typical CIM file might look like the following illustration:

```
Line 1: <log file> <log directory>
Line 2: <sequence name>
Line 3: <action>
Line 4: <input-output OS directory>
```

<log file>. The name of the file receiving the output log. When an existing log file is specified, the current CIM output is appended to the end of the existing log. The default value is the value of the `mfguser` variable. This has the format of `TMP9999` where `9999` is a four-digit number that uniquely identifies the CIM session. If the `mfguser` value is NULL (“”), the log file is named `mgsqmt03`.

<log directory>. The location where the log file is stored. The blank value NULL (“”) is specified as the default. When a *<log directory>* is not specified, the *<log file>* is placed in the PROPATH.

<sequence name>. Specifies the set of sequences to be maintained. You can specify a single sequence or the entire set. The default value is NULL (“”), indicating all sequences will be maintained.

<action>. Specifies the activity to be performed, either (1) dumping or (2) loading. The default activity is dumping (1).

<input-output OS directory>. The directory in which the sequence files are maintained. The default value is the local directory.

A time stamp is issued to the log file at the beginning of each session. This permits the same log file to accumulate a history of the session logs. All log files have the `.log` suffix.

Example The following is an example of a working CIM file:

```
@@batchload mgsqmt01.p
“sq_err.log” “/qad”
-
2
“/qad/backup”
@@end
```

This file outputs the error log to the directory `/qad` with the name `sq_err.log`. All sequences are maintained. The hyphen (-) indicates that the default value, in this case `all sequences`, is accepted. Number two (2) indicates that the sequences are loaded. The directory in which the sequence files are maintained is `/qad/backup`.

Note Only sequences currently implemented can be maintained using CIM.

Maintaining Audit Trails

The system maintains audit trail for all updates made to sequences using sequence maintenance routines. Each sequence has a separate set of audit entries.

For each updated sequence, the audit trail records original and final values. If the current value is the same as the original value, the system creates only one record.

Maintaining Sequences in Oracle

Normally, you use Database Sequence Initialization to set the starting sequence values in an Oracle database. The following information is provided if you need to manually maintain sequence values in Oracle, which cannot be done using Sequence Maintenance.

The standard sequence definition in Oracle is:

```
CREATE SEQUENCE <sequence name> START WITH <initial value>
INCREMENT BY 1 CACHE 75
```

Where `<sequence name>` is the same as defined in the Progress `df` and `<initial value>` is the starting value specified by the customer.

See “Maintaining Sequences Manually” on page 214.

The initial value of a sequence is set to the highest value found in the field related to the sequence. The content of a sequence is the last value applied by a system function.

Example In a database with no user transaction processing, the maximum value of `tr_hist.tr_trnbr` is `1010`. This value is used as the starting value of the sequence.

As user `qad`, you would enter the following SQL:

```
DROP SEQUENCE tr_sq01;
CREATE SEQUENCE tr_sq01 START WITH 1010 INCREMENT BY 1
   CACHE 75;
```

Component Record Numbering

The Record Numbering function lets you set a starting number for a specific business component, for example, numbers for printed checks. Numbers are always generated by financial year and by entity.

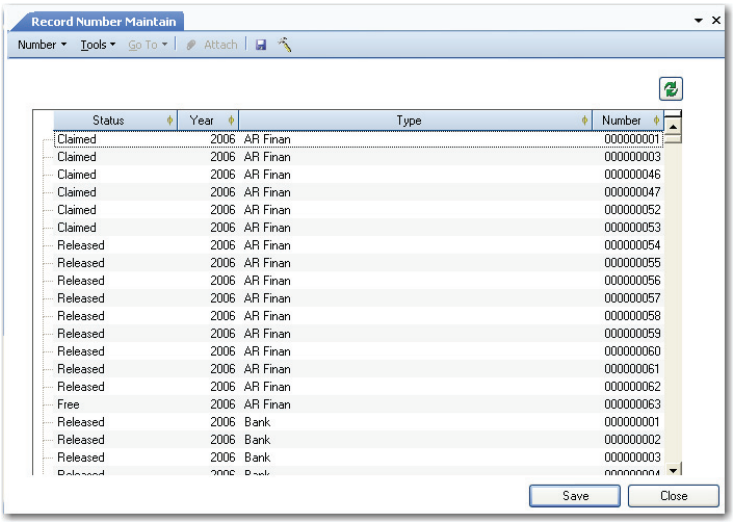
Usually, the number is reset to one at the beginning of a new fiscal year, but you can also set it to a certain value during the accounting year. To add a new number, right-click in the grid and choose Add New Row.

Note It is recommended that only experienced system administrators use this feature. Changes to the numbering system are logged for audit, and unauthorized changes could lead to issues of accounting legality where gaps in the numbering sequence result.

For external daybooks, sequence numbers are generated by the external transaction. However, the numbering function still generates a number series, which will be unused. The system also validates duplicates when numbers are passed from an external application. For transactions originating externally, the Record Number function is only used if a sequence number is not passed from the external system.

The Record Number function supports a single activity, Maintain, which lets you modify the numbering system. You cannot modify an existing number, but you can right-click and add a new row, to reset the sequence for a numbering type.

Fig. 8.11
Record Number Maintain (36.16.21.2)



Field Descriptions

Status. Displays the status of the number. When creating a number record, the field is automatically filled with the status Free.

The numbering statuses are as follows:

Free: The number is being used for the first time.

Claimed: The system selects the lowest available number in the series with the status Free or Released. If the transaction has not been saved, the system assigns the status Claimed to the number. When the transaction is saved, the system removes the number from the list of available numbers. If the transaction is not saved, the system assigns the status Released to the number.

Claimed numbers are currently being used by a running transaction.

Example Numbers 18 and 19 are claimed by running transactions. The transaction that claimed sequence number 18 is abandoned and 18 is released. Sequence number 18 then becomes the next available number, even though 19 is already used.

The Claimed status is also required because, in a client-server environment, if the connection is lost, the number will remain claimed on the client side. This is repaired the next time the server is started and system housekeeping takes place.

Released: The number was used previously for a transaction that was not saved. The system releases the number again for use for the next transaction.

Draft: The number is used in a draft object. If the object is deleted, the system sets the number status to Released. The system removes the number from the list of available numbers when the user saves the draft object.

Year. Specify the accounting year for which you want to create a numbering sequence. The combination of the number year and type must be unique within the entity.

Type. Specify the accounting daybook or number series for which to create the number series. The combination of the number year and type must be unique within the entity.

Number. Specify the first number in the numbering sequence. If you do not specify an initial value, the system starts with 1. Each value in the sequence is incremented by one.

Setting Up Multiple Time Zones

Accommodating variations in local time is a special global business challenge. The Multiple Time Zones Setup menu (36.16.22) lets you create and maintain time zone data.

- Use Multiple Time Zones Maintenance (36.16.22.1) to define and maintain multiple time zones, including the changes required by daylight savings time.
- Use Multiple Time Zones Inquiry (36.16.22.2) and Multiple Time Zones Report (36.16.22.3) to display and report time zone information.
- Use Multiple Time Zones Load Utility (36.16.22.13) to load sample time zone data.
- Use Database Control (36.24.1) to specify a server time zone for the database. See “Database Control” on page 160 for details.

You should restrict access to these programs, with the possible exception of the report and inquiry. Do not change time zone information without carefully evaluating the impact.

Multiple Time Zones Maintenance

Use Multiple Time Zones Maintenance (36.16.22.1) to define and modify time zones.

Note The Multiple Time Zones Load Utility creates sample data upon which you can base your own time zones.

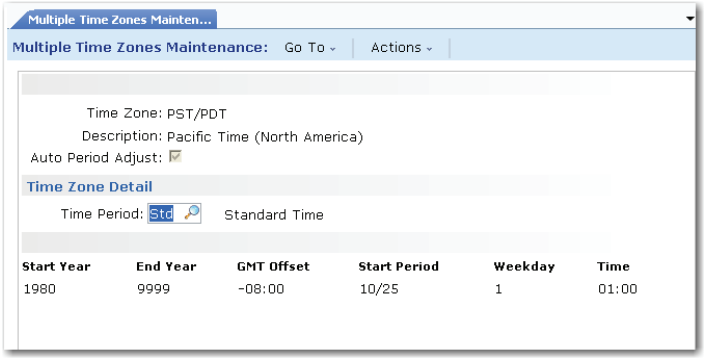
This program supports two ways of setting up a time zone:

- In the simplest format, you can base a time zone on an offset from GMT.
- The system can also track daylight savings time adjustments from a baseline you set.

If you choose the second approach, you must specify when the change in time occurs. You can also use effective dates with time zone information, if the start and end points for daylight savings time only apply for a range of years.

After you define the time zones, you can generate reports with Multiple Time Zones Report (36.16.22.3). Figure 8.12 illustrates Multiple Time Zones Maintenance.

