Administration Guide

QAD Planning and Scheduling Workbenches

Introduction
Workbench Processes
System Load and Performance
Configure
Personalization Architecture

70-3174-3.2.4
QAD Enterprise Applications
August 2013

PlanSchedWrkBench_AG_v324.pdf/crl/crl

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

Contents

Change Summary

The following table summarizes significant differences between this document and the version released with previous Standard Edition releases.

Date/Version	Description	Reference
August 2013/3.2.4	Rebranded for version 3.2.4	--
	Added new information to User Exit section regarding the Calculate Capacity Quantity Completed (wosccqcx.p) user exit program.	page 38
May 2013/3.2.3.2	Rebranded for version 3.2.3.2	--
April 2013/3.2.3	Added new Modify Browse Filters section.	page 27
October 2012/3.2.2.1	Added new section, referencing a browse collection utility that you run during installation.	page 19
September 2011/2011 SE	Rebranded for QAD 2011 SE	--

Chapter 1

Introduction

This chapter introduces Planning and Scheduling Workbenches and its administration functions.

Overview 2
Introduces basic components of the QAD Planning and Scheduling Workbenches and introduces the data presented within this administration guide.

MSW and PSW License 3
Provides information on obtaining and registering the Maintenance license, needed to access MSW and PSW.

Conversion for Planning and Scheduling Workbenches 4
Provides information on conversion programs that automatically run when you update or retrofit versions.

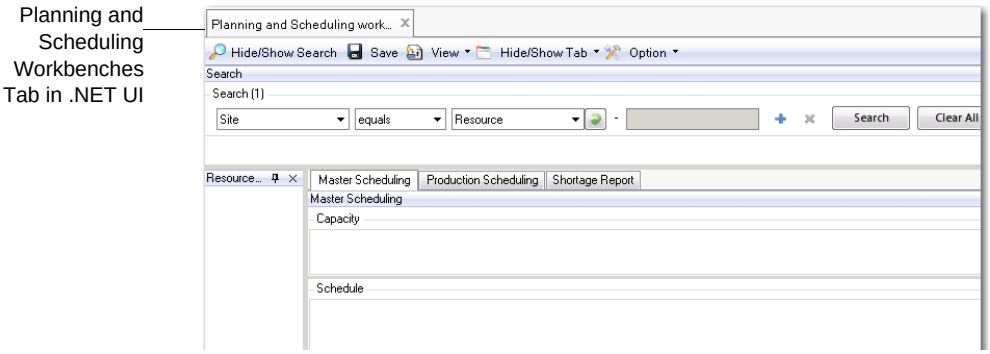
Important Because the Planning and Scheduling Workbenches are powerful tools that summarize all your demand, supply, and capacity information across your planning/scheduling horizon, you need to allocate one processor (core) per production planner/scheduler to prevent performance degradation.

Overview

The QAD Planning and Scheduling Workbenches are .NET UI-only scheduling tools that run in the QAD Enterprise Edition. The tool lets you effectively plan and schedule resources for a master schedule or a production schedule from a single workbench for each type of schedule.

You run the workbenches from the .NET UI's list of applications. The .NET UI tab indicates the Planning and Scheduling Workbenches; see Figure 1.1.

Fig. 1.1
QAD Planning and Scheduling Workbenches



Once you access the workbench, you can use the following functions and features:

- Master Scheduling Workbench (MSW)
- Production Scheduling Workbench (PSW)
- Integrated component check capabilities
- Several QAD SE programs and browses that provide supporting information
- The flexibility to modify and configure virtually every aspect of the UI to reflect the needs of each application user in your organization

This document focuses on topics of interest to system administrators who troubleshoot issues or problems, analyze behavior, interact with the components or processes, or customize the workbenches. Topics within this guide include discussions on:

- How the workbenches work and background processing
- System load and performance and troubleshooting information
- Data dimensions
- XML that builds the UI
- UI customization

The following topics briefly discuss each major component; for more information, see U*ser Guide: Planning and Scheduling Workbenches*.

Master Scheduling Workbench (MSW)

The MSW increases master scheduling efficiency by letting you simulate and commit scheduling changes, factoring demand, supply, inventory, production orders, and MRP data from several QAD SE programs in a single workbench.

Note Production orders are all orders associated with production—production lines, work centers, or other production areas—including discrete orders, repetitive orders, scheduled orders, cumulative orders, and so on.

You can use the MSW to interact with production line and work center schedules and make changes where necessary. Within MSW, you can update the production order status, as well as release, create, or close production orders, while considering all supply, demand, and capacity sources from the single workbench. You can also identify items with demand issues and check component availability for each production order to be released.

User-configurable parameters control the number of days that you are in control of the schedule as opposed to MRP control. You can also set the number of future and historical days to which you want visibility to your production data in the MSW.

Event-based color coding lets you easily identify areas of concern. You can review and manipulate schedule and production order data. Once satisfied, saving your schedule creates firm repetitive production schedules or revised and new production orders in QAD SE applications.

Production Scheduling Workbench (PSW)

Once you generate a master production schedule over a daily, weekly, or monthly horizon, you may need to create a production schedule for a shorter, two-to-five day period for the shop floor to drive shop floor execution. The production scheduling goal is to optimize shop floor efficiency by scheduling and sequencing production orders that have like attributes together, reducing machine setups and maximizing labor utilization.

Some companies run a single production order over several days, while others run multiple production orders within a single day. Further, some companies define a production sequence/priority by shift to monitor shift performance or to ensure that products are available for a specific shipment time. The PSW lets you schedule discrete and repetitive items on production lines. You can schedule, sequencing items within a day and shift.

MSW and PSW License

Like Operational Metrics, MSW and PSW require an active Maintenance license before you can access them. You obtain the Maintenance license key through the following Web site:

https://support.qad.com/license_keys/activemaintenance/

In the Web URL, click the Generate License Key button to generate a license key, then follow the prompts to generate the license key. Once you select the Accept key, the system generates the key, displaying it on the Web URL screen, and e-mails the license key to you.

Note The Support Web site requires your Support login information.

Once you obtain the license key, you must register the Maintenance license with the system through License Registration (36.16.10.1).

If you attempt to run either the MSW or PSW and you do not have an active Maintenance license, the system displays an error message. The system displays a warning if the active Maintenance license is close to the license expiration date. Refer to *User Guide: QAD Security and Controls* for information on registering the license key in QAD SE. Refer to the *QAD .NET User Interface 2.8.1 Release Notes* for information on Operational Metrics Licensing.

Conversion for Planning and Scheduling Workbenches

A conversion program for the workbenches automatically installs and runs during the standard QAD SE upgrade process. There is no user interaction required for the conversion program.

The program examines existing production line/item detail (lnd_det) records and sets the Run Crew Size field to 1 (one) if it is currently 0 (zero). When you enable the Run Crew Size field for the workbenches calculations, the system requires that the value of Run Crew Size be set to 1 to avoid issues that can arise when the system encounters a 0 in the field and attempts to divide by 0 in workbench calculations.

For information on other update programs that mass update existing data for use with the workbenches, refer to *User Guide: Planning and Scheduling Workbenches*.

Oracle and the Workbenches

If you use Oracle with the workbenches, note that QAD does not provide standard support for customers who use standard Oracle products. To ensure proper Oracle-workbenches processing and functionality, you should submit a request to QAD for an Oracle-compliant version.

Chapter 2

Workbench Processes

This chapter tells you how the system works and explains background processing.

Introduction

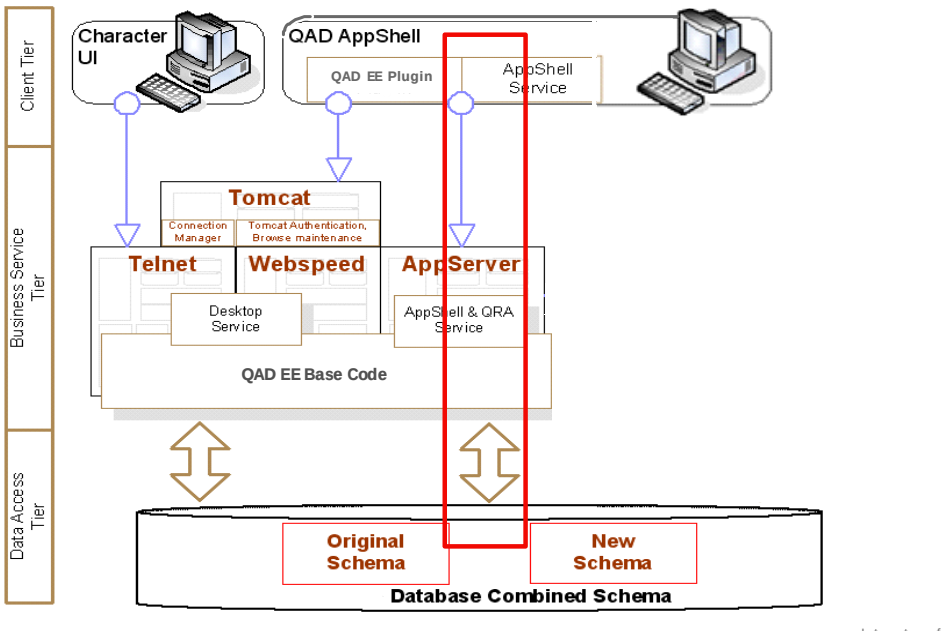
The Planning and Scheduling Workbenches, which includes two separate workbenches (MSW and PSW), a Shortage Report, several supporting browses and programs, and various frames that let you navigate resources, search, display capacity, and so on, is a plugin that runs inside the client; that is, it runs in the AppShell on the user's computer. The client connects with the database and business logic at the backend, or the *server*.

A *plugin* is a computer program that interacts with a host application (a Web browser or an e-mail client, for example) to provide a certain, usually very specific, function.

Schedulers create and maintain schedules in the workbenches, but they can also view how schedule changes impact the production line or work center load, supply and demand, and material availability. They also have simulation capabilities in the workbench. Typically, there is no impact on the database until the user commits by saving the data.

The following graphic depicts client-server interactions. The red line indicates the area of interest for MSW/PSW.

Fig. 2.1
Client-Server Interactions



Basic Processes

To understand what happens when users interact with the Planning and Scheduling Workbenches, the following topics present four common user processes:

- Search
- Modify
- Create new
- Save

These four processes typically constitute the basic processes performed in most applications. All of the processes, except modification, involve the client interacting with the server to access databases and run business logic.

Search Process

The following occurs when the user performs a search in the workbench:

1 The client sends search criteria to the program, running on the Progress appserver.

2 The program executes a search against the database.

3 The program transforms data into a format that the client needs.

4 The client receives data from the program and further transforms it.

5 The client loads data into memory, which, in turn, loads data on the screen.

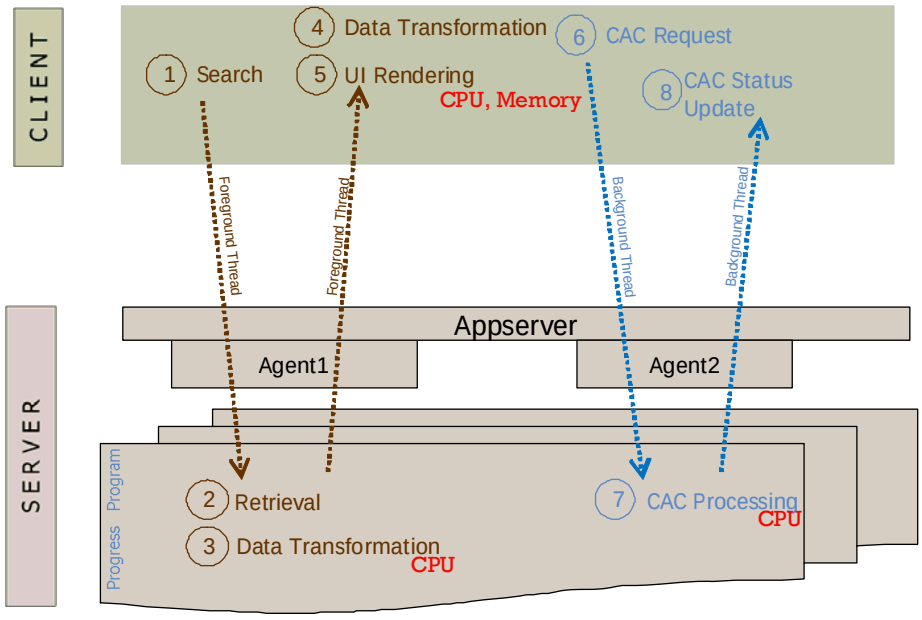
6 The client initiates a background thread to compute component availability check (CAC) statuses. The background thread calls the CAC computation program on the server.

