Learning QAD Web UI

With the QAD Web UI, you can quickly get things done from anywhere, anytime, on any device.

- Menu Bar on page 242
- Action Centers on page 244
- Menu Search on page 249
- User Options on page 250
- Favorites on page 254
- Views on page 255
- Browses on page 257
- Forms on page 275
- Actions on page 279
- Stored Views on page 281
- Activity on page 286
- Attachments on page 290
- Drill-Down Links on page 292
- Messages on page 295
- Reports on page 296
- Approval Delegation on page 389
- Future Approvals on page 391

Menu Bar



The menu bar starts with the QAD icon, located in the upper left. Clicking the QAD icon takes you to your home page.

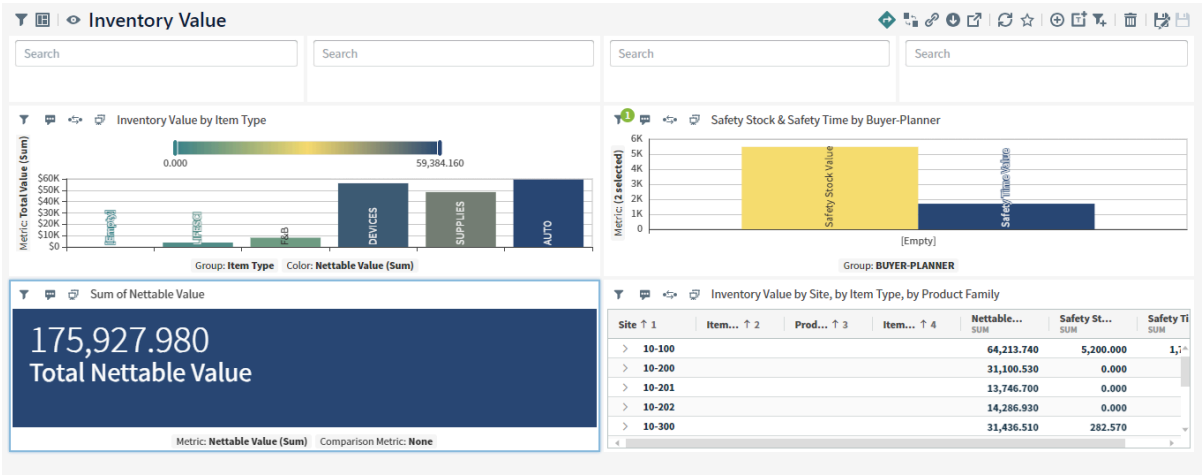
After the QAD icon, the menu bar includes:

- **Menu Selector** — from the menu selector drop-down, choose a role menu from the list of role menus that are available to you. Alternatively, choose Favorites, located at the bottom of the drop-down list, to have the menu bar display all your favorite menu items.
- **Action Centers** — choose an action center to view from the Action Centers drop-down menu listing all the action centers available to you. Various role-based action centers are included. Note that you can set an action center as your home page.
- **Role Menu Options** — the selections here can vary depending on your current role, as selected from the role menu. For example, if your role is Customer Service Manager, the menu options include Sales Orders, Customers, Sales Quotes, Items, and Requisitions.
- **Workspace** — choose the workspace from the drop-down of workspaces available to you, where workspace is the domain and entity within which you are working. Changing workspaces applies to all browser tabs running in the QAD Web UI. Be sure you have saved your work in all the tabs before changing workspaces. When you change workspaces, the system will ask you to confirm that you are ready to change workspaces.
- **Menu Search** — use the Menu Search icon to find all the menu items you can access on the QAD Web UI.
- **Inbox** — use the Inbox icon to see your messages. If you have new messages, an indicator on the Inbox icon tells you how many new messages you have. The Inbox includes Tasks, including Requisition Approvals, and Notifications of system activity and scheduled report availability.
- **Help** — from the Help icon, you can view the online help, use Guide Me, open process maps, access the QAD Document Library, and view the About page, including the copyright. The online help opens in the user's language set up in the profile. If the online help is not available in the specified language, it opens in English.
- **User Options** — from the User Options icon, you can:
  - View your user profile settings and change your language and locale
  - Review submitted background processing

- Review submitted approvals and set approval delegation
- View and organize your favorites
- Set the current page as your home page or reset to the default home page
- Sign out of the QAD Web UI

Action Centers

Action Centers provide managers with insights into their business in a visual format. They offer next-generation analytics, which allow managers to monitor KPIs that provide insights into particular areas of responsibility.