Fig. 8.12
Multiple Time Zones Maintenance (36.16.22.1)



Time Zone. Enter an eight-character label identifying a time zone.

Description. Enter up to 40 characters describing this time zone. The description appears in the time zone lookup.

Auto Period Adjust. This field indicates whether the system should adjust the time zone you are defining for a given period—usually daylight savings time or its equivalent.

Selected: Define the period to be adjusted in the subsequent detail frame.

Clear: Time Period defaults to STD (standard). You cannot change it.

Time Period. This field is editable if Auto Period Adjust is selected. Valid choices are STD for standard time, Day for daylight-saving time, and Sum for summer time. You can define details for two periods: a standard period, and a special adjusted period for daylight savings or its equivalent. This field determines which of the detail fields are required.

Note Set up values for time period as language details to reflect the terms you use.

Start Year. Enter the beginning year of the range associated with this time zone definition. In some countries, the implementation of time zones varies from year to year. Using start and end dates, you can set up multiple records effective at different periods of time.

End Year. Enter the ending year of the range associated with this time zone definition. If you do not know when the current definition ceases to be effective, use an end year such as 9999.

GMT Offset. Enter the actual offset in hours and minutes from Greenwich mean time (GMT) for this time zone. Enter this number with either a plus sign (+) or minus sign (–) indicating the direction of the offset.

GMT is the base for establishing the relationships among time zones and is never affected by daylight-saving time adjustments.

Start Period. When Auto Period Adjust is selected, enter the first day of the week when the change of time occurs in MM/DD format. Use the MM/DD format regardless of the date format you use. For the United States, daylight-saving time normally begins on the first Sunday in April—identified by a start date of 04/01—and ends on the last Sunday in October—identified by a start date of 10/25.

This field, in conjunction with the Weekday and Time fields, identifies precisely when the time change occurs.

Weekday. When Auto Period Adjust is selected, enter a number from 0 to 7 indicating the day of the week—identified by the Start Period field—when the time change occurs. In the U.S., time changes always occur on Sunday (1).

- Enter 0 if the change occurs on the date in the Start Date field, regardless of the day of the week on which it falls.
- Enter a number in the range 1-7 corresponding to Sunday through Saturday if the change occurs on a certain day of the week.

Time. When Auto Period Adjust is selected, enter the exact time of day—identified by the Start Period and Weekday fields—using a 24-hour clock, when the time change occurs. Enter this time in standard time.

In the United States, enter 02:00 when switching from standard time to daylight-saving time, but 01:00 when switching from daylight savings time back to standard.

Multiple Time Zone Load Utility

Use the Multiple Time Zone Load Utility (36.16.22.13) to load a set of sample data based on a snapshot of time zone information. The data assists in the setup process and is a sample only.

After you load this data, verify that the time zones are valid and appropriate for your business practices. Use Multiple Time Zones Report (36.16.22.2) or Inquiry (36.16.22.3) to review definitions and ensure they conform to your requirements. Each organization is responsible for maintaining and updating time zone data to correspond to changing realities and business requirements.

If needed, you can delete existing time zone data and reload the sample data.

Figure 8.13 illustrates the Multiple Time Zone Load Utility.

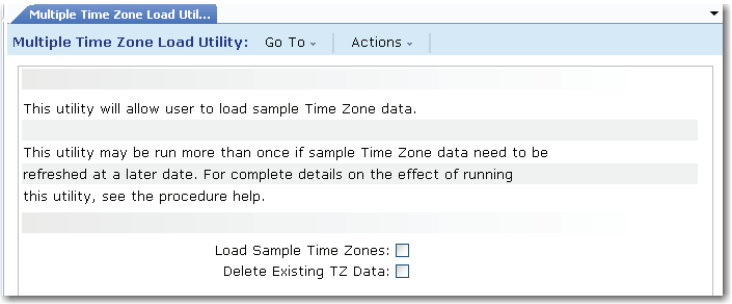


Fig. 8.13
Multiple Time Zone Load Utility (36.16.22.13)

Load Sample Time Zones. Select to indicates you want the system to load sample time zone data. You can use this data as the basis for your own time zone maintenance.

After loading, verify that the time zones are valid and appropriate for your business. Use Multiple Time Zone Report or Inquiry to ensure the definitions conform to your requirements.

Delete Existing TZ Data. The system checks this field only when Load Sample Time Zones is selected. If you are loading time zone data, you can also delete current time zone definitions. Use this feature if you want to reinitialize the sample data.

Setting a Default Time Zone

The server time zone applies to the entire database. Use Database Control (36.24) to specify the time zone of the database. You should do this before defining users since the time zone specified here defaults when new user records are created. See “Database Control” on page 160 for details.

Chapter 9

Reports and Utilities

This chapter includes information on master table audit reports, system cross-references, which let you identify how and where fields and tables are used, application server definitions, operating system commands, and delete/archive utilities.

Generating Master Data Reports

Use the Master Data Reports (36.17) menu to generate audit trail reports showing modifications to master tables, as well as reports showing master comments and control program settings.

Auditing Reports

Use audit trails to track and log which users have made changes to fields in master tables. The system tracks high-level information for changes to all master tables.

See *User Guide: QAD Financials A.*

To maintain detailed information for a critical subset of master tables, select Audit Trail in Domain/Account Control (36.9.24). Table 9.1 lists the database tables that the system tracks.

Table 9.1
Audited Tables

Table	Description
ad_mstr	Addresses
cm_mstr	Customer Data
eu_mstr	End User
eud_det	End User Contacts
flpw_mstr	Field Security
slr_mstr	Site Linking Rules
slrd_mstr	Site Linking Rule Details
usr_mstr	Users
vd_mstr	Suppliers

The audit record contains the user ID, table name, field name, and old and new data values.

Review modifications to tracked master tables with either of the following:

- Use Master Data Audit Report (36.17.1) to print changed records in master tables. The report includes the database table name, current version of the changed record, user ID of the person who made the change, and date.

- Use Master Data Audit Detail Report (36.17.2) to show details about audited changes when Audit Trail is selected in Domain/Account Control (36.9.24). The report includes current and previous versions of the record, with the time and date of any changes.

The system offers other auditing functions:

- Auditing information for unposted general ledger (GL) transactions is maintained when GL Transaction Audit Trail is selected in GL Operational Transaction Control (36.9.13). When selected, any changes made in Unposted GL Transaction Correction (25.13.16) are logged. This data also displays on the Master Data Audit Detail Report.
- Use Change Tracking Maintenance (36.2.22) to track changes to sales order detail fields.

See “Tracking Changes” on page 33.

Other Reports

Use Master Comments Report (36.17.5) to print the text of master comments selected by a range of references and by type and language.

Use Control Tables Report (36.17.6) to generate a report listing the current values defined for all control tables in the system. This report is especially important during implementation. It enables you to verify that settings are appropriate for your business environment.

Using System Cross-References

The System Cross-Reference menu (36.18) contains programs that identify how and where fields and tables are used within standard Progress programs.

Note These functions do not report any information about component-based functions.

System cross-reference activities can be customized to reflect your system setup. This lets you update cross-references when you add or change menu items. If you do not customize the system, you can use the cross-reference as it is.

The cross-reference database requires about 50 MB of disk space, and consists of a set of reports summarizing database relationships such as:

- Which X and Y are used by this Z? X, Y, and Z can be tables, fields, menu items, or programs. *Used* can mean referenced, updated, or called.
- Which database tables are referenced or updated by this menu item?
- Which menu items call this field?
- Which program source files use this include file?

You construct a cross-reference in two steps:

1. Compile the entire system.
2. Build a bill of material from the menu structure.

The end result is a bill of material for each program, in which all programs called by the initial program are components, as well as fields called or updated by those programs.

Cross-reference reports provide different ways of organizing the bill of material.

Background

The core modules of QAD Enterprise Applications consists of approximately 6200 programs that call some 10,000 fields. The programs consist of normal, executable Progress programs (`.p` files) and include files (`.i` files), which can be called from many different `.p` files.

The menu system calls approximately 1400 of those 6200 programs. These called programs call numerous other `.p` and `.i` files. Progress programs can be nested, enabling you to place `.i` files within `.i` files, and so on.

These Progress programs read or change information in database tables, such as the item master (pt_mstr) or the printer master (pr_mstr). The database tables consist of records containing entries in a group of fields.

When Progress is compiled, the list of programs called and the tables and fields read or potentially updated by those programs can be output. This output, along with other utility information, is the source of the cross-reference.

Table, Field, and Menu Reports

The eight cross-reference reports answer such questions as “What does this table, field, message, menu item, or program do?” The syntax is XYZ. For example, the Tables/Fields by Menu Report tells you what tables X and/or fields Y are called by or updated by menu item Z. Similarly, Menu Item by Message Report tells you which menu items X/Y call a particular message Z.