Note A thread is a flow of control within a program, such as Java or C#, and is either a background or foreground thread. Background threads are similar to foreground threads except that they do not keep the managed execution environment running. So, when the last foreground thread stops, then all background threads stop, and the process stops.

7 The program computes statuses and returns them to the client.

8 The client loads statuses into memory, which, in turn, loads them on the screen.

Fig. 2.2
Search Process



Modify Process

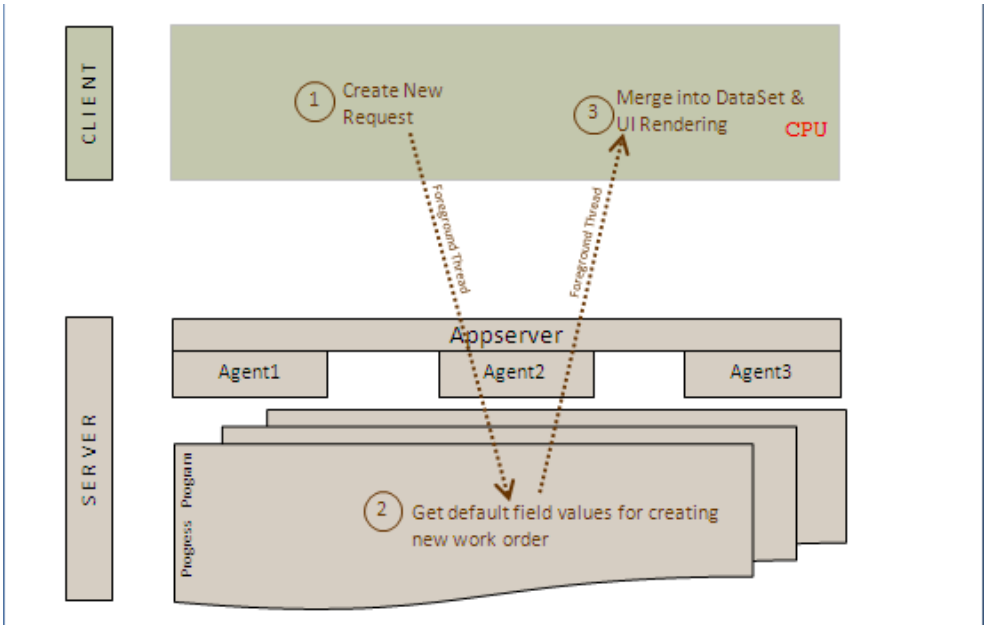
When the user modifies a production order, the client modifies data in memory. There is no client-server interaction, however.

Create New Process

When the user creates a new production order:

1. The client calls the program that runs on the Progress appserver.
2. The program makes a copy of the new production order, discards the production order created, but returns a copy.
3. The client loads the copy of the new production order into memory which, in turn, loads it on the screen.

Fig. 2.3
Create New Process



Save Process

When users save data in the workbenches, the following occurs:

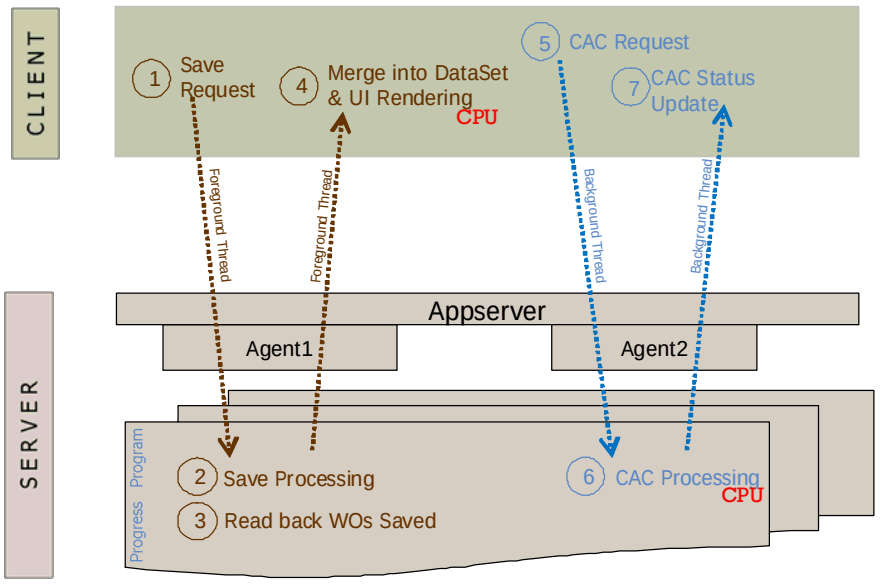
1. The client sends new/updated production orders to the program running on the Progress appserver.
2. The program saves the production orders after they have been validated.
3. The program reads back saved production orders.
4. The client reloads saved production orders into memory that it displays on the screen.
5. The client initiates a background thread to compute CAC statuses.

   The background thread calls the CAC computation program on the server.

6 The program computes statuses and returns the statuses to the client.

7 The client loads statuses into memory which, in turn, loads them on the screen.

Fig. 2.4
Save Process



As noted in the images, some steps demand significant CPU and/or memory resources. These steps largely decide the application response times. Chapter 4, “Configure,” discusses the parameters that affect system load. The key to getting better performance is to affect the system load and help these resource-intensive steps go faster.

Each request from the client results in the system running a Progress program on one of the appserver agents that is not busy. Until the processing for that request is complete, the agent is busy. Concurrent incoming requests are handled by one of the appserver agents that is available. In other words, concurrently running clients share a pool of appserver agents. It is important to ensure there are a sufficient number of agents in the pool to service all concurrent clients. Having insufficient agents results in incoming requests being queued, which adversely affects response times.

Steps that are resource intensive on the server are likely to keep the appserver agent busy longer. As the number of workbench users grows, it is important to monitor appserver agent usage and if necessary, to increase the agent pool size by configuring the appserver.

Chapter 3

System Load and Performance

This chapter describes system load and performance issues.

Introduction

The Planning and Scheduling Workbenches let schedulers simulate results. They can see the results of their updates and modifications before they commit this data to the database. The simulations on the schedule changes, however, can require that the system read and load a great deal of data from the database into system memory.

The system load comes from these elements:

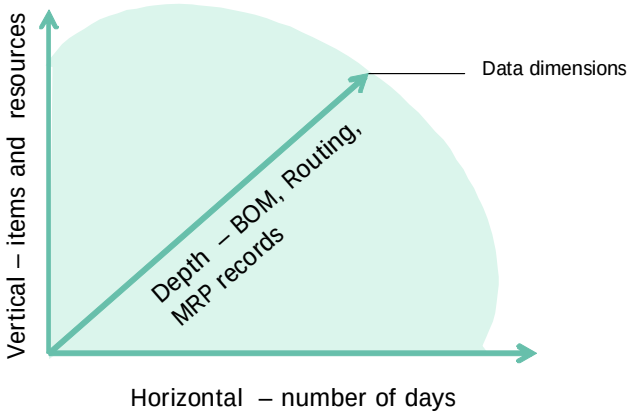
- Amount of data
- User preference parameters
- System configuration

System Load

System load comes in the following three dimensions:

- Item and resources
- Scheduling horizon days
- The number of MRP details; that is, production order records, component records (BOM), routing records, and so on

Fig. 3.1
System Load



Load on the client and server increases as any of dimensions increase, impacting system response time and, hence, performance.

Search criteria and user preference settings impact the three dimensions, too, as shown in the following table, which presents the performance impact by order of impact.

Table 3.1
Performance Impact by Order of Impact

Impact (1=Highest)	Area	Impact
1	Search criteria	This is the most significant impact, affecting the number of item/resources and therefore, indirectly the number of production orders, components, and routings.
2	Scheduling horizon	The horizon affects the number of production orders, components, and routings. After search criteria, this is the most significant parameter that affects performance.
3	Other user preferences	Setting other user preferences can impact performance; see “User Preferences Impact” on page 13.

User Preferences Impact

Use the data in Table 3.2 to determine whether a user preference can be disabled to improve performance. For example, if you disable CAC, you can improve performance when retrieving and saving data.

Table 3.2
User Preference Impact

Search/User Preference	Impact
Enable CAC	When enabled, the system computes component availability statuses during both search and save operations. This is a processor-intensive computation that can add a significant load on the server. ***Important*** If you do not need CAC functions, you should disable this feature as it can make a significant difference in performance.
Process Operation Details	Disabling this function prevents the search from transferring routing records to the client. Not having routing records on the client removes the required additional processing the system needs to keep routings updated. Users who schedule for production lines only should use this function.
Display Search Progress	When searching, this option makes the client divide the entire search into smaller groups. It also causes a progress bar to display as it processes each group. A processing overhead is associated with dividing the search into smaller groups. If the search takes too long, you should try disabling this feature.
CAC Horizon	The CAC horizon determines the number of days for which the system computes CAC status. This affects the number of MRP detail records that the server processes and sends to the client.

System Configuration

System performance is affected by system configuration, both hardware and software, as follows:

- Client PC

Processor speed and amount of memory have a very significant impact on response times. Chapter 2, “Workbench Processes,” discusses the common processes that demand significant memory and processor resources. Internal performance tests reveal that using PCs with older processors can result in less-than-optimal performance. So, it is important to have a client PC with a processor that has at least 3GB of RAM.

- Server machine

  Chapter 2, “Workbench Processes,” discusses a number of common processes that demand significant processor resources on the server machine. For this reason, it is important that the server machine has enough spare processing capacity to handle the load.

- Network

  All processes involving client-server interaction are affected by network speed; however, during data retrieval, the system transfers significant amounts of data from the server machine to the client over the network. System administrators should look for poor response times caused by network latency.

- Software configuration

  Progress administrators should ensure that there are a sufficient number of agents to service all concurrent QAD AppShell users; otherwise, the system can queue appserver requests from the client, resulting in poor response times.

Troubleshooting Performance

Although there is no single tool that lets you measure the application’s response times, system throughput, and system stress levels, you can improve performance by increasing processors (cores) and curtailing the use of some user preferences to improve performance. You can view log files to aid in troubleshooting performance.

Increasing Processors

The Planning and Scheduling Workbenches provide schedulers/planners visibility into future supply and demand, projected shortages, projected capacity overloads/underloads, and projected component shortages—allowing schedulers/planners to make faster, more informed, and more efficient decisions.

To provide the data, the workbenches process and summarize large amounts of transactional data. The processing required depends on the volume of transactional data for the query parameters that schedulers/planners specify. When the processing involved is large, it can add a significant load on the server machine during the processing period. Users of other applications that run on that server machine may experience a degradation in response time during those periods.

Since every environment is different, the effect of the additional load on server machine users is unpredictable. The effect depends upon the:

- Server machine’s spare processing capacity
- Scope of query
- Database transactional data volume

To address the slower response time other users may experience during workbench processing, QAD recommends augmenting the server's processing capacity by adding an additional processor (core) for each concurrent scheduler.

User Preference Settings

The query-selection process, which lets you specify the sites, resources, and resource types, combined with the number of history and future days you specify, results in searches that can take from seconds to minutes.

It is important that all workbenches users properly define the query selection and correctly set the historical and future days of data to retrieve. Performing a search for all resources across sites or all resources within a site is strongly discouraged. Users should try to use Scheduler ID as a minimum, not just site ID, to improve search retrieval time.

There are also user preference settings defined that let the system perform data retrievals in a single request as opposed to an incremental request method; the latter method could require 20 to 30 percent more processing load over the single request method. Material on performing searches is available in segments on Getting Familiar with MSW and PSW within the Planning and Scheduling Training Guide and videos. For information on setting user preferences, refer to *User Guide: Planning and Scheduling Workbenches*.

Log Files

You can examine the system log files to determine what took place in the system and how long it took. When users make requests to the back-end processor, or server, the system tracks the request in the appserver log file as well as in the client log file.

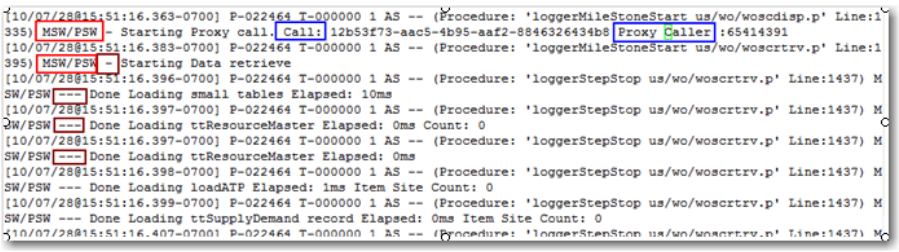
The client log file is on the user PC and only contains log messages pertaining to that user's activity. The appserver log file, on the other hand, is shared by all users. So it contains log messages pertaining to activity of all users.

Note For additional information on reading log files, refer to the following link:

http://tracker.qad.com/confluence/display/QDN/Planning+and+Scheduling+Workbenches+Support

To effectively troubleshoot an issue, you should be able to associate a client request with a server response. Since the server log is for all clients, associating client requests with server response can be very difficult. To facilitate this association, both log files show Call ID and Proxy Caller ID. In addition, all messages are prefixed with the words MSW/PSW.

Fig. 3.2.
Appserver Log File, Proxy Caller



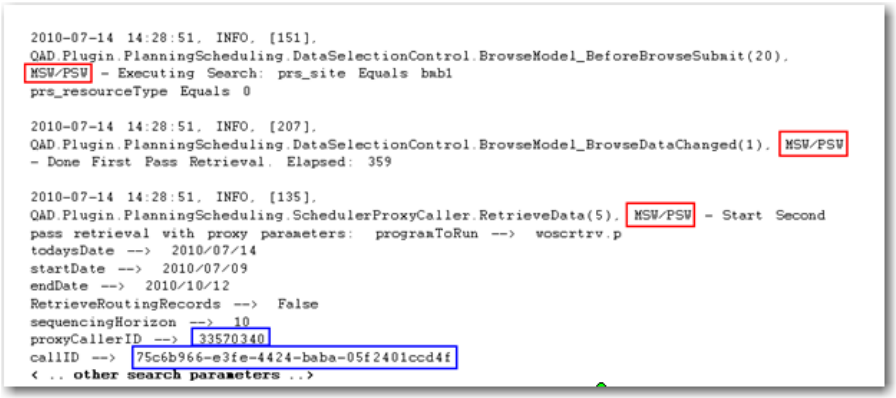
In the log file, you can determine when a user starts data retrieval as the log file depicts the start and finish of the data retrieval in milliseconds. The information can help you determine exactly how many records were retrieved, the type of record, how long the retrieval took, which user retrieved the records, and so on.

Knowing this can help you curtail long user retrievals. When retrieval issues arise, you can help users resolve the issues by successfully employing the MSW/PSW search filters or disabling functions through the user preferences.

You can find the client log file in .NET UI. At the top menu, select Help, then About, then View Log. If you operate on the appserver side, the system displays the appserver log file; if you operate on the client side, the system displays the client log file.

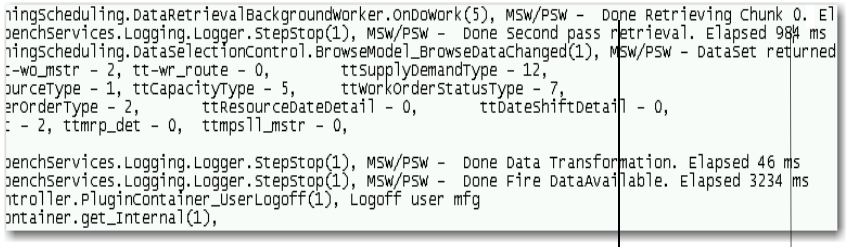
Each MSW or PSW request is preceded by the word MSW or PSW; see figure Figure 3.3.

Fig. 3.3.
Client Log File



The following graphic shows that the first pass of the data retrieval on the server took 259 milliseconds.

Fig. 3.4.
Appserver Log File, Timing



If you work on the client, the system retains a client log file that uses the same format and keeps track of the same events and requests that the appserver log file tracks.

Additional Files

The `/Application Data` folder of the client machine contains files for each plugin. Inside the `/plugin` folder is a layout folder that contains a list of XML files. The files are the views that Windows saves for the user as set in the View option. Note that the system saves views for Windows, not QAD SE users. That is, the system provides a capability that allows the layout to be saved to files; so, the XML files depend upon the view you have saved.

You can clear out all Application Data files for an AppShell installation; however, doing so would affect all data related to all plugins. So, if you choose to clear Application Data files, it clears all of previously mentioned files for MSW/PSW.

In addition, in the Application Data plugin folder, the following files exist:

- `Browsesearchconditions.xml`
- `Preferences.xml`

`Browsesearchconditions.xml` saves the last search that was performed by that Windows user. `Preferences.xml` contains the preferences saved by that Windows user.

Chapter 4

Configure

This chapter discusses the following configuration information:

Overview

This section describes how the plugin is internally configured. It describes the layout and location of the configuration XML file. It also provides an example and procedures to add a new browse to the workbench.

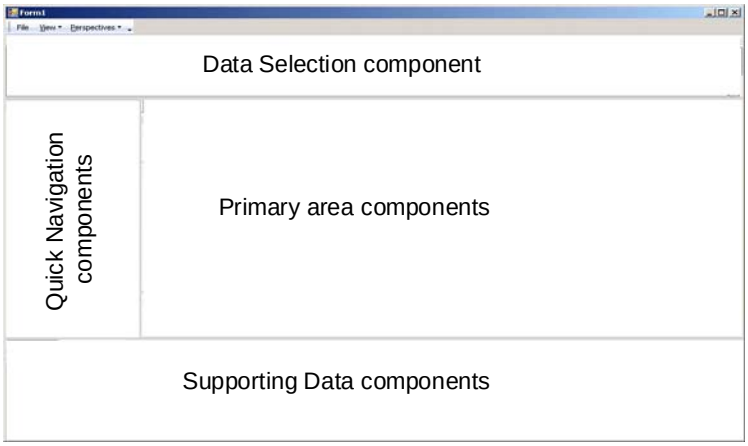
To understand how you configure the workbench, you must first understand the general layout of the workbench. The workbench user interface is divided into four areas:

1. Data selection: This area includes the basic search fields and expanded search fields that let you filter records the system retrieves in the workbenches. This area lets users select records from a very broad range—for example, all records associated with a site—to a very specific set of records—for example, a small set of orders for a particular production line. Users can optionally set user preferences so that the amount of time for the retrieval displays.
2. Primary data: This area is the heart of the MSW and PSW workbenches as well as the shortage report and CAC features. It is the area in which most users spend their time. Users select fields, rows, columns, or multiple fields, rows, and columns to manipulate data. They can also drag and drop data in this area within the MSW. Selections made in this area can change the data that displays in the supporting data area.
3. Quick navigation: This area is similar to the quick navigation that displays in many Internet browsers that helps you quickly navigate through an abundance of data.
4. Supporting data: This area includes browses and other programs that display data that supports the primary data. Programs that are new for the Planning and Scheduling Workbenches as well as browses are available through a row of tabs.

For the MSW/PSW plugin, the components that make up the four areas are as follows:

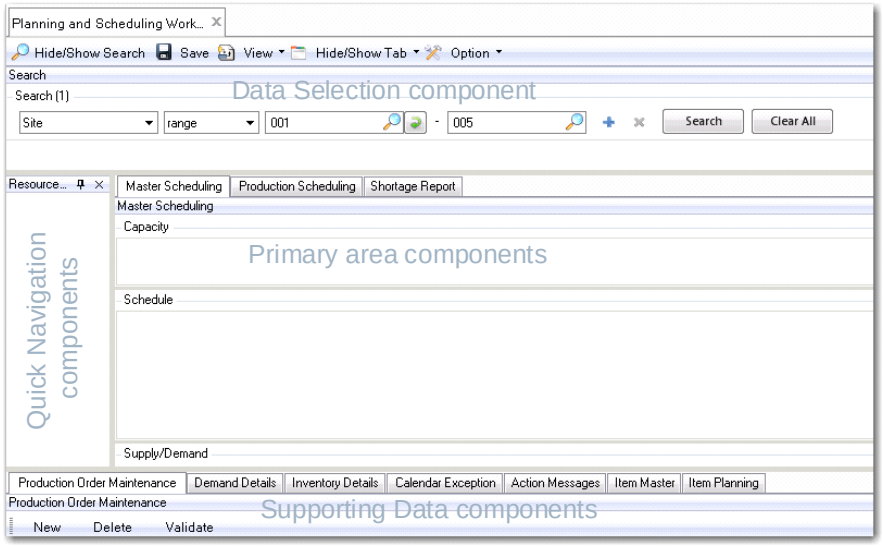
1. Browse component, `qpbr001.p`
2. Resource Navigator component
3. MSW, PSW and Shortage Report components
4. Browses Demand Details, Inventory Details, Calendar Exception, Action Messages, and Item Master data and maintenance programs, Production Order Maintenance, and Calendar Exception Maintenance

Fig. 4.1
Workbench Layout



The following graphic depicts the components on the actual Planning and Scheduling Workbenches UI to help relate the components to the interface.

Fig. 4.2
UI with Components



ControlConfig.XML File

The `ControlConfig.XML` file sets up and defines the QAD Planning and Workbench configuration. It defines the major frames that display and the actual fields that display within the supporting data frames; that is, the fields that display in the programs and browses that display in the tabs at the bottom of the workbench.

The XML consists of four tags, one for each area that the user interface has as shown in the blue boxes. Each tag can contain one or more control tags. Each control tag is for a component that displays in the user interface area. Each control tag consists of a class, assembly, and name tags. In addition, there may be other tags that are unique to a certain component. The visible tag is currently not used; see Figure 4.3.

Fig. 4.3
`ControlConfig.XML` Code

The file is located in the /`config` directory of your plugin installation on the local PC. In a typical system, the /`config` directory is located in a path like the following:

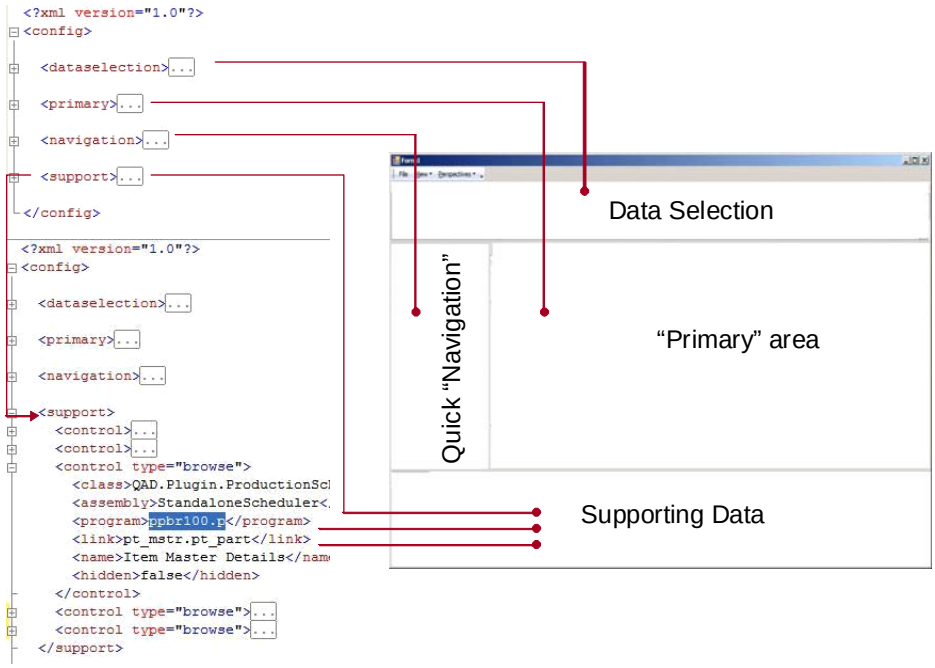
```
C:\Program Files\QAD\QAD Enterprise
Applications...\plugins\QAD.Plugin.PlanningScheduling\config\...
```

The file consists of control elements that define components of the workbench configuration in a hierarchical manner. For example, in the primary area are three controls are defined. These display on the user interface as three tabs:

- MSW
- PSW
- Shortage Report

Figure 4.4 depicts the code in the XML file and how it relates to the workbench framework.

Fig. 4.4
Code-UI Relationship



The Resource Navigator is defined as a control element for the `<navigation>` element, and Production Order Maintenance is defined as a `<control>` element for the `<support>` element. The actual programs are defined within the class `<class:>` definitions; see Figure 4.5.