Table 9.2
Table, Field, and Menu Reports

Program Name	Description
Tables/Fields by Menu Report (36.18.1)	Shows what tables or fields are referenced or updated by programs called by a top-level menu. Limit searches further by execution file, database table, and field. Report includes the type of actions performed by the selected programs on each table or field listed. Action types are create, search, update, delete, and access.
Tables/Fields by Program Report (36.18.2)	Similar to 36.18.1, but not limited to menu-level programs. Shows what tables or fields are referenced or updated by the named Progress program. Before running this report for a top-level program, first use Program Source File Report (36.18.16) to generate a list of subprograms called by the program. Then, run Tables/Fields by Program Report for each relevant subprogram.
Menu Items by Field Report (36.18.4)	Shows which menu items call a field or range of fields. Further limit searches by execution file and database table. Shows field name and table, calling menu item, and kind of action performed. Action types are create, search, update, delete, and access.
Menu Items by Table Report (36.18.5)	Similar to 36.18.4, but limited to a database table or range of tables, rather than fields. Shows which menu items call a table or range of tables. Further limit searches by execution file and menu item. Shows table name, menu item, execution file, and kind of action performed. Action types are create, search, update, delete, and access.
Menu Items by Message Report (36.18.6)	Shows which menu items call a particular message or range of messages. Further limit searches by menu and execution file. Shows message numbers and message text.
Messages by Menu Item Report (36.18.8)	Shows all the messages called by a particular menu item. Further limit searches by menu and execution file. Shows message numbers and message text.

For all reports, the top-level selection is the first one searched. To speed up processing, enter values in the top level.

Using Program Reports

Program reports list all programs—`.i` files and `.p` files—called by a menu item.

Table 9.3
Program Reports

Program Name	Description
Programs by Field Report (36.18.13)	Shows all programs that call a particular field or range of fields. Further limit searches by table name and program name. The report includes the following: • The name of the database table to which each selected field belongs. • The names of the programs and subprograms that reference each selected field. • The types of actions performed on selected fields by each program or subprogram listed. Action types are create, search, update, delete, and access. This program may be useful when a field characteristic has been changed and the programmer wants to know what programs are affected. When you generate a report on programs that reference a specific field such as pt_part, programs using phrases like `where so_part=pt_part` are not included in the report.
Programs by Table Report (36.18.14)	Similar to 36.18.13. Shows all programs that call a particular database table or range of tables. Further limit searches by program name. Useful when a table has changed.
Program Source File Report (36.18.16)	Creates a list of program components, or bill of material, for a specified program or range of programs. Shows all component parts, including nested executable files and include (`.i`) files, that are directly called by the specified programs.
Program Run Report (36.18.17)	Creates a multilevel list of components, or bill of material, for a specified program or range of programs. Shows all component parts, including nested executable files and include (`.i`) files, that are either: • Directly called by the specified parent program • Indirectly called by subprograms or include files that are, in turn, called by the specified parent program Use the Levels field to specify the number of levels to include in the report. For example, set Levels to 1 to list only the subprograms and include files directly called from the parent program.

Program Name	Description
Source File Where-Used Summary (36.18.19)	Shows which executable files use a specified source (`.p`) or include (`.i`) file or range of files. Useful if you change an include file and want to see the executable files affected. This program does not list intermediate include files. Use Source File Where-Used Detail (36.18.20) to generate a report on intermediate include files as well as top-level program files.
Source File Where-Used Detail (36.18.20)	Similar to 36.18.19. Shows which executable files use a specified source or include file; also shows intermediate include files. Use the Levels field to specify the number of levels to include in the report. For example, set Levels to 1 to list only the executable files that directly call the specified source or include files.
Run Program Where-Used Detail (36.18.21)	Shows which source (`.p`) and include files (`.i`) reference a specified subprogram. Lists both top-level source files and intermediate include files. Useful if a called program has changed, and you want to check the behavior of the calling programs. Use the Levels field to specify the number of levels to include in the report. For example, set Levels to 1 to list only the files that directly call the specified subprograms.
Program Summary Bill File Create (36.18.23)	Creates a list of components, or bill of material, for a specified program or subprogram, showing all files in the order in which they are called. List includes all subprograms called by the specified parent program, as well as fields updated by those subprograms. Can include multiple calls of the same file. Report output is placed in an ASCII file, where you can manage it using operating system tools. For example, if you change the name of a called program, use Program Summary Bill File Create to make sure you change each instance of it in the source code.

Updating the Cross-Reference

The cross-reference is built by compiling programs, then checking the compiled programs against the menu. If you change menus or change programs, rebuild the cross-reference using Cross-Reference Update Menu (36.18.24).

Rebuild cross-references as follows:

1. If the source has changed, run Cross-Reference Update from Source (36.18.24.1).
2. Run Missing Component Program (36.18.24.15), Missing Menu Execution File (36.18.24.16), and Programs with No Menu Exec (36.18.24.18) reports.

   These reports show any errors in menu or program listings. Missing Menu Execution File Report, for instance, shows names of programs called by the menu that do not exist.
3. After making corrections, add parent-component relationships not included in step 1. Missing parent-components are supplied by the cross-reference.
4. Run Menu Item Cross-Reference Create (36.18.24.3) to link all cross-reference items with the menu.
5. Delete obsolete cross-reference items.

Setting Up Application Servers

Progress AppServer

See the Progress documentation for more information on setting up and using AppServers.

The Progress Open Application Server, or AppServer, is a brokered collection of 4GL engines that can execute Progress programs on the server in response to remote client requests. Each AppServer instance is identified by a unique name, and contains a broker that manages a pool of 4GL engine processes, each of which is available for processing client requests.

The client connects to an AppServer indirectly through the Progress Name Server. This provides for location transparency (and also provides the logical basis for load balancing and failover) since the clients do not

need to know the host and port of the AppServer broker. The client only needs to know the unique name of the AppServer broker, which is used by the Name Server to determine the broker's host and port.

Each AppServer instance can be configured to have its own set of parameter values, such as the PROPATH, database connections, startup/shutdown procedures, and log files. These parameter values are specified in the `ubroker.properties` file, located in the `DLC\properties` directory, where `DLC` is the Progress installation directory.

One extremely useful example of the AppServer is to improve the throughput speed of the processing-intensive task of running material requirements planning (MRP). The AppServer can distribute processing load across multiple threads, dramatically improving performance.

See *User Guide: QAD Manufacturing* for information on MRP.

As an example of how an AppServer can be used, this chapter includes instructions for setting up an AppServer to support enhanced MRP performance.

See "Example: Using an AppServer to Run MRP" on page 237.

Before you can run applications using a Progress AppServer, the AppServer instance must be defined in AppServer Service Maintenance (36.19.1).

Defining the AppServer

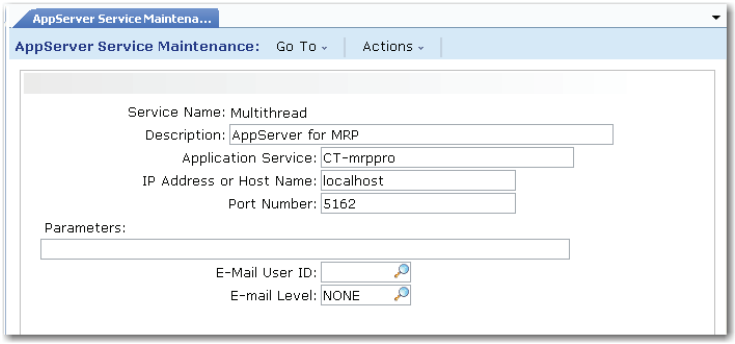
Use AppServer Service Maintenance (36.19.1) to define the information needed to connect to a Progress AppServer.

You can specify a set of standard connection parameters used to connect to this server. Optionally, you can also define server-specific parameters required by the AppServer.

Note The example shown in Figure 9.1 includes the data you would enter to define an AppServer used to improve MRP performance.

See page 237.

Fig. 9.1
AppServer Service Maintenance (36.19.1)



Service Name. Enter a name to identify this application server.

Description. Optionally enter a description of the application server.

Application Service. Enter the name of the Application Server defined in the `ubroker.properties` file during configuration of the AppServer.

IP Address or Host Name. Enter the IP address or host name used as the `-H` parameter when connecting to this application server. This is the IP address or host name of the machine on which the application server is running. If the AppServer is running on the same machine as your QAD database, enter localhost.

Port Number. Enter the port number used when connecting to this application server.

- If you are running a Progress name server, enter the name server port number. The default value is 5162.
- Otherwise, enter the port number on which the AppServer is running.

Parameters. Optionally enter any other parameters required when connecting to this application server.

E-mail User ID and E-mail Level. These fields are not implemented and have no effect on processing.

Example: Using an AppServer to Run MRP

This section shows a practical example of how to set up an AppServer to dramatically improve the performance of MRP.