Fig. 4.5
`ControlConfig.XML` File

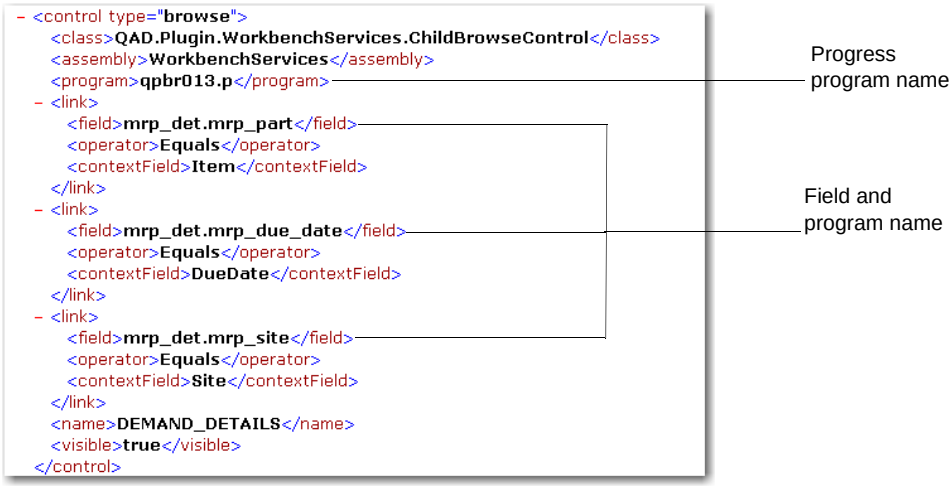
```
<?xml version="1.0" ?>
<config>
- <dataselection>
  - <control>
      <class>QAD.Plugin.PlanningScheduling.DataSelectionControl</class>
      <assembly>PlanningSchedulingPlugin</assembly>
      <visible>true</visible>
      <name>SEARCH</name>
    </control>
  </dataselection>
- <primary>
  - <control>
      <class>QAD.Plugin.PlanningScheduling.MasterSchedulerControl</class>
      <assembly>MasterScheduler</assembly>
      <visible>true</visible>
      <name>MASTER_SCHEDULING</name>
    </control>
  - <control>
      <class>QAD.Plugin.PlanningScheduling.ProductionSchedulerControl</class>
      <assembly>ProductionScheduler</assembly>
      <hidden>false</hidden>
      <name>PRODUCTION_SCHEDULING</name>
    </control>
  - <control>
      <class>QAD.WorkOrder.ShortageMonitorControl</class>
      <assembly>WorkOrder</assembly>
```

Defines the three major tabs that display in the workbench.

The following graphic depicts how the configuration, defined in `ControlConfig.XML`, relates to the UI.

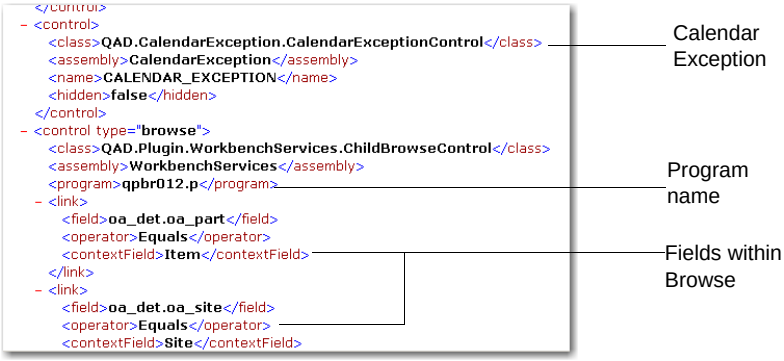
Browses that display within Production Order Maintenance are defined in `ControlConfig.XML` as `<control type=browse>`. The file defines the QAD SE Progress program name `(.p)`, then the fields within the browse are defined next, which includes the table and field name as defined within QAD SE.

Fig. 4.6
Browse and Field Definitions



Other programs in tabs that display at the bottom of the workbench are also defined as control elements. This is followed by defining the control type as browse, then field definitions that display within the browses. The following figure depicts the Calendar Exception tab.

Fig. 4.7
Other Program Definitions



Adding Browses to the Workbench

You can configure the workbench to add browses. Adding new components other than browses to the XML requires that you have a knowledge of C# and workbench framework internals.

To configure, you start by accessing and editing the `ControlConfig.XML` file within the `/plugins` directory. In most cases, companies have their own data with which they work when creating master or production schedules, so the most common configuration change is the addition of a customer browse that pulls fields that hold data of interest to your schedulers.

To add a new browse that displays as a tab in the Supporting Data area alongside other QAD SE browses and programs, use the following example.

Example You have a QAD browse, *`YourBrowsebr001.p`*, customized for your business needs. You need the fields in the browse to display in a browse in the Planning and Scheduling Workbenches.

To easiest way to do this is to copy an existing browse configuration in the `ControlConfig.XML` file; then edit the copied text for your new browse entry.

To use this method, use the following procedure.

1 In `ControlConfig.XML`, locate and copy a browse control type definition:

```
- <control type="browse">
    <class>QAD.Plugin.WorkbenchServices.ChildBrowseControl</class>
    <assembly>WorkbenchServices</assembly>
    <program>qpbr011.p</program>
-   <link>
       <field>pt_mstr.pt_part</field>
       <operator>Equals</operator>
       <contextField>Item</contextField>
   </link>
-   <link>
       <field>si_mstr.si_site</field>
       <operator>Equals</operator>
       <contextField>Site</contextField>
   </link>
  <name>ITEM_PLANNING</name>
  <visible>true</visible>
  </control>
```

2 In the copied code, set the `<program>qpbr011.p</program>` entry to reflect the name of your new browse program, for example: `<program>`*`YourBrowsebr001.p`*`</program>`.

3 Determine which fields should be linked to the workbench, using QAD SE Browse Maintenance (36.20.13) to find valid field names for the fields that display in your browse; see Figure 4.8.

For example, if you are only interested in linking the item field with the workbench, you use Browse Maintenance for *`YourBrowsebr001.p`* to determine that the name of the item field is `xx_mstr.xx_part`.

4 Locate the `<link>` entry of the copied text, then enter the field name as depicted in Browse Maintenance within the `<field> </field>` parameters for the field to display in the browse.

Using the example in Step 3, enter:

```
<field>xx_mstr.xx_part</field>
```

5 Enter the operator as:

```
<operator>Equals</operator>
```

6 Enter the context for the field.

Using the example in Step 3, enter:

`<contextField>Item</contextField>`

For the Planning and Scheduling Workbenches, you can only enter a certain set of values for `<contextField>`. You cannot increase the set of values without QAD assistance. The set includes the following `<contextField>` values:

- Item
- Site
- DueDate
- Resource
- Resource2
- ResourceType

Where:

Resource is the production line ID or work center ID.

Resource2 is used for the machine.

ResourceType is zero (0) for the production line and one (1) for the work center or machine.

7 Continue editing existing `<link>` entries or add new `<link>` entries.

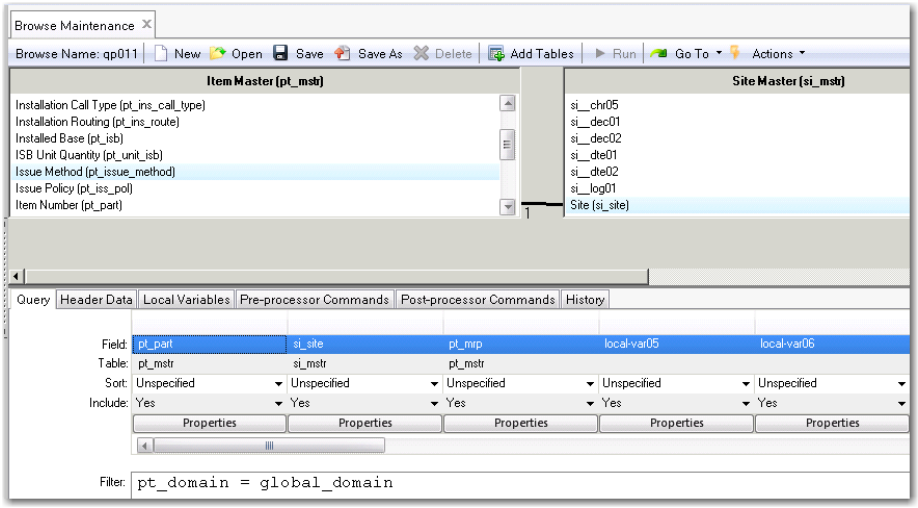
8 Name your browse and enter the name of the browse as:

`<name>YourBrowse</name>`

9 Save your entries.

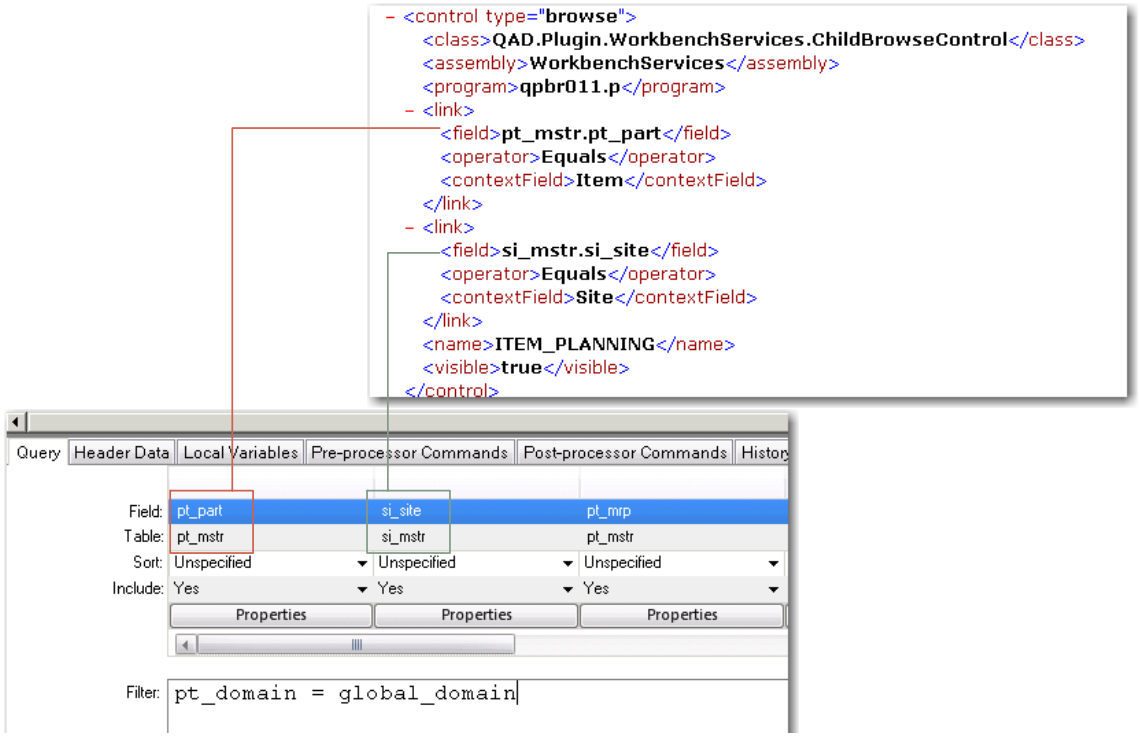
Important You should configure the XML for each client machine where the Planning and Scheduling Workbenches are configured.

Fig. 4.8
Browse Maintenance (36.20.13)



The following graphic depicts the code for a browse definition within `ControlConfig.XML`. It depicts how the fields names that you enter into the XML file were selected from Browse Maintenance.

Fig. 4.9
XML Fields



Modify Browse Filters

By modifying the `ConfigControl.xml` file, you can modify the filters that the supporting browses use when they display records. You can modify to filter by dates, or by other specific values, such as an order status, or a specific numeric value that has meaning in your company within item numbers. For example, you can view all transactions for the last two years, only records with dates of today plus all future dates, or specify a specific date upon which to view records.

To modify filters for the browses, you set two variables in `ConfigControl.xml` for the browses:

`<relativeConstant>`

True: Set to true when you set `<Constantvalue>` to a relative value, such as a relative date value of TODAY, YESTERDAY, FIRST PERIOD, and so on. See Table 4.1for a list of valid relative date values to enter.

False: Set to false when you set `<Constantvalue>` to an absolute date, such `2013-01-01`, `C`, or `2010.`

`<ConstantValue>`

Set this to a relative value, such as TODAY, YESTERDAY, FIRST PERIOD, or to an absolute value, such as `2013-01-01` for a date, or `15` for a item number.

Note For a constant date value, the format is YYYY-MM-DD. So for April 15, 2013, the value would be 2013-04-15.

For example, when you set <relativeConstant> to true and <ConstantValue> to TODAY, the browse displays all records set to today's date. Or, when you set <relativeConstant> to true and <ConstantValue> to YESTERDAY, the browse displays all records relative to yesterday's date in the browse.

Table 4.1 provides a list of date-related values in the correct format to enter in the ConfigControl.xml file when filtering by dates.

Table 4.1
Date Value Formats

Today	XML Code to Enter
Yesterday	YESTERDAY
Tomorrow	TOMORROW
Current year	CURRENT_YEAR
Current Quarter	CURRENT_QUARTER
Current Month	CURRENT_MONTH
Current Week	CURRENT_WEEK
Last Week	LAST_WEEK
Last Year	LAST_YEAR
Next Year	NEXT_YEAR
Two Years Ago	TWO_YEARS_AGO
Last Two Years	LAST_TWO_YEARS
Last Quarter	LAST_QUARTER
Next Quarter	NEXT_QUARTER
Last Month	LAST_MONTH
Next Month	NEXT_MONTH
Last Week	LAST_WEEK
Next Week	NEXT_WEEK
Last 7 Days	LAST_7_DAYS
	Note: You can change 7 to 30, 60, 90, or 120.
Next 7 Days	NEXT_7_DAYS
	Note: You can change 7 to 30, 60, 90, or 120.

Additional Examples

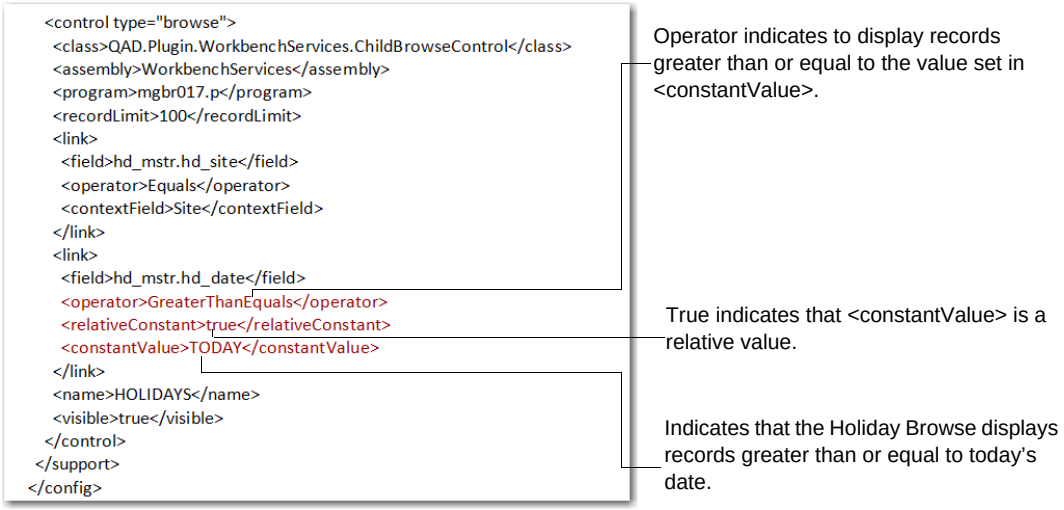
The following includes additional examples that you can set:

To see whether today or future days are holidays in the Holiday Browse, set:

```
<relativeConstant>true</relativeConstant>
<constantValue>TODAY</constantValue>
```

The following depicts the code for the Holiday Browse filter example within ControlConfig.XML.

Fig. 4.10
Code to Filter Holiday Browse



As another Holiday Browse filter example, to see New Year's Day on 2011 in the Holiday Browse, set:

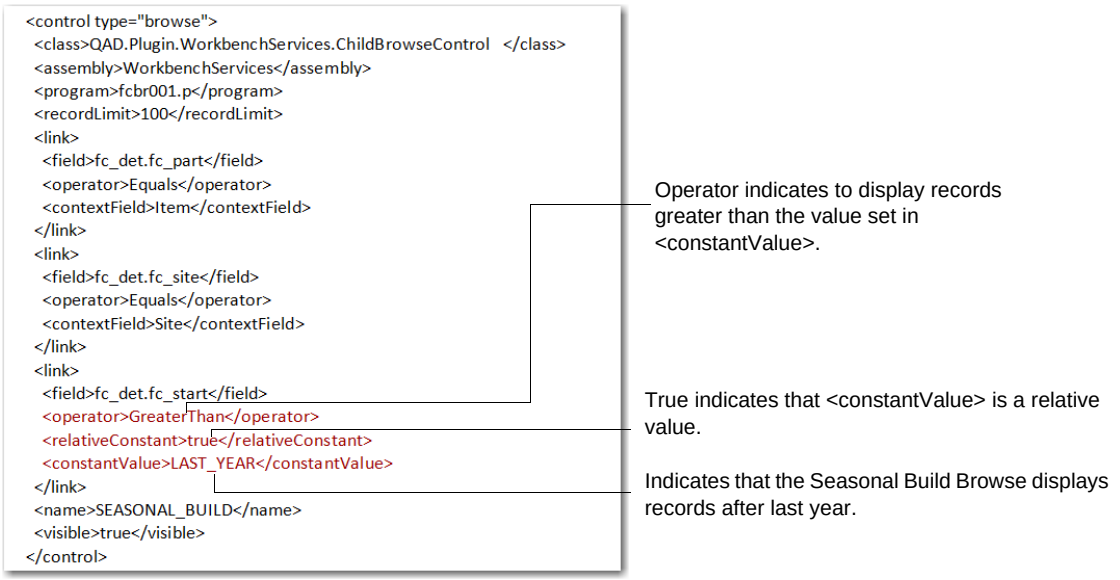
```
<relativeConstant>false</relativeConstant>
<constantValue>2011-01-01</constantValue>
```

In the Seasonal Build Browse, to see this year's records, set:

```
<relativeConstant>true</relativeConstant>
<constantValue>LAST_YEAR</constantValue>
```

The following depicts the code for the Seasonal Build Browse filter example within `ControlConfig.XML`.

Fig. 4.11
Code to Filter Seasonal Build Browse Records

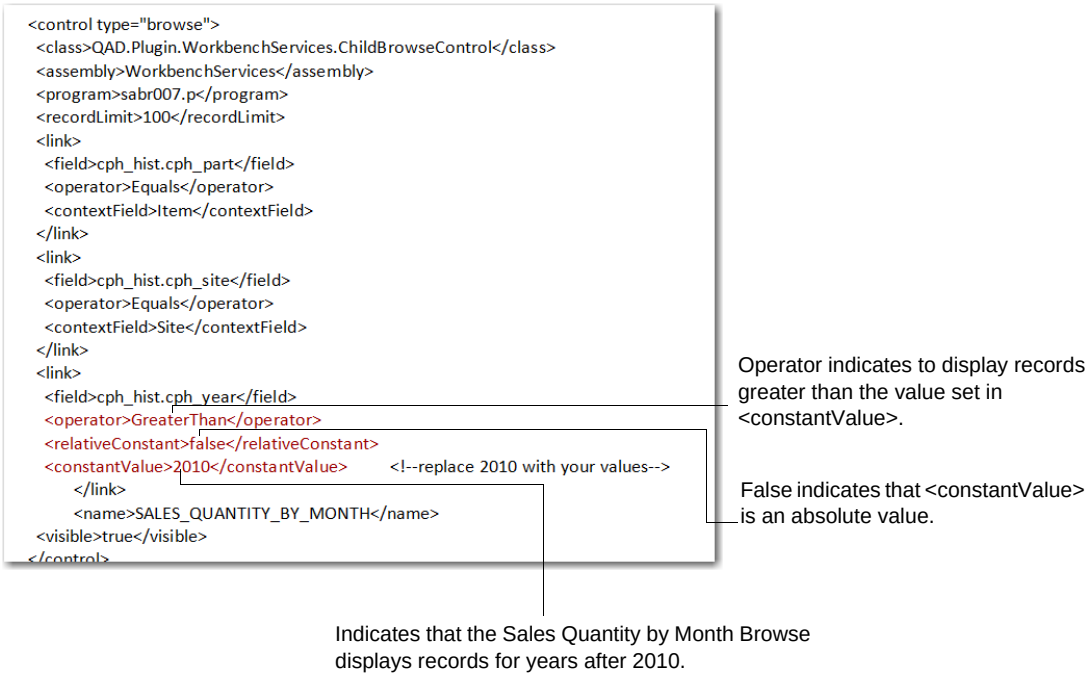


In the Sales Quantity By Month Browse, to see only records after the year 2010, set:

```
<relativeConstant>false</relativeConstant>
<constantValue>2010</constantValue>
```

The following depicts the code for the Sales Quantity by Month filter example within `ControlConfig.XML`.

Fig. 4.12
Code to Filter Sales Quantity By Month

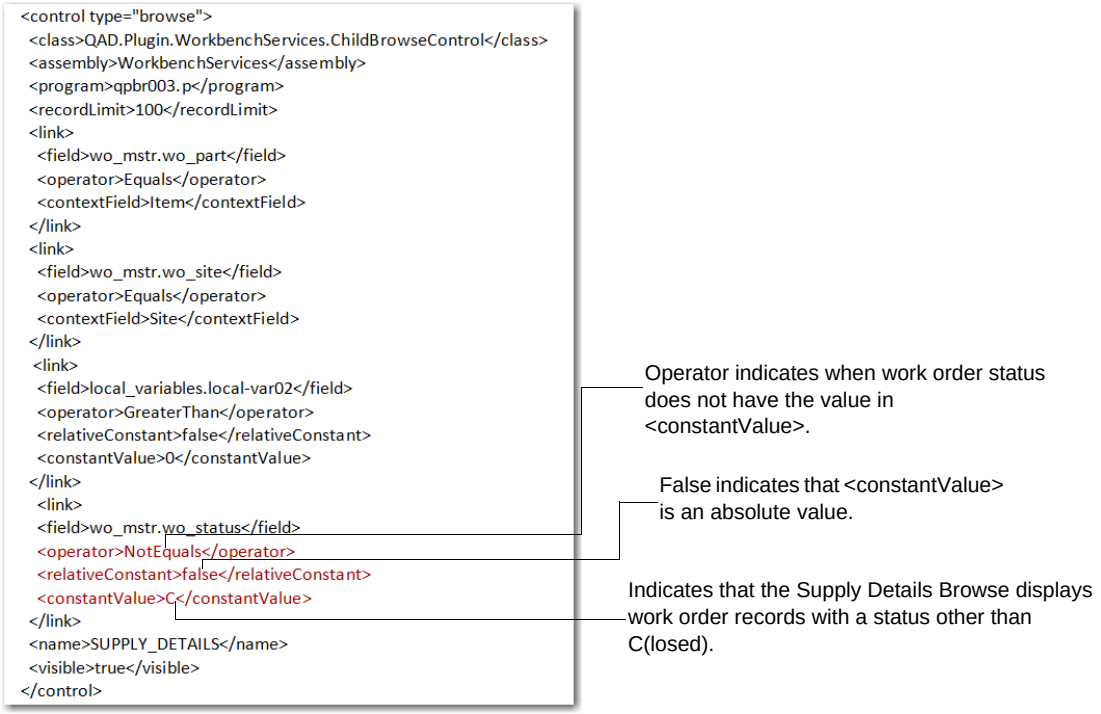


To see work orders with a status other than C(losed), set:

```
<relativeConstant>false</relativeConstant>
<constantValue>C</constantValue>
```

The following depicts the code for this example within `ControlConfig.XML`.

Fig. 4.13
Code to Filter Work Order Records By Status



Procedure

To do this, you start by accessing and editing the `ControlConfig.XML` file within the `/plugins` directory; then, use the following procedure:

1. In `ControlConfig.XML`, locate the code for the browses; see “Adding Browses to the Workbench” on page 24.
2. Locate the `<Link>` element for the filter you want to change. The following example depicts the `<link>` element for the work order status:

```
<link>
   <field>wo_mstr.wo_status</field>
   <operator>NotEquals</operator>
   <relativeConstant>false</relativeConstant>
   <constantValue>C</constantValue>
 </link>
```

3. Modify the `<relativeConstant>` value to true for a relative `<constantValue>` or false for an absolute `<constantValue>`.
4. Modify the `<constantValue>` by changing the value to either an absolute value or a relative value.
5. Save your changes; see “Saving Your Changes for New Releases” on page 33.

Note You can also copy `<link>` elements by copying all content between `<link>` and `</link>`, then pasting the copied content to another browse section within the file. Or, you can make `<link>` filter elements comments by inserting the link between the `<!-->` and `-->` markers.