See *User Guide QAD Manufacturing*.

To use the MRP AppServer, you need to perform three main tasks:

- Modify the `ubroker.properties` file for the AppServer instance.
- Configure the AppServer.
- Start and stop the AppServer as required.

Modify the Properties File

To set up an AppServer to support MRP processing, you must add a set of parameters to the Progress `ubroker.properties` file to identify information about the AppServer instance.

You can modify `ubroker.properties` in two ways:

- Manually edit the file.
- Use the Progress Explorer tool to change parameters through a graphical user interface. The Explorer tool is available only on Windows.

Progress Explorer can also be used to start and stop the AppServer, and for remote administration.

1. Choose Start|Programs|Progress|Progress Explorer Tool.
2. Choose File|Connect.
3. Specify the host name and Admin Server port of the machine you want to administer remotely.
4. Enter a valid user ID for the remote machine and a password, if required.

Configuring the AppServer

Improved MRP performance requires a single AppServer with multiple threads, which is used to execute the programs that process planning data when you run MRP. Use the following instructions to configure that AppServer.

All QAD Installations

See “Additional Oracle Tasks” on page 240.

Use this procedure to configure an AppServer instance for all QAD installations. If you have an Oracle installation, additional configuration tasks are required.

In the Progress example shown below, the name for the AppServer instance is `mt-mrppro`. However, you can use any name, as long as all references to the name are consistent.

Add an entry for the required AppServer instance to the `ubroker.properties` file in the `DLC\properties` directory. You can copy the following text into the file. Be sure to change the parameters to match your environment. Parameter changes are described after the sample text.

Note Separate examples are provided for Progress and Oracle environments.

Progress Example

```
[UBroker.AS.mt-mrppro]
  appserviceNameList=mt-mrppro
  brokerLogFile=$WRKDIR/mt-mrppro.broker.log
  controllingNameServer=NS1
  initialSrvrInstance=12
  maxSrvrInstance=20
  minSrvrInstance=12
  operatingMode=State-reset
  portNumber=50000
  PROPATH=/dr05/mfgpro/pro/qad:/dr05/mfgpro/pro/qad/us/bbi:
    ${PROPATH}${WRKDIR}
  srvrLogFile=$WRKDIR/mt-mrppro.server.log
  srvrMaxPort=50202
  srvrMinPort=50002
  srvrStartupParam=-db /dr05/mfgpro/pro/qad/db/mfgprod -ld qaddb
    -znotrim -trig triggers -db /dr05/mfgpro/pro/qad/db/hlpprod -ld
    qadhelp -db /dr05/mfgpro/pro/qad/db/admprod -ld qadadm -d mdy
    -yy 1920 -Bt 3500 -c 30 -D 100 -mmax 6000 -nb 200 -s 63 -noshvarfix
  uuid=fdf73fbf039907:6ce891fc:ec7f530e95:-7eed
```

Oracle Example

```
[Environment.mt-mrpora]
  ORACLE_BASE=/dr02/apps/oracle/
  ORACLE_HOME=/dr02/apps/oracle/8.1.7
  ORACLE_SID=mrp

[UBroker.AS.mt-mrpora]
  appserviceNameList=mt-mrpora
  brokerLogFile=$WRKDIR/mt-mrpora.broker.log
```

```
controllingNameServer=NS1
environment=mt-mrpora
initialSrvrInstance=12
maxSrvrInstance=20
operatingMode=State-reset
portNumber=54000
PROPATH=
  .:/dr05/mfgpro/qad:/dr05/mfgpro/qad/us/bbi:${PROPATH}:${WRKDIR}
srvrLogFile=$WRKDIR/mt-mrpora.server.log
srvrMaxPort=54202
srvrMinPort=54002
srvrStartupParam=-db /dr05/mfgpro/qad/db/oraprod -RO -znotrim
  -trig triggers -db /dr05/mfgpro/qad/db/mrp -dt ORACLE -U qad
  -P qad -c 250  -d mdy -yy 1920 -Bt 350 -c 30 -D 100 -mmax 3000
  -nb 200 -s 63 -noshvarfix
uuid=59fdf73fbf039907:6302bfc1:ec513ed2fd:-6fd7
```

The parameters of interest are described below. Parameters not listed should generally not be changed from the values given in the example.

Important The first line of the entry specifies the name of the AppServer instance. If this is changed from the name in the example, be sure to change all other occurrences of this name in the other parameters.

- `BrokerLogFile` and `srvrLogFile` are the two log files for the AppServer instance. They should be appropriately named and located in a convenient directory of your choice.
- `PROPATH` is the Progress path used to locate code to run. This should reference the r-code directory where your QAD software was installed.
- `uuid` is a global unique identifier value associated with this AppServer instance. The Progress tool genuuid should be used to generate a value. This tool can be run from the command line and is found in the Progress `DLC\bin` directory.

  Note If you use the Progress Explorer tool to create the AppServer definition, the `uuid` will be generated automatically.
- `appServiceNameList` should match the AppServer instance name that you have chosen, which is listed in the first line of the properties entry.
- `portNumber` is the port number for the AppServer broker for this instance. Its value can be an arbitrary integer, as long as it does not conflict with any port assignments of other applications running on this machine, including other AppServer instances.

- `srvrMinPort` and `srvrMaxPort` specify a range of port values to use for the 4GL engines spawned by the AppServer instance. The range should be large enough to accommodate the maximum number of 4GL engines that can be spawned—specified by the `maxSrvrInstance` parameter—and should not have any conflicts with ports used by other applications, including other AppServer instances.
- `srvrStartupParam` specifies the Progress startup parameters to be used by each 4GL engine that is spawned. The specific DB, host, and service names should match the values that correspond to your QAD database installation.

  Other values should remain as specified in the examples.
- `controllingNameServer` specifies the Progress Name Server instance with which the AppServer broker will register its name. The Progress default is NS1.

See "Defining the AppServer" on page 235.

Since the AppServer broker `mt_mrppro` is used internally by the QAD software, you must use AppServer Service Maintenance (36.19.1) to define an application server connection master record.

Additional Oracle Tasks

If you have an Oracle installation, you must perform the following additional task:

- Add an Environment entry like the example below to the `ubroker.properties` file:

```
[Environment.MRP_ORACLE]
   ORACLE_HOME=/Oracle/OracleAppServer
   ORACLE_SID=YourSystemIdentifier
   ORACLE_BASE=/Oracle
```

Where:

See the Progress AppServer documentation.

- *`/Oracle/OracleAppServer`* is the directory where the Progress AppServer for Oracle has been installed; for example, `/dr01/app/oracle/product/8.1.7`
- *`YourSystemIdentifier`* is the Oracle system ID (SID) for your system

- */Oracle* is the base Oracle directory, which contains version-specific subdirectories; for example, /dr01/app/oracle

Starting and Stopping the AppServers

The AppServer instance configured in the example on page 237 can be administered using the asbman command (located in DLC\bin), which can be invoked from the command line of a DOS window. Click Start|Programs|Command Prompt to launch a DOS window.The DLC\bin directory must be in your PATH environment variable in order to run these commands; alternatively, you can change directories to the DLC\bin directory to run them. On UNIX, these commands are located in the DLC/bin directory, and the user must have Progress administrative privileges to execute them.

Important Make sure that all databases to be connected to the AppServer are running before you start the AppServer.

The command usage is as follows:

- To start an AppServer instance:

  asbman -i *appServerInstanceName* -start
- To stop an AppServer instance:

  asbman -i *appServerInstanceName* -stop
- To check the status of an AppServer instance:

  asbman -i *appServerInstanceName* -query

Example To start the agents for the AppServer name used in the sample ubroker.properties file shown on page 238, type the command:

```
asbman -i mt-mrppro -start
```

After starting an AppServer, use the -query option to check its status, and do not proceed until all of the AppServers are in the available state.

For troubleshooting, verify that the databases that the AppServer connects to are running. Do this by running a Progress client session and trying to connect to the same database servers.

Note For the AppServer instance to run properly, the Progress Name Server must be running. In turn, for the Name Server to run properly, the Progress Admin Server must be running. Although the Name Server and Admin Server are usually configured by default to start up automatically at boot time, it may be necessary to administer them manually. On Windows, these commands are located in the `DLC\bin` directory, and should be run from a DOS window. On UNIX, these commands are located in the `DLC/bin` directory, and the user must have Progress administrative privileges to execute them.

To start, stop, or query the Progress Admin Server, use the appropriate command:

```
proadsv -start
proadsv -stop
proadsv -query
```

Note In a Windows environment, it is recommended that you use Start|Settings|Control Panel|Services to start and stop the Admin Server.