Modify the Number of Browse Records that Display

By modifying the `ConfigControl.xml` file, you can change the default number of records that display in the supporting browses in the workbenches. To change the default, locate the following marker and enter the desired value:

```
<recordLimit>100</recordLimit>
```

For example, to change the number of records from the default of 100 to 25 that display for the following supporting browse, locate `<recordLimit>`, then change the value to 25.

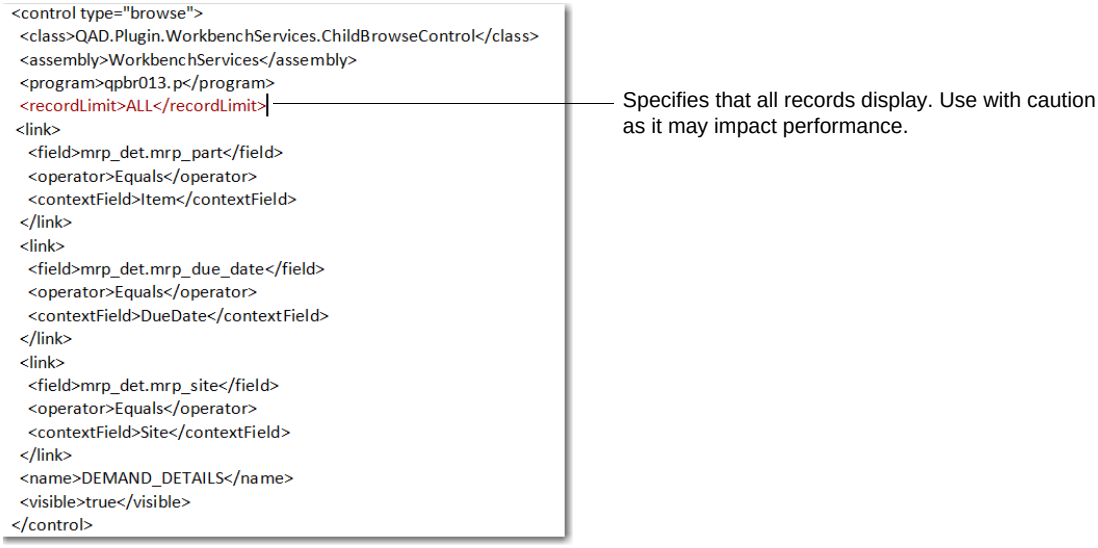
Fig. 4.14
Code to Change Browse Record Limit

```
<control type="browse">
 <class>QAD.Plugin.WorkbenchServices.ChildBrowseControl</class>
 <assembly>WorkbenchServices</assembly>
 <program>qpbr003.p</program>
 <recordLimit>100</recordLimit>
 <link>
  <field>wo_mstr.wo_part</field>
  <operator>Equals</operator>
  <contextField>Item</contextField>
 </link>
 <link>
  <field>wo_mstr.wo_site</field>
  <operator>Equals</operator>
  <contextField>Site</contextField>
 </link>
 <link>
  <field>local_variables.local-var02</field>
  <operator>GreaterThan</operator>
  <relativeConstant>false</relativeConstant>
  <constantValue>0</constantValue>
 </link>
 <link>
  <field>wo_mstr.wo_status</field>
  <operator>NotEquals</operator>
  <relativeConstant>false</relativeConstant>
  <constantValue>C</constantValue>
 </link>
 <name>SUPPLY_DETAILS</name>
 <visible>true</visible>
</control>
```

Change the default of 100 records to display to the desired value.

Important You can change the record limit to display all records by inserting the word ALL within the `<recordLimit>` and `</recordLimit>` markers as shown in Figure 4.15; however, when you do, it can potentially cause an impact in performance.

Fig. 4.15
Displaying All Records



Saving Your Changes for New Releases

Should you update the MSW/PSW with additional releases, you must save your `ControlConfig.XML` file, then move it to another directory to hold while you install new files from the installation disk. Once the files install for the newer version of MSW/PSW, you can then move or copy your custom `ControlConfig.XML` file, overwriting the version of the file you installed for the update.

Customizations When Updating to a New Release

When you update to a later release of the MSW/PSW, you must delete the MSW/PSW and CAC browse definitions before loading data files. The installation process presents a utility that lets you select browse definition data files to delete. When you customize any browses, you must reapply the customizations after the installation procedure is complete. See *Installation Guide: QAD Planning and Scheduling Workbenches*, item 78-0953-3.2.3.

Chapter 5

Personalization Architecture

This chapter discusses the following configuration information:

Overview

This chapter describes how to use the MSW/PSW Personalization Architecture. Personalization architecture provides an easy way to add columns to the Work Order Master (wo_mstr) and Work Order Component Detail (wod_det) grids that display in workbenches-embedded Production Order Maintenance, PSW, and Shortage Report sections. In addition, the architecture provides a way to add associated processing logic.

Adding Additional Columns

Each Work Order Master (wo_mstr) and Work Order Component Detail (wod_det grid) has a default list of columns to display. Using Personalization Architecture, you can add additional columns to this list. You can add the following columns:

- Existing columns in the wo_mstr or wod_det database tables. The columns that you add can be either display-only or editable. The edited data can be stored in the database.
- Calculated columns. These are not defined in the database tables, but instead are defined in temporary tables (temp-tables) The columns that you add can be either display-only or editable. The edited data can be stored in the database.

Refer to the “Customization Scenarios” for examples on how to add specific aspects of columns.

Adding Additional Processing Logic

Depending on the type of columns that you add, associated processing logic may be needed. For example, you may need to add processing logic to:

- Save edited data into the wo_mstr, wod_det or other database table(s)
- Calculate the values for a calculated column
- Perform validations on edited data

The architecture provides user exits that you use to implement additional processing logic; see “User Exits”.

ProDataSets and Temp-Tables

The MSW/PSW uses ProDataSets to send/receive data between the MSW/PSW .NET UI client and the server-side .NET UI. Basically, a ProDataSet is an in-memory data structure that consists of a number of temp-tables. Temp-tables are like database tables except that they are in-memory data structures. There are temp-tables defined for wo_mstr and wod_det.

When you invoke a search, the server-side logic creates a ProDataSet and populates its wo_mstr and wod_det temp-tables with records from the corresponding database tables and sends it to the client-side logic. When you save, the client-side logic creates a ProDataSet and populates its wo_mstr temp-table with modified work order records and sends it to the server-side logic.

Work Order Master Temp-Table Definition

The wo_mstr temp-table definition is contained in the `woscwo.i` include file. The temp-table is defined like the wo_mstr database table, so it contains all fields that are defined in the wo_mstr database table schema. When you need to add a calculated field to this temp-table definition, you add it to the `woscwox.i` include file which `woscwo.i` references.

Work Order Detail Temp-Table Definition

The wod_det temp-table definition is contained in the `woscwod.i`.include file. The temp-table is defined like the wod_det database table, so it contains all fields that are defined in the wod_det database table schema. When you need to add a calculated field to this temp-table definition, you add it to the `woscwodx.i` include file which `woscwod.i` references.

Enabling the Personalization Architecture

To begin using the architecture, invoke Workbench Control (22.20.24) and set Enable Additional Default Fields to Yes. Setting this field to Yes enables the use of additional columns and the user exits.; see Figure 5.1 on page 37. This allows you to easily enable or disable the architecture.

Specifying Additional Columns

You can use the following Additional Default Display fields in Workbench Control (22.20.24):

Work Order. This is a comma-separated list of fields to add to the work order grid default column display list. This can include any field that is defined in the wo_mstr temp-table which is defined in the include files woscwo.i and woscwox.i.

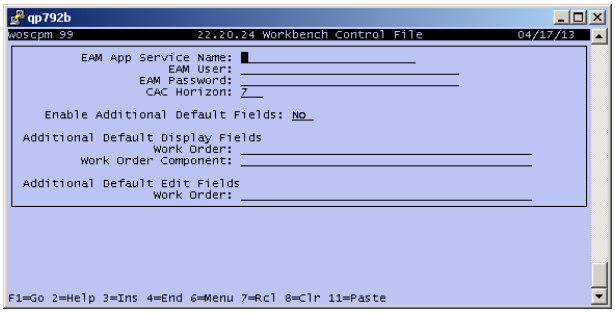
Work Order Component. This is a comma-separated list of fields to add to the work order component detail grid default column display list. This can include any field that is defined in the wod_det temp-table which is defined in the include files woscwod.i and woscwodx.i.

You can use the following Additional Default Edit fields in the Workbench Control (22.20.24):

Work Order. This is a comma-separated list of fields that are both displayed and editable. When you want to specify the order in which fields entered here are displayed, enter them in the desired order in the Additional Default Display Fields / Work Order field.

Note Editable fields for the work order component table are not supported at this time.

Fig. 5.1
Workbench Control, New Fields



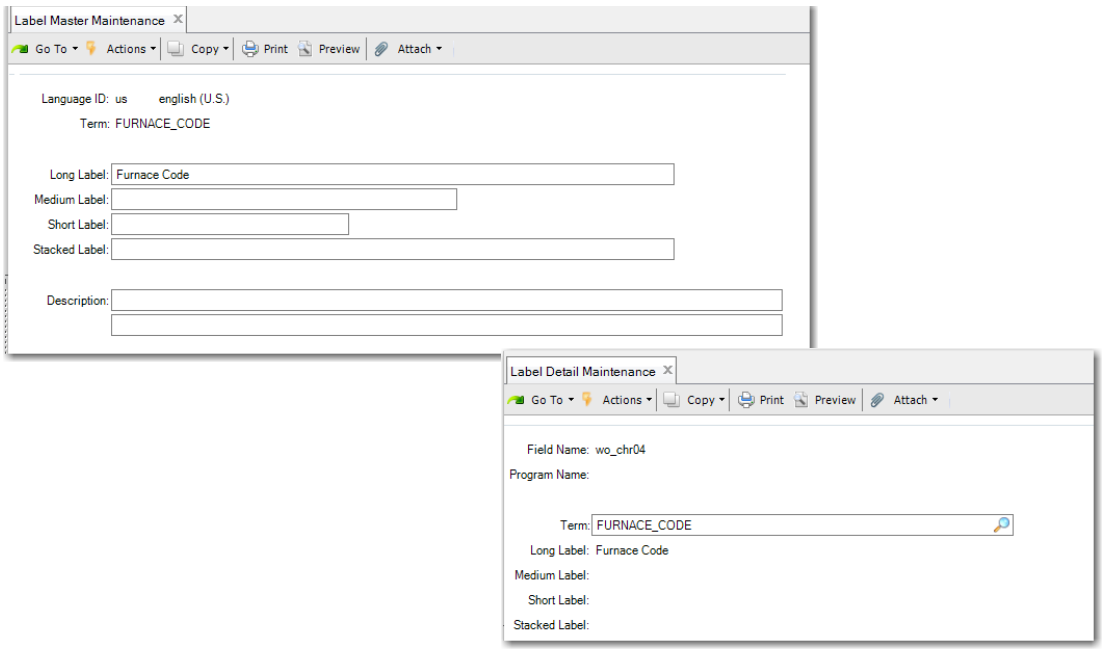
Adding Column Header Labels

The column header label of a newly-added column can be specified as follows:

1. Using Label Master Browse (36.4.17.2) locate an existing label term that you can use for the column's header label. When no appropriate label exists, use Label Master Maintenance (36.4.17.1) to create one.
2. Using Label Detail Maintenance (36.4.17.5) create an entry for the column. Reference the label term of the label selected or created in the previous step.

Example You want to add the field wo__chr04 to all work order header grids. Its column label is *Furnace Code*. Figure 5.2 shows you the setup for the new column label in Label Master Maintenance as well as Label Detail Maintenance.

Fig. 5.2
Label Master Maintenance (36.4.17.1) and Label Detail Maintenance (36.4.17.5)



User Exits

The architecture provides a way to add additional processing logic. You add processing logic by adding code to the user exit programs.

The user exit programs have access to the ProDataSets that are used to pass data between the workbenches client and the server-side OpenEdge AppServer. Depending on what needs to be done, the user exit programs can read data from, and write data to, the ProDataSet's temp-tables.

- Post Search Read (woscrtrvx.p): The system invokes this user exit program immediately after search processing. For example, you can add code to it to populate calculated column values.
- Work Order Pre/Post Save Update: The system invokes these user exit programs during save processing. For each work order that has changes (add/modify/delete) to be saved, the save process:

- Starts a database update transaction.
- Pre Save Update (woscpreupdatex.p): The system invokes this user exit program immediately after starting the transaction. This is where you add code to do processing just before the system writes changes to a work order to the database. For example, you can implement additional validation logic here. The result of the validation can cause the system to cancel the update transaction and display the associated red error icon and error message in the MSW/PSW. See the “Add a Validation” on page 44 scenario.
- Invokes the work order save logic
- Post Save Update (woscpostupdatex.p): The system invokes this user exit program immediately after invoking the work order save logic. This is where you can add code to do processing right after the system writes changes to a work order to the database. For example, this is where you can add logic to save the changes you make to added editable fields. You can save the changes either to the wo_mstr or wod_det tables or to other tables. See the “Add an Editable wo_mstr Column to a Default Column Display List and Save Updates” on page 40.
- Commits the transaction.

- Post Save Read (woscupdatereadx.p): The system invokes this user exit program right after save processing has completed for all work orders. This is where you can add code to populate calculated column values, in the event that saving changes to work orders would require a recalculation of calculated column values. See the “Add a Calculated Display-Only Field” on page 41.
- Post Component Availability Check Process (wosccacx.p): The system invokes this user exit program immediately after CAC processing. This is where you can add code to populate calculated column values that are based on CAC values.
- Calculate Capacity Quantity Completed (wosccqcx.p): The system invokes this user exit for each production order that is processed during a search. It provides a way to modify the quantity completed that the system uses for capacity calculation purposes. As a default, the quantity completed that the system uses for capacity calculation purposes is the sum of the production order’s quantity completed and quantity rejected. This user exit lets you override it; for example, you can use the quantity completed at a particular operation instead of the production order’s quantity completed.

  The system uses the capacity quantity completed value as follows:

  - PSW panels:

    Required capacity is based on the following:

    Capacity open quantity (quantity ordered - capacity quantity completed)

    In the left-hand structured panel, the release date and shift band totals—Required, Remaining, Cum Remaining, Carry-over, and so on—are based on capacity open quantity.
  - MSW Capacity Panel:

    The Required Capacity and Scheduled Quantity rows are based on capacity open quantity.
  - Workbenches-embedded Production Order Maintenance:

    In the Details Panel, the Required Capacity (Hrs) field is based on capacity open quantity.

  Other panels, such as the MSW Scheduling and Supply/Demand panels, are not affected.

Note These user exit programs are invoked only when you set the field Enable Additional Default Fields in Workbench Control (22.20.24) to Yes.

Customization Scenarios

This section describes some typical customization scenarios and includes examples. The following scenarios are described:

- Add a display-only field to a default column display list
- Add an editable wo_mstr column to a default column display list and save updates
- Add a calculated display-only field
- Add a validation
- Add an editable calculated column to a default column display list and save updates

Add a Display-Only Field to a Default Column Display List

In this scenario, the field exists in the schema for either the wo_mstr or the wod_det tables.

1. Add the field to the additional default display fields list as described above in “Specifying Additional Columns” on page 37.
2. Add a column label for the field as described above in “Adding Column Header Labels” on page 38.

Add an Editable wo_mstr Column to a Default Column Display List and Save Updates

In this scenario, the field exists in the schema for the wo_mstr table. The system saves any changes you make to the column values in the database record.

1. Add the field to the additional default display fields list as described above in “Specifying Additional Columns” on page 37.
2. Add a column header label for the field as described above in “Adding Column Header Labels” on page 38.
3. Add the field to the additional default edit fields list as described above in “Specifying Additional Columns” on page 37.

In the Post Save Update user exit, add code to save the value of the field into the wo_mstr database record. The following example code writes the value of wo__chr04 to the database record:

```
/* woscpostupdatex.p - MSW/PSW Personalization Architecture user exit program */
/* Copyright 1986 QAD Inc. All rights reserved.                               */
/* $Id:: woscpostupdatex.p 4721 2013-04-04 20:52:09Z wug                $: */
/*V8:ConvertMode=NoConvert                                                   */

/*
 * Post Save Update user exit program
 *
 * For more information see the Personalization Architecture
 * section in the MSW/PSW Administrative Guide
 */

define variable svnId{&SEQUENCE} as character no-undo initial "svnId $Id:
woscpostupdatex.p 4721 2013-04-04 20:52:09Z wug $".

{mfdeclre.i}
```

```
{wodstt.i}

/*THE INPUT-OUTPUT PARAMETER DATASET REFERS TO A DATASET
 *THAT IS DEFINED BY wodstt.i.  THE TEMP-TABLES DEFINED IN
 *THIS DATASET (tt-wo_mstr, tt-wod_det etc) CAN BE REFERENCED
 *STATICALLY.  THIS DATASET IS WHAT IS USED AS INPUT TO THE
 *WO UPDATE API*/

define input parameter ipDomain as character no-undo.
define input parameter ipLot as character no-undo.
define input-output parameter dataset for dsWorkOrder.
define output parameter opSuccessful as logical no-undo initial true.
/*hdswoscwxad REFERS TO THE DATASET THAT IS SENT BY THE
 *CLIENT. IT CONTAINS THE TEMP-TABLES tt-wo_mstr (defined
 *BY woscwo.i/woscwox.i), tt-wod_det (defined by
 *woscwod.i/woscwodx.i), and tt-wr_route (defined by
 *woscwr.i).  THE CONTENTS OF THIS DATASET WAS PREVIOUSLY
 *LOADED INTO THE INPUT-OUTPUT PARAMETER DATASET.*/

define shared variable hdswoscwxad as handle no-undo.

/*hbtt-wo_mstr IS A HANDLE REFERENCING tt-wo_mstr.  THE CONTENTS
 *OF tt-wo_mstr CAN BE ACCESSED USING DYNAMIC TECHNIQUES.  IT
 *WOULD BE POSSIBLE TO TYPECAST hbtt-wo_mstr TO A PROCEDURE
 *USING THE TABLE-HANDLE MECHANISM IF YOU WANTED TO USE THE
 *woscwo.i/woscwox.i ETC STATIC DEFINITIONS.*/

define variable hbtt-wo_mstr as handle no-undo.
hbtt-wo_mstr = hdswoscwxad:get-buffer-handle("tt-wo_mstr").

/*your code goes here*/

find tt-wo_mstr where tt-wo_mstr.wo_domain = ipDomain
and tt-wo_mstr.wo_lot = ipLot
no-error.

if available tt-wo_mstr
and
(row-state(tt-wo_mstr) = ROW-CREATED or row-state(tt-wo_mstr) = ROW-MODIFIED)
then do:
   find wo_mstr where wo_mstr.wo_domain = ipDomain
   and wo_mstr.wo_lot = ipLot
   exclusive-lock.

   wo_mstr.wo__chr04 = tt-wo_mstr.wo__chr04.
end.

opSuccessful = true.
```

Add a Calculated Display-Only Field

In this scenario, the field does not exist in the schema for either the wo_mstr or the wod_det database tables. Instead, it is defined in the temp-table definition and is populated using processing code in user exits. This description uses as an example the following field:

- Field name: wo_release_day
- Column label: Release Day
- Data Type: character
- Content: the day of the week of the work order's Release Date, i.e. Monday, Tuesday, Wednesday, …

1 Add the field definition to the include file woscwox.i (tt-wo_mstr extensions):

```
field wo_release_day as character
```

2 Recompile the code that references the include files woscwo.i (tt-wo_mstr):

- woscrtrv.p
- woscpm.p
- woscttps.p

3 Add the field wo_release_day to the additional default display fields list as described in “Specifying Additional Columns” on page 37.

4 Add a column label for the field as described above in “Specifying Additional Columns” on page 37.

5 In the Post Search Read user exit, add code to populate the new field and compile it:

```
/* woscrtrvx.p - MSW/PSW Personalization Architecture user exit program       */
/* Copyright 1986 QAD Inc. All rights reserved.                               */
/* $Id:: woscrtrvx.p 4721 2013-04-04 20:52:09Z wug                         $: */
/*V8:ConvertMode=NoConvert                                                    */

/*
 * Post search read user exit program
 *
 * For more information see the Personalization Architecture
 * section in the MSW/PSW Administrative Guide
 */

define variable svnId{&SEQUENCE} as character no-undo initial "svnId $Id: woscrtrvx.p
4721 2013-04-04 20:52:09Z wug $".

{mfdeclre.i}

define input parameter dataset-handle    hdsInputData.
define input parameter ipTodaysDate      as date no-undo.
define input parameter ipStartDate       as date no-undo.
define input parameter ipEndDate         as date no-undo.
define input parameter ipSequencingHorizon as integer no-undo.
define input parameter ipIncludeItemsWithNoSupplyDemand as logical no-undo.
define input parameter ipRetrieveRoutingRecords as logical no-undo.
define input parameter ipCACEngineEnabled as logical no-undo.
define input-output parameter dataset-handle iphdsinout.

define variable whtt-wo_mstr as handle no-undo.
define variable whtt-wod_det as handle no-undo.

/*THE INPUT-OUTPUT PARAMETER DATASET-HANDLE REFERS TO A DATASET
 *THAT IS DEFINED BY woscrtds.i.  THE CODE BELOW GETS
 *A HANDLE TO THE tt-wo_mstr AND tt-wod_det TEMP-TABLES.
 *YOU CAN CALL A PROCEDURE USING THE TABLE-HANDLE CONSTRUCT IF
 *YOU WANT TO ACCESS THESE TABLES STATICALLY.*/

whtt-wo_mstr = iphdsinout:get-buffer-handle("tt-wo_mstr").
whtt-wo_mstr = whtt-wo_mstr:table-handle.

whtt-wod_det = iphdsinout:get-buffer-handle("tt-wod_det").
whtt-wod_det = whtt-wod_det:table-handle.

/*your code goes here*/

/* RESOURCE TYPE CONSTANTS */
&SCOPED-DEFINE PRODUCTION_LINE      0
&SCOPED-DEFINE WORK_CENTER          1

{gplabel.i}
{woscwo.i}

define variable wWeekdays as character extent 7.
```

```
wWeekdays[1] = getTermLabel("SUNDAY", 60).
wWeekdays[2] = getTermLabel("MONDAY", 60).
wWeekdays[3] = getTermLabel("TUESDAY", 60).
wWeekdays[4] = getTermLabel("WEDNESDAY", 60).
wWeekdays[5] = getTermLabel("THURSDAY", 60).
wWeekdays[6] = getTermLabel("FRIDAY", 60).
wWeekdays[7] = getTermLabel("SATURDAY", 60).

run process_tt-wo_mstr(input-output table-handle whtt-wo_mstr by-reference).

procedure process_tt-wo_mstr:
   define input-output parameter table for tt-wo_mstr.

   for each tt-wo_mstr:
      tt-wo_mstr.wo_release_day = wWeekdays[weekday(tt-wo_mstr.wo_rel_date)].
   end.
end procedure.
```

6 In the Post Save Read user exit, add code to populate the new field and compile it. Notice that it is almost exactly the same as the Post Search Read user exit code:

```
/* woscupdatereadx.p - MSW/PSW Personalization Architecture user exit program */
/* Copyright 1986 QAD Inc. All rights reserved.                               */
/* $Id:: woscupdatereadx.p 4721 2013-04-04 20:52:09Z wug                  $: */
/*V8:ConvertMode=NoConvert                                                    */

/*
 * Post update read user exit program
 *
 * For more information see the Personalization Architecture
 * section in the MSW/PSW Administrative Guide
 */

define variable svnId{&SEQUENCE} as character no-undo initial "svnId $Id:
woscupdatereadx.p 4721 2013-04-04 20:52:09Z wug $".

{mfdeclre.i}

define input-output parameter dataset-handle iphdsinout.
define input parameter ipTodaysDate      as date no-undo.
define input parameter ipStartDate       as date no-undo.
define input parameter ipEndDate         as date no-undo.
define input parameter ipSequencingHorizon as integer no-undo.
define input parameter ipIncludeItemsWithNoSupplyDemand as logical no-undo.
define input parameter ipRetrieveRoutingRecords as logical no-undo.
define input parameter ipCACEngineEnabled as logical no-undo.

/*THE INPUT-OUTPUT PARAMETER DATASET-HANDLE REFERS TO THE DATASET
 *THAT IS SENT BACK TO THE CLIENT.  IT CONTAINS THE TEMP-TABLES
 *tt-wo_mstr (defined BY woscwo.i/woscwox.i), tt-wod_det (defined
 *by woscwod.i/woscwodx.i), and tt-wr_route (defined by
 *woscwr.i). THE CODE BELOW GETS A HANDLE TO THE tt-wo_mstr
 *AND tt-wod_det TEMP-TABLES.  YOU CAN CALL A PROCEDURE
 *USING THE TABLE-HANDLE CONSTRUCT IF YOU WANT TO ACCESS
 *THESE TABLES STATICALLY.*/

define variable whtt-wo_mstr as handle no-undo.
define variable whtt-wod_det as handle no-undo.

whtt-wo_mstr = iphdsinout:get-buffer-handle("tt-wo_mstr").
whtt-wo_mstr = whtt-wo_mstr:table-handle.

whtt-wod_det = iphdsinout:get-buffer-handle("tt-wod_det").
whtt-wod_det = whtt-wod_det:table-handle.

/*your code goes here*/

/* RESOURCE TYPE CONSTANTS */
&SCOPED-DEFINE PRODUCTION_LINE      0
&SCOPED-DEFINE WORK_CENTER          1
```

```
{gplabel.i}
{woscwo.i}

define variable wWeekdays as character extent 7.

wWeekdays[1] = getTermLabel("SUNDAY", 60).
wWeekdays[2] = getTermLabel("MONDAY", 60).
wWeekdays[3] = getTermLabel("TUESDAY", 60).
wWeekdays[4] = getTermLabel("WEDNESDAY", 60).
wWeekdays[5] = getTermLabel("THURSDAY", 60).
wWeekdays[6] = getTermLabel("FRIDAY", 60).
wWeekdays[7] = getTermLabel("SATURDAY", 60).

run process_tt-wo_mstr(input-output table-handle whtt-wo_mstr by-reference).

procedure process_tt-wo_mstr:
   define input-output parameter table for tt-wo_mstr.

   for each tt-wo_mstr:
      tt-wo_mstr.wo_release_day = wWeekdays[weekday(tt-wo_mstr.wo_rel_date)].
   end.
end procedure.
```