The Progress Name Server will be started automatically during the successful startup of the Admin Server. If it is necessary to start, stop, or query the Progress Name Server (assuming the default NS1 name is used for the Name Server), use the following commands:

```
nsman -i NS1 -start
nsman -i NS1 -stop
nsman -i NS1 -query
```

Using Operating System Commands

The Operating System Commands menu provides four ways to access the operating system and execute commands directly from your QAD application. Use them as a convenient way of viewing and manipulating information.

- Use Exit to Operating System (36.22.1) to invoke a UNIX or Windows command session. To return to your QAD application, enter Exit.

- Use Program/Text File Display (36.22.4) to display the content of an ASCII file, such as a program or print file. If the file is not in the current directory, specify the path. Add this function to the User Menu so that users can generate reports to a file and quickly review the content.
- Use Disk Space Inquiry (36.22.13) to execute an operating system command to display statistics regarding the current database file size.

Using Delete/Archive Utilities

Audit Detail Delete/Archive

To delete data from an audit log, use Audit Detail Delete/Archive (36.23.1). This program works differently from other system delete/archive functions. It does not delete each record specified. Instead, for each unique combination of user ID, table, and field, it keeps the latest record and deletes/archives the rest.

See “Audit Detail Delete/Archive” on page 198 for an exact procedure.

Use this function to produce a report of records before deleting them.

GL Transaction Delete/Archive

General ledger transactions created in operational functions such as purchasing, inventory control, manufacturing, and fixed assets are stored in the unposted transaction table until they are posted. Review unposted transactions using Unposted Transaction Inquiry (25.13.13).

You can delete and archive transactions created in these operational modules using GL Transaction Delete/Archive (36.23.2). Use this program when:

1. You are not using QAD financials.
2. You implemented other modules prior to implementing Financials. Before implementing General Ledger, delete the GL transactions in the unposted transaction table since these are already reflected in the beginning balances you enter.

For a standard implementation, GL Transaction Delete/Archive is used only during early stages of implementation. After transactions of a particular type (PO, IC, WO, or FA) have been posted to the general ledger, no further transactions of that type can be deleted or reloaded.

Command Line Application Control

The application control program, `applicationcontrol.p`, can be used to start and stop applications and daemons (except the Report daemon), and to perform a command-line synchronization. This program lets you perform application maintenance from the command line, without launching the application interface.

The application options include the ability to start the application with or without housekeeping, which deletes inactive data from the database. The Synchronize options include the ability to synchronize all or specific application components

The command line syntax for `applicationcontrol.p` is:

```
_progres -p applicationcontrol.p -param <options> -b > <LogFile>
```

where `<options>` is a string to pass options to the program, and has the format `-<option> [<value] [-<option> [<value]]`.

You can configure the following options:

- URL

  Specify the appserver URL with the following syntax:

  ```
  appserver://<HostName>:<NameServerPort>/<AppServiceName
  ```

- PROPATH

  Indicate the propath to be used by the `_progres` session in cases where the logic is run in the process itself rather than on the central appserver. You indicate the propath when you have not specified an appserver URL.

- ACTION

  Specify an action to be performed. The `<value>` string can be one of the following:

  ```
  StartApplication
  StartApplication_no_housekeeping
  ```

```
StopApplication

StopApplication_no_wait

StartDaemon [<DaemonName>]
```

Use this option to start a new instance of the named daemon. If you do not specify a daemon, the system starts all configured daemons.

```
StopDaemon <DaemonName>
```

This option sends a stop request for all instances of the named daemon.

```
DaemonStatus [<DaemonName>]
```

This returns status information for the named daemon, for example:

```
Daemon Name:                 XmlDaemon
[Status] Status:             Active
[Status] #Running instances: 2
[Status] Proces Ids:         1884, 4856
[Status] Start date of daemon:04/12/2007
[Config] Log file:           /users/xyz/logs/xmldaemon.log
[Config] Startup folder:     /users/xyz/daemons/xmldaemon
[Config] Daemon login:       user
[Config] XML input folder:   /users/xyz/daemons/
                             xmldaemon/input
ResetDaemonConfiguration [<DaemonName>]
```

Note If the daemon is not specified, the system returns the status of all defined daemons in the system

```
ResetDaemonConfiguration [<DaemonName>]
```

By default, this resets all daemons, but by specifying a Daemon name as an extra parameter, it can be limited to one daemon only.

Use `ResetDaemonConfiguration` when you are copying an application database from one environment to another. This ensures that daemon configurations are reset to initial values, and prevents daemons from using previous.

By default, this resets all daemons, but you can limit the reset to a single daemon by specifying the daemon name as an extra parameter.

```
Synchronize [Full|Limited|Topic<XX>]
```

This performs a full or limited synchronization, or for specified topics.

Each topic is represented by a topic number, as listed in Table 9.4. The Synchronization Type column indicates if the topic is automatically included in limited or full synchronizations if you do not specify it by number.

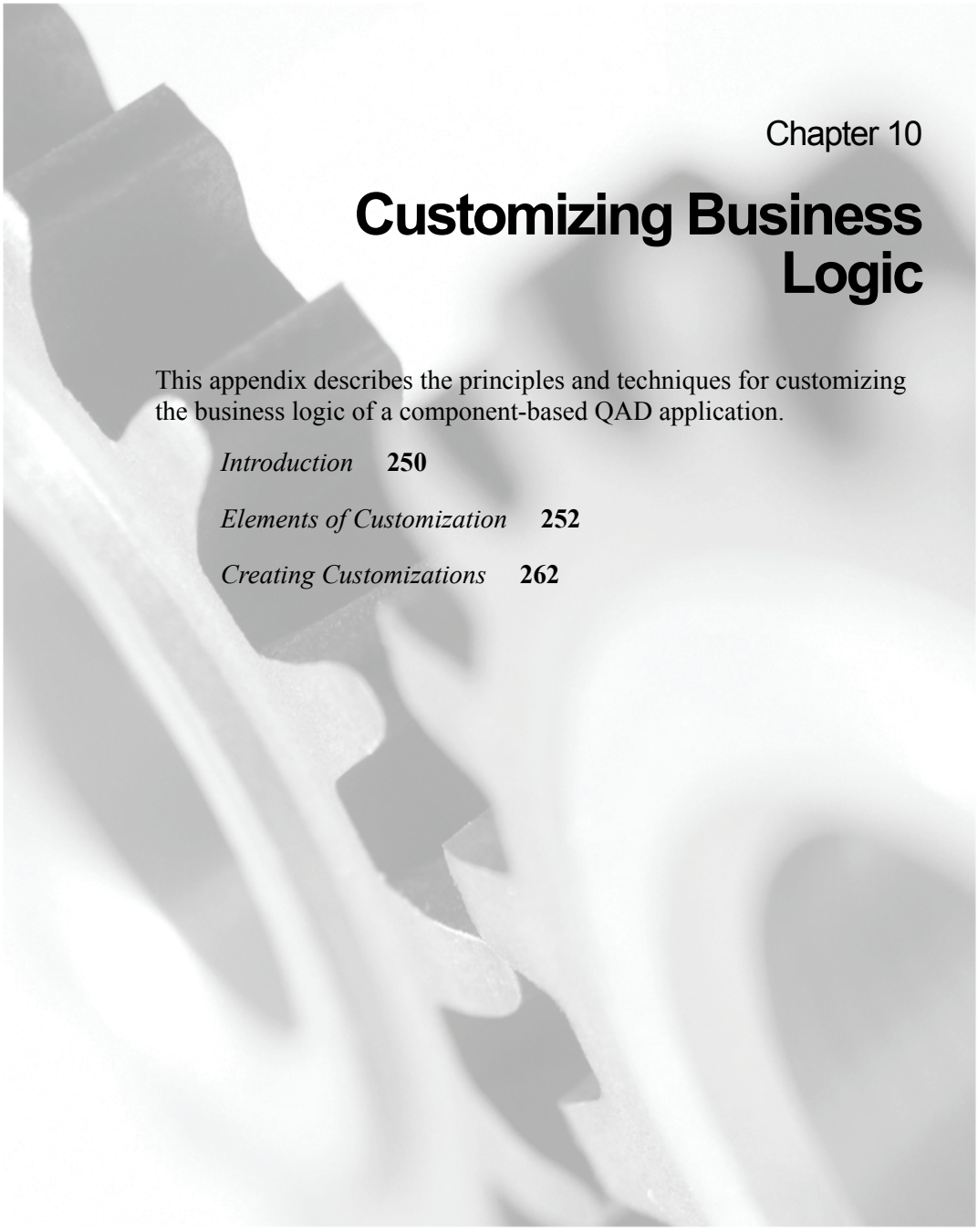
Table 9.4 Synchronization Topic List

Synchronization Topic	Topic Number	Synchronization Type
Languages	1	LIMITED
Areas, Classes, and Activities	2	FULL
Default Activities	3	FULL
Shared Set Types	4	LIMITED
Stored Searches Defaults	5	FULL
Superuser Role	6	FULL
Report Strings	7	FULL
Daemons	8	FULL
Settings Configuration	9	FULL
Users	10	LIMITED
Default Entity	11	FULL
Translations	12	FULL
System	14	FULL
Resources	15	FULL
Default Roles	16	FULL
GL Types	101	LIMITED
Daybook Types	102	LIMITED
Profile Types	103	LIMITED
Exchange Rate Types	104	LIMITED
Address Types	106	LIMITED
GL System Types	107	LIMITED
Currencies	108	FULL
Business Relations	109	FULL
SAF Concept	110	LIMITED
Countries	111	FULL
Rounding Methods	112	FULL
Domains	113	FULL
China Accounting Interface Reports	114	FULL