Add a Validation

In this scenario, you add additional validation logic. This is done by adding logic to the Pre Save Update user exit program. There can be validations on any field, whether it's a field that is already implemented in the MSW/PSW—for example, Release Date, Remarks—or a field to be implemented—for example, wo__chr04.

The validation logic works like any other validation in the MSW/PSW work order save logic: that is, when the validation fails, the:

- Save request is rejected
- Red error icon is displayed in the MSW/PSW
- Reason message is added to the icon's tooltip text.

In this example, a validation is added to the work order Remarks (wo_rmks) field to ensure that it is always non-blank.

```
/* woscpreupdatex.p - MSW/PSW Personalization Architecture user exit program  */
/* Copyright 1986 QAD Inc. All rights reserved.                               */
/* $Id:: woscpreupdatex.p 4721 2013-04-04 20:52:09Z wug                  $: */
/*V8:ConvertMode=NoConvert                                                    */

/*
 * Pre Save Update user exit program
 *
 * For more information see the Personalization Architecture
 * section in the MSW/PSW Administrative Guide
 */

define variable svnId{&SEQUENCE} as character no-undo initial "svnId $Id:
woscpreupdatex.p 4721 2013-04-04 20:52:09Z wug $".

{mfdeclre.i}
{wodstt.i}

/*THE INPUT-OUTPUT PARAMETER DATASET REFERS TO A DATASET
 *THAT IS DEFINED BY wodstt.i.  THE TEMP-TABLES DEFINED IN
 *THIS DATASET (tt-wo_mstr, tt-wod_det etc) CAN BE REFERENCED
 *STATICALLY.  THIS DATASET IS WHAT IS USED AS INPUT TO THE
 *WO UPDATE API*/

define input parameter ipDomain as character no-undo.
```

```
define input parameter ipLot as character no-undo.
define input-output parameter dataset for dsWorkOrder.
define output parameter opSuccessful as logical no-undo initial true.

/*hdswoscwxad REFERS TO THE DATASET THAT IS SENT BY THE
 *CLIENT. IT CONTAINS THE TEMP-TABLES tt-wo_mstr (defined
 *BY woscwo.i/woscwox.i), tt-wod_det (defined by
 *woscwod.i/woscwodx.i), and tt-wr_route (defined by
 *woscwr.i).  THE CONTENTS OF THIS DATASET WAS PREVIOUSLY
 *LOADED INTO THE INPUT-OUTPUT PARAMETER DATASET.*/

define shared variable hdswoscwxad as handle no-undo.

/*hbtt-wo_mstr IS A HANDLE REFERENCING tt-wo_mstr.  THE CONTENTS
 *OF tt-wo_mstr CAN BE ACCESSED USING DYNAMIC TECHNIQUES.  IT
 *WOULD BE POSSIBLE TO TYPECAST hbtt-wo_mstr TO A PROCEDURE
 *USING THE TABLE-HANDLE MECHANISM IF YOU WANTED TO USE THE
 *woscwo.i/woscwox.i ETC STATIC DEFINITIONS.*/

define variable hbtt-wo_mstr as handle no-undo.
hbtt-wo_mstr = hdswoscwxad:get-buffer-handle("tt-wo_mstr").

/*your code goes here*/

find tt-wo_mstr where tt-wo_mstr.wo_domain = ipDomain
and tt-wo_mstr.wo_lot = ipLot
no-error.

if available tt-wo_mstr
and (row-state(tt-wo_mstr) = ROW-CREATED or row-state(tt-wo_mstr) = ROW-MODIFIED)
and tt-wo_mstr.wo_rmks = ""
then do:
   /*NOTE: SET &FIELDNAME='oid_wo_mstr' TO APPLY THE MESSAGE
    *TO ONLY THE ROW ERROR ICON*/

    /*MESSAGE 40="BLANK NOT ALLOWED"*/

   {pxmsg.i
   &MSGNUM=40 &ERRORLEVEL=3 &MSGARG1='' &FIELDNAME='wo_rmks'
   &CALLINGPGM=execname &CONTEXT=ipLot}

   /*THIS CAUSES THE UPDATE TO BE CANCELLED*/
   opSuccessful = false.
   return.
end.

opSuccessful = true.
```

Add an Editable Calculated Column to a Default Column Display List and Save Updates

In this scenario, the field does not exist in the schema for the wo_mstr table. Instead, it is defined in the temp-table definition. It is also defined as an editable field.

- Field name: moveDates
- Column label: Move Dates
- Data Type: character
- Content: blank. When you enter Yes in this field, seven days are added to the Work Order Release Date and Due Date fields before saving.

1. Add the field definition to the include file woscwox.i (tt-wo_mstr extensions):
   ```
   field moveDates as character
   ```
2. Recompile the code that references the include files woscwo.i (tt-wo_mstr):

- woscrtrv.p
- woscpm.p
- woscttps.p

3 Add the field moveDates to the additional default display and edit fields list as described in “Specifying Additional Columns” on page 37.

4 Add a column label for the field as described above in “Specifying Additional Columns” on page 37.

5 In the Pre Update user exit, add code to check the value of moveDates, and when set to Yes, add seven days to the work order Release Date and Due Date fields:

```
/* woscpreupdatex.p - MSW/PSW Personalization Architecture user exit program  */
/* Copyright 1986 QAD Inc. All rights reserved.                                */
/* $Id:: woscpreupdatex.p 4721 2013-04-04 20:52:09Z wug                  $: */
/*V8:ConvertMode=NoConvert                                                    */

/*
 * Pre Save Update user exit program
 *
 * For more information see the Personalization Architecture
 * section in the MSW/PSW Administrative Guide
 */

define variable svnId{&SEQUENCE} as character no-undo initial "svnId $Id:
woscpreupdatex.p 4721 2013-04-04 20:52:09Z wug $".

{mfdeclre.i}
{wodstt.i}

/*THE INPUT-OUTPUT PARAMETER DATASET REFERS TO A DATASET
 *THAT IS DEFINED BY wodstt.i.  THE TEMP-TABLES DEFINED IN
 *THIS DATASET (tt-wo_mstr, tt-wod_det etc) CAN BE REFERENCED
 *STATICALLY.  THIS DATASET IS WHAT IS USED AS INPUT TO THE
 *WO UPDATE API*/

define input parameter ipDomain as character no-undo.
define input parameter ipLot as character no-undo.
define input-output parameter dataset for dsWorkOrder.
define output parameter opSuccessful as logical no-undo initial true.

/*hdswoscwxad REFERS TO THE DATASET THAT IS SENT BY THE
 *CLIENT. IT CONTAINS THE TEMP-TABLES tt-wo_mstr (defined
 *BY woscwo.i/woscwox.i), tt-wod_det (defined by
 *woscwod.i/woscwodx.i), and tt-wr_route (defined by
 *woscwr.i).  THE CONTENTS OF THIS DATASET WAS PREVIOUSLY
 *LOADED INTO THE INPUT-OUTPUT PARAMETER DATASET.*/

define shared variable hdswoscwxad as handle no-undo.

/*hbtt-wo_mstr IS A HANDLE REFERENCING tt-wo_mstr.  THE CONTENTS
 *OF tt-wo_mstr CAN BE ACCESSED USING DYNAMIC TECHNIQUES.  IT
 *WOULD BE POSSIBLE TO TYPECAST hbtt-wo_mstr TO A PROCEDURE
 *USING THE TABLE-HANDLE MECHANISM IF YOU WANTED TO USE THE
 *woscwo.i/woscwox.i ETC STATIC DEFINITIONS.*/

define variable hbtt-wo_mstr as handle no-undo.
hbtt-wo_mstr = hdswoscwxad:get-buffer-handle("tt-wo_mstr").

/*your code goes here*/

define variable moveDates as character no-undo.
define variable newreldate as date no-undo.
define variable newduedate as date no-undo.

/*LOOK AT ROW STATE TO DETERMINE ADD/MODIFY/DELETE*/
```

```
find tt-wo_mstr where tt-wo_mstr.wo_domain = ipDomain
and tt-wo_mstr.wo_lot = ipLot
no-error.

if available tt-wo_mstr then do:
   if row-state(tt-wo_mstr) = ROW-MODIFIED or row-state(tt-wo_mstr) = ROW-CREATED
   then do:
      /*hbtt-wo_mstr IS A HANDLE THAT REFERS TO THE tt-wo_mstr TEMP-TABLE
       *THAT'S IN THE PRODATASET SENT BY THE CLIENT.  THIS TEMP-TABLE
       *IS DEFINED IN woscwo.i/woscwox.i.   HERE WE ARE ACCESSING
       *ITS FIELD VALUES USING DYNAMIC CODING TECHNIQUES.  IT WOULD BE
       *POSSIBLE TO TYPECAST hbtt-wo_mstr TO AN INTERNAL PROCEDURE
       *USING THE TABLE-HANDLE MECHANISM IF YOU WANTED TO USE THE
       *woscwo.i/woscwox.i STATIC DEFINITION.*/

      hbtt-wo_mstr = hdswoscwxad:get-buffer-handle("tt-wo_mstr").

      hbtt-wo_mstr:find-first("where tt-wo_mstr.wo_domain = " + quoter(ipDomain)
      + " and tt-wo_mstr.wo_lot = " + quoter(ipLot), NO-LOCK).

      moveDates = hbtt-wo_mstr:buffer-field("moveDates"):buffer-value().

      if moveDates begins "y" then do:
         newreldate = hbtt-wo_mstr:buffer-field("wo_rel_date"):buffer-value().
         newduedate = hbtt-wo_mstr:buffer-field("wo_due_date"):buffer-value().

         /*NOTE THAT THE tt-wo_mstr TEMP-TABLE REFERENCED IN THE FOLLOWING
          *FIELD ASSIGNMENTS IS DEFINED IN wodstt.i.  wodstt.i DEFINES THE
          *PRODATASET USED AS INPUT TO THE WO API PROGRAM (wowoxr.p).  WHAT HAPPENS
          *BEFORE THE API IS INVOKED IS THAT TEMP-TABLE CONTENTS IN THE PRODATASET
          *SENT BY THE CLIENT ARE COPIED INTO THE TEMP-TABLES DEFINED IN wodstt.i.*/

         tt-wo_mstr.wo_rel_date = newreldate + 7.
         tt-wo_mstr.wo_due_date = newduedate + 7.
      end.
   end.
end.
else do:
   /*THERE IS NO AFTER-IMAGE RECORD SO THERE MUST BE A BEFORE-IMAGE RECORD
    *IN OTHER WORDS, IT'S A DELETE REQUEST.*/

   find tt-wo_mstr-before where tt-wo_mstr-before.wo_domain = ipDomain
   and tt-wo_mstr-before.wo_lot = ipLot
   no-error.

   if available tt-wo_mstr-before then do:
   end.
   else do:
   end.
end.

opSuccessful = true.
```