Synchronization Topic	Topic Number	Synchronization Type
China Accounting Interface Filter	115	FULL
Bank Account Validations	116	LIMITED
Non Taxable Tax Group	117	LIMITED

Example Use the following command to run the application installed to the directory `C:\qad2008` and to check the status of the XML daemon running on this server. You must include the `\lib` and Progress procedure library directories in the propath. The command displays the command results in the log file `vek.log`.

```
_progres -p applicationcontrol.p -param "-PROPATH
C:\qad2008\fin\lib,C:\qad2008\fin\qadfin.pl -ACTION
DaemonStatus[xmldaemon]" -b > vek.log
```



Chapter 10

Customizing Business Logic

This appendix describes the principles and techniques for customizing the business logic of a component-based QAD application.

Introduction

You can customize component-based QAD applications after installation in a number of ways:

- By using the Design Mode feature to customize the user interface during run-time
- By creating user-defined fields and adding them to the interface
- By customizing business component code using a standard Progress editor

These first two options let the user or developer add fields and modify forms in the current UI. You can also use built-in .Net features to customize components such as searches, browses, and reports. These options are described in *User Guide: QAD Financials* and in *User Guide: QAD User Interfaces*.

This appendix describes how to customize business component code for an installed application using a standard Progress 4GL editor.

Customization Overview

QAD application component code is based on a standard code template, and you can create customizations for the following standard application activities:

- Creating an object
- Modifying an object
- Deleting an object
- Browsing an object
- Generating a report
- Retrieving data for a field into the object form
- Retrieving related activities for an activity

You customize any of these activities by writing code that is activated with a publish-subscribe mechanism, in which a customized event is published before or after the component code. A code template (`BComponent.p`) is provided for each business component, which contains the procedure definitions for every before or after event that can be customized, and which you use to implement the customization. The

template code is commented, and to hook the event code into the component process, you uncomment the event and add the necessary component code. This process is called *non-intrusive* customization, because the customized code is added before or after the standard code, but does not intrude on the standard code itself. Template code does not contain any application framework code, but does contain descriptions of methods and method parameters.

Each component-based application also contains a customization controller class `CustomizationController`. This class controls the available customization pieces and their implementation, and ensures that the correct customized code is executed in the right order for the different processes.

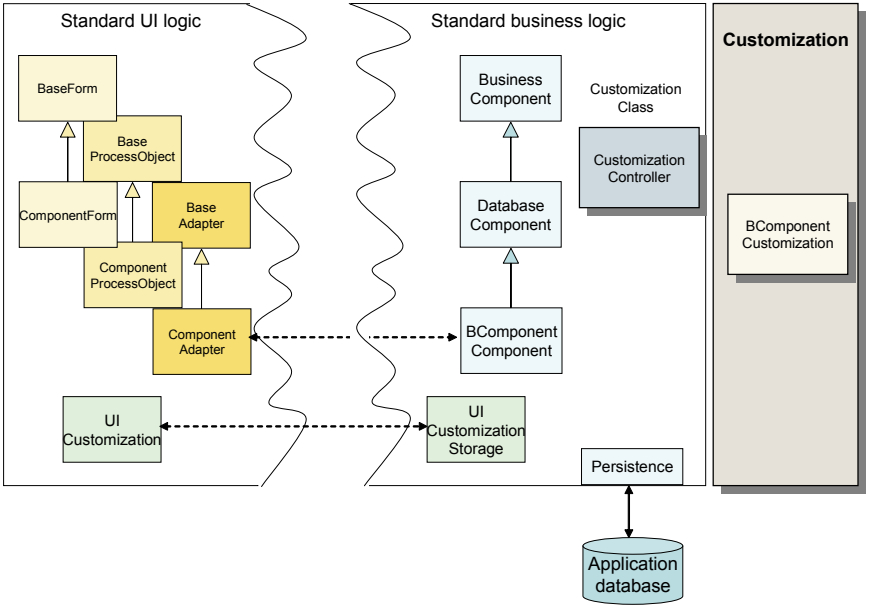


Fig. 10.1
Component Customization

Customizations are run on the same appserver as the internal code, and are portable from one application instance to another. You can also call standard methods and queries from within customized code.

Elements of Customization

The following section describes the code elements used in non-intrusive customization.

Parameters

The parameters of all business methods are stored in the temporary table `t_Parameter`. To access a parameter, you use the field of the same name in `t_Parameter` table. The `t_Parameter` table always contains exactly one available record, which means that you do not need to use `find` or `for each` when using the table. When you assign a value to a field in the table, this value is retrieved at runtime by the standard business code.

You typically assign a value to an input parameter in a *before* event, and assign a value to an output parameter in an *after* event. A value assigned to an output parameter in an *before* event can be overwritten by the standard business logic.

Example The following example uses the input parameter `icPkeys`.

```
/**/
PROCEDURE BItem.DataLoad.before:
if t_Parameter.icPkeys <> ""
then t_Parameter.icPkeys = t_Parameter.icPkeys + chr(4) + "MyKey".
/* load something extra */
END PROCEDURE.
/**/
```

Datasets

All datasets available to the standard business logic are also available to the customization code. These are not just copies of these datasets, but are direct bindings, which means that any dataset changes you make go directly to the standard business logic.

Class Datasets

Each business component has a set of class tables grouped in a class dataset. These tables are used to update the application database. Database updates are first prepared in the class tables, then validated, and when correct, are written to the application database.

Each business component has three class datasets:

- `<classname>O (tables t_o<databasetablename>)`

  Records in this dataset are updated by the business logic. These values are written to the application database.

- `<classname>I (tables t_i<databasetablename>)`

  Records in this dataset represent values read from the database (used for optimistic lock checking), and must not be modified.

- `<classname>S (tables t_s<databasetablename>)`

  Records in this dataset represent the input data of business functions that use the class dataset for input parameters (for example, `SetPublicTables`). These records are first validated and, when correct, are copied into the `<classname>O` dataset. This means that all business validation rules are written to use this dataset.

A buffer `t<databasetablename>` is available in all components that represents the `t_o<databasetablename>`. This makes programming easier and more readable. Since the code is working with the t_o data in most methods, it can be used in most cases, except for the `Validate*` methods, in which the `t_s` data is worked on.

Class Tables

Each class table contains the fields `tc_Rowid` and `tc_ParentRowid`. They are used to define a generic relation between tables of the class dataset:

```
<childtable>.tc_ParentRowid = <parenttable>.tc_Rowid.
```

Important You must not change the value of these fields. Check the documentation of each business component individually to establish the relations between class tables.

Each class table also contains the field `tc_Status`. The value of this field defines the kind of update that is performed on the database, and can be one of the following:

- (empty)

  The record is identical to the record in the database. It is ignored when the database is updated. If you do make changes to the record but forget to set the value of `tc_Status`, your changes are not written to the database.

- `'N'` (new)

  Records with this status are created in the database. Do not use a `create` statement to add records in the class dataset. Instead, use the business function `AddDetailLine` which assigns correct default values to the newly created record.

- `'C'` (change)

  Records with this status are updated in the database. The record to update is retrieved using the value of `tc_Rowid`, instead of the primary key fields. This does not mean that primary key fields can be modified, as these fields may be copied into related tables, and by modifying them, you break the link to those tables.

- `'D'` (delete)

  Records with this status are deleted in the database. The record to delete is retrieved using the value of `tc_Rowid`. If the table is the parent of table relations with delete constraint '`cascaded`', related records are deleted as well.

Error Handling

Error messages to the end-user can be created by running the `SetMessage` procedure in combination with the parameter `oiReturnStatus`.

An error condition is raised when you assign a negative value to `t_Parameter.oiReturnStatus`. When this occurs, standard business logic stops executing and passes the return status to the client that runs the business logic.

Note The `ValidateComponent` and `ValidateBC` methods are exceptions to this behavior. When the return value for `oiReturnStatus` is negative, these methods continue to execute and attempt to validate the component.

The possible values for `oiReturnStatus` are:

- `-1 Validation error`

  The input for the operation was incorrect. The operation could not proceed.

- `-3 Run-time error`

  The operation did not proceed because of a run-time error in the operation code.

- `-5 Fatal error`

  The operation did not proceed because of a permanent error. The application is shut down.

Database Access and Updates

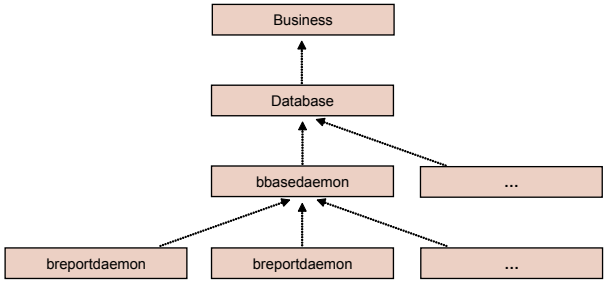
There are no rules or restrictions on database access and updates. Each event is a transaction on its own and all updates to it are committed when the event ends. The exceptions to this are the events on the business method `DataSaveWorker` and the sub-methods `PreSave` and `PostSave` (see “Identifying Events” on page 256). These methods are typically contained in larger transactions, and updates to them are not implemented if the larger transactions fail.

Inheritance

When you implement an event on a business component that has descendant business components, the event is also implemented for all the descendants, except descendants that override the standard business logic for that specific method.

Example

Fig. 10.2
Inheritance Structure



Any customization event implemented on `bbasedaemon` is active when running the component `bxmldaemon`, `breportdaemon` or any other daemon.

Identifying Events

To view the execution flow in full detail, use the system logging functions. Set Debug Level (36.24.3.5) lets you set the level of detail displayed in the system `ctlog` file. You can display full business code, including parameter values and database queries.

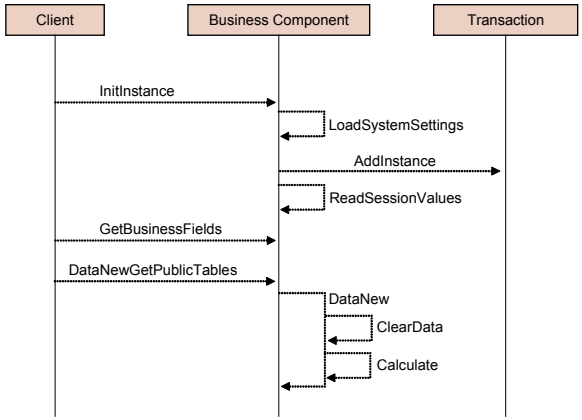
When logging is enabled, you can run the function you want to customize, and then display the execution flow in `ctlog`. The log displays the business methods used in the function and how they are implemented.

See “Set Debug Level” on page 167.

The following business logic flows illustrate the call stack of some of the most used business functions.

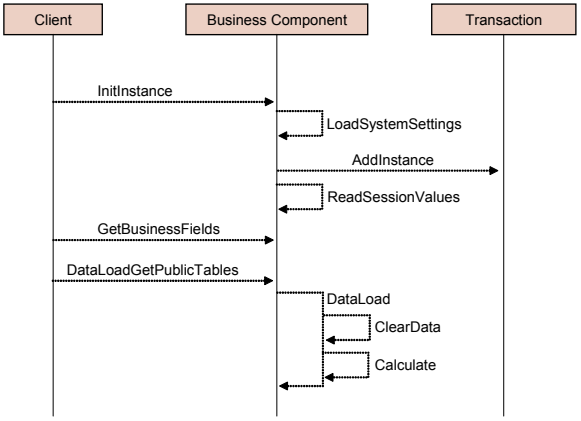
Instantiate a Maintenance Form in Create Mode

Fig. 10.3
Create Call Stack



Instantiate a Maintenance Form in Update mode

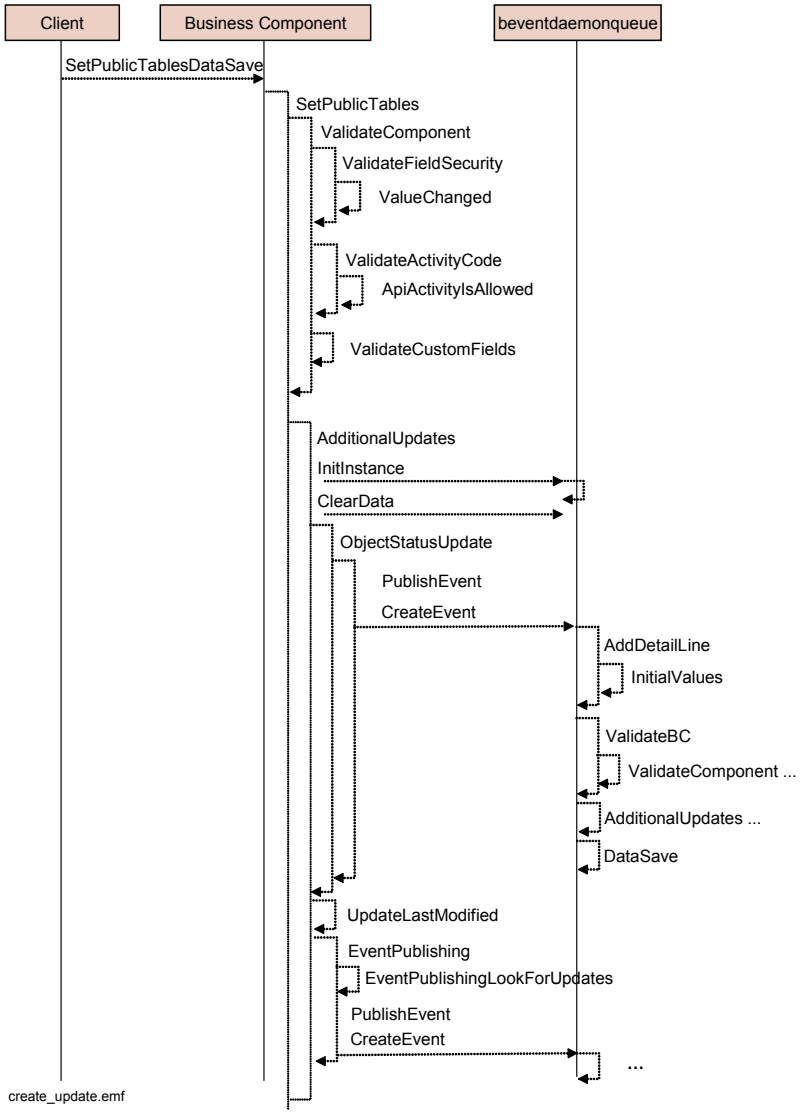
Fig. 10.4
Update Call Stack



Save Data in a Maintenance Form (Create and Update)

The following diagrams illustrate the call stack for saving data in a maintenance form.

Fig. 10.5
Save Call Stack



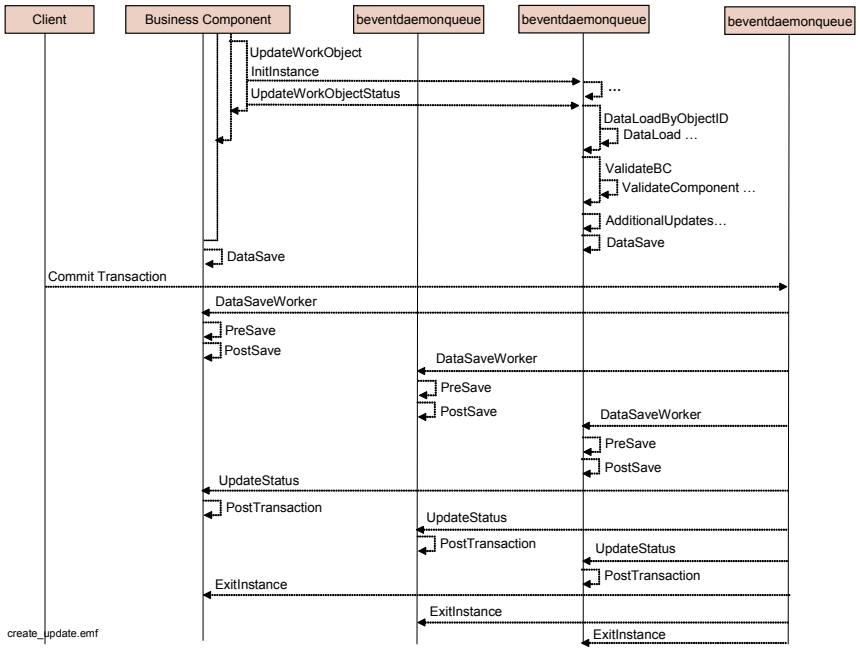


Fig. 10.6
Save Call Stack (contd.)

Deleting an Object in a Maintenance Form

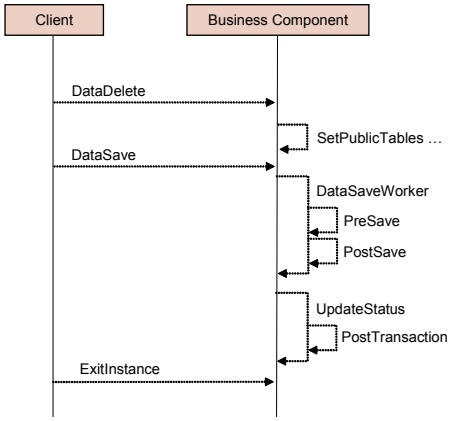
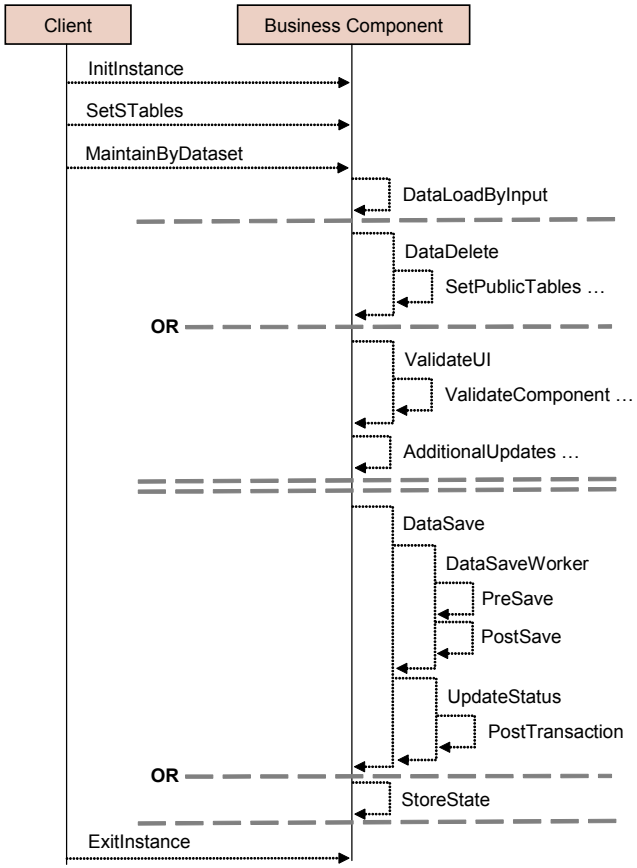


Fig. 10.7
Delete Call Stack

Importing Data using the XML Daemon

Fig. 10.8
Import Using XML Call Stack



Sample Code for Non-Intrusive Customization

The following code illustrates the `GetBusinessFields`, `InitialValues`, and `ValidateComponent` methods contained in the customization template file `BLanguage.p`.

```
/*
Procedure: GetBusinessFields
Description:
Return a list of public business fields of a business class.
Parameters:
```

```
input icReference
    (business class or (query) method name)
output dataset tBusinessFields
    (business fields)
output dataset tCustomRelation
    ()
output oiReturnStatus
    (Return status of the method.)
*/
/**
PROCEDURE BLanguage.GetBusinessFields.before:

END PROCEDURE.
**/
/**
PROCEDURE BLanguage.GetBusinessFields.after:

END PROCEDURE.
**/

/*
Procedure: InitialValues
Description:
Add code here to initialize the calculated fields of a 'new' record
(= a record that must be created in the application database) in a
class temp-table.

Parameters:
input icTableName
    (Name of the database table of which a record is created in the
    class temp-table.)
output oiReturnStatus
    ()
*/
/**
PROCEDURE BLanguage.InitialValues.before:

END PROCEDURE.
**/
/**
PROCEDURE BLanguage.InitialValues.after:

END PROCEDURE.
**/
/*
Procedure: ValidateComponent
Description:
Write here all tests on database update (new / modify / delete) that
cannot be coded with a validation mask.
The type of update can be found in tc_status (N/C/D).
If you find incorrect data, you must write an entry in tFcMessages
(using SetMessage) and set the return status of this method to either
+1 or -1.
Return status +1 = data will still be accepted.
Return status -1 = data will not be accepted.
This method is run from SetPublicTables, before transferring the
received data into the class temp-tables.
Parameters:
```

```
output oiReturnStatus
    ()
*/
/**
PROCEDURE BLanguage.ValidateComponent.before:

END PROCEDURE.
**/
/**
PROCEDURE BLanguage.ValidateComponent.after:

END PROCEDURE.
**/
```

Creating Customizations

Before and After events are provided for every method of the business component, and each event header describes the business method and its parameters. To customize the event, you uncomment the event you want to implement and add code into the internal procedure.

The Customization folder in the application root contains two sub-folders:

- Definition

  This folder contains an include file for each business component.

- Template

  This folder contains the `.p` program file for each business component. This program contains the Before and After event code for the component.

  Note You must include the Template folder in the PROPATH of each progress session used for compiling the customized component code. This ensures that the system can detect code version mis-matches during an application upgrade.

Writing Customizations

Use the following steps for every component that you want to customize.

1 Copy the file in the template folder to a local source code folder.

   You must always code in a locally stored copy of the template file, and not in the original file. You can store this copy anywhere on your local drive, but you must ensure that you copy only those template files that you are modifying. Ensure that the local folder you use is contained in the PROPATH for compilation.

   You also copy the include file from the definition folder to the local folder. This file is needed for compilation, and must not be modified.

2 Use a Progress 4GL editor to uncomment the events in the template file to be customized, and to add related code.

Compiling Customizations

Compile all program files in your local source code folder into a folder named `customcode`.

This `customcode` folder is placed in a folder that is part of the PROPATH of the appserver running the business logic.

Once the folder is included in the PROPATH, the application automatically runs the customization when the appserver is restarted or all agents are trimmed.

Updating Customizations

Whenever a new version of the application is installed, the customized programs in your local source code folder must be updated to the new version. In most cases, you can update the version number in the header of the program (`&scoped-define class-version xx.yy`) to the version number displayed in the new template folder. If the template file has undergone many changes since the previous version, it may be more useful to copy the template file and add your customizations again.

When you have reinstalled the application, you must compare the methods in your customizations with those contained in the new template files, because methods may change with every new version.

You must also evaluate the method descriptions and parameters, because parameters can also be removed between installations. You must resolve any differences before you continue with customization. Use a standard file diff tool to make your comparisons.

Running Other Business Methods

You can run any customizable procedure from within the customization, using a standard run statement.

The procedure itself can be customized or standard. The procedure is run within the standard business component through procedure overloading.

Example In this example, you run the method `GetCustomFieldList` from within a customization of `BItem.ValidateCustomFields`:

```
/**/
PROCEDURE BItem.ValidateCustomFields.before:
    define variable vcList as character no-undo.
    define variable viReturn as integer no-undo.
    run GetCustomFieldList (output vcList, output viReturn).
END PROCEDURE.
/**/
```

Examples of Customization

This section describes some sample customizations.

Example Customize the default value for `GLExchangeMethod` when creating a G/L account:

```
/**/

PROCEDURE BGL.InitialValues.after:


    /* tgl is a shared buffer on table t_ogl */

    if t_Parameter.icTableName = "gl"

    then assign tgl.GLExchangeMethod = "USERDEFINED".

END PROCEDURE.
/**/
```

Example Add default values for SAF codes when creating a G/L account:

```
/**/
```

```
PROCEDURE BGL.InitialValues.after:

    define variable viReturn as integer no-undo.

    if t_Parameter.icTableName = "gl"
    then do:
        /* AddDetailLine will create a record in tGLSafDefault */
        run AddDetaiLLine (input "GLSafDefault",
            input tgl.tc_Rowid,
            output viReturn).
        /* parent rowid */
        /* error handling */
        if viReturn < 0
        then assign t_Parameter.oiReturnStatus = viReturn.
        else assign tGLSafDefault.GL_ID = tgl.GL_ID
            tGLSafDefault.tcSafConceptCode = "<CustomValue>"
            tGLSafDefault.tcSafCode = "<CustomValue>".
    end.
END PROCEDURE.
/**/
```

Example Add a custom table to a report dataset:

```
/* define the custom table */
define temp-table glhistoryCustomData no-undo
    field MyCustomCode as character
    field MyCustomValue as character
    index ip is primary unique MyCustomCode.

/* create custom data */
PROCEDURE BGLReport.GLHistory.after:
    create glhistoryCustomData.
    assign glhistoryCustomData.MyCustomCode = "demo-code"
        glhistoryCustomData.MyCustomValue = "demo-value".
END PROCEDURE.
/**/
```

```
/* publish the custom data */

PROCEDURE BGLReport.ApiProcessReportLogic.after:

    if t_Parameter.icReportName = "glhistory"

    then t_Parameter.ozReportData:add-buffer(temp-table
    glhistoryCustomData:default-buffer-handle).

END PROCEDURE.

/**/
```

Index

M

N

O

P

U

V

W

X