Predefined QAD action centers include the following:

- Revenue & Margin Analysis action center
- Customer Sales action center
- Inventory Value action center
- Purchasing Commitments and Spending action center

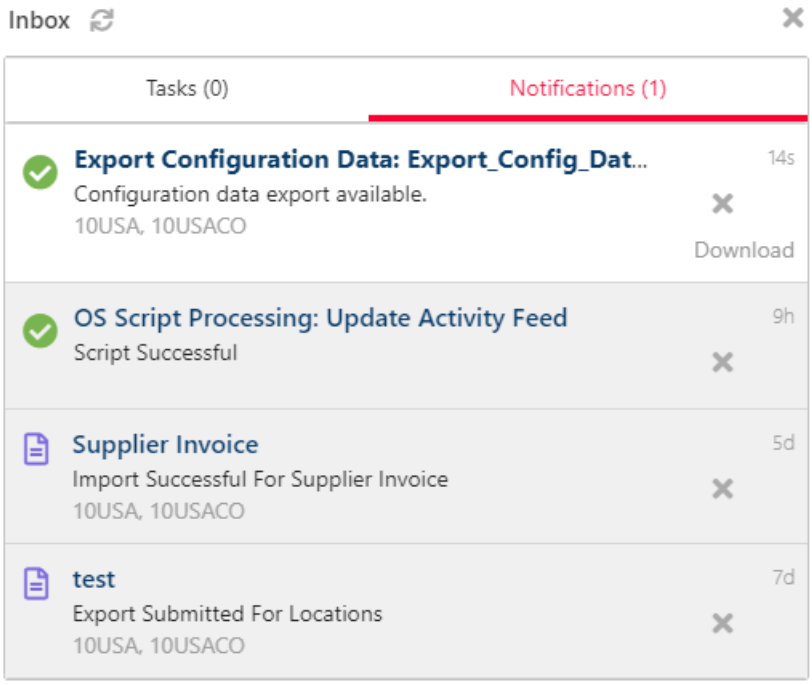
For more information, see: Action Centers on page 400.

Inbox

Click Inbox on the menu bar on page 242 to see your messages. The Inbox lets you know of how many new messages you have with an indicator on the icon.

Inbox messages include Tasks and Notifications.

- Tasks include approval requests such as Requisition Approvals.
- Notifications indicate activity such as system events, comments from other users, and updates on transactions that you are following.



View Inbox

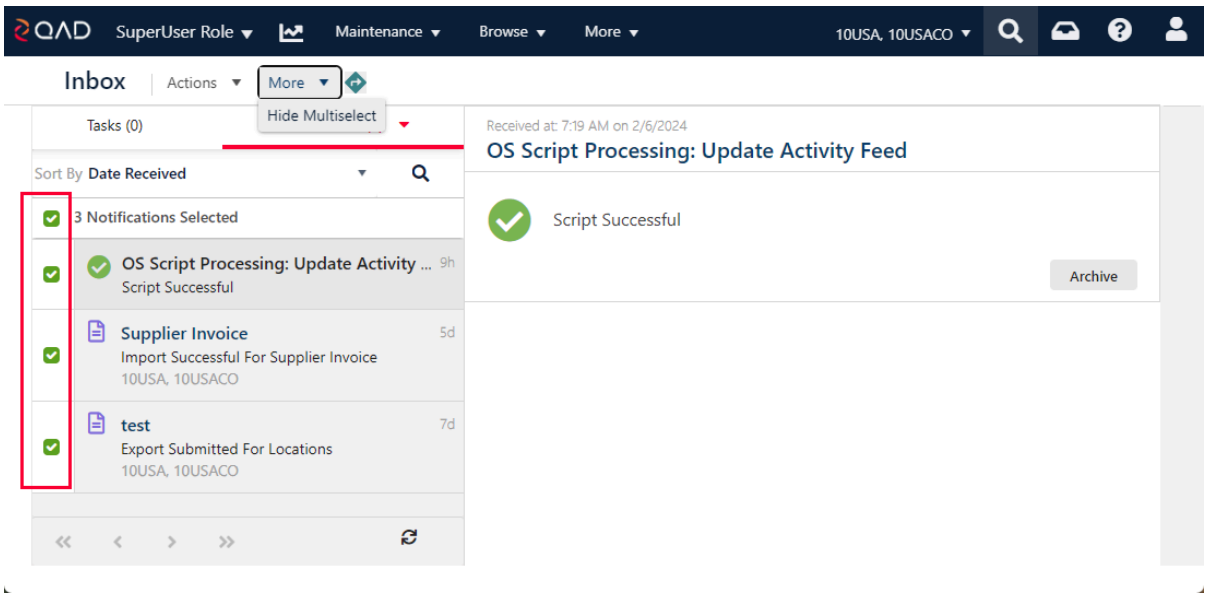
Initially, the Inbox displays a listing of your recent messages in a drop-down box. You can toggle between the Tasks and Notifications tabs, and scroll within each tab to view the messages. Some notifications link to the related views for quick access. And for some notification messages, you can click Comment and enter your comments in the comment box directly.

To view all task messages and notification messages, you can go to the Extended Inbox.

To go to Extended Inbox, click View Inbox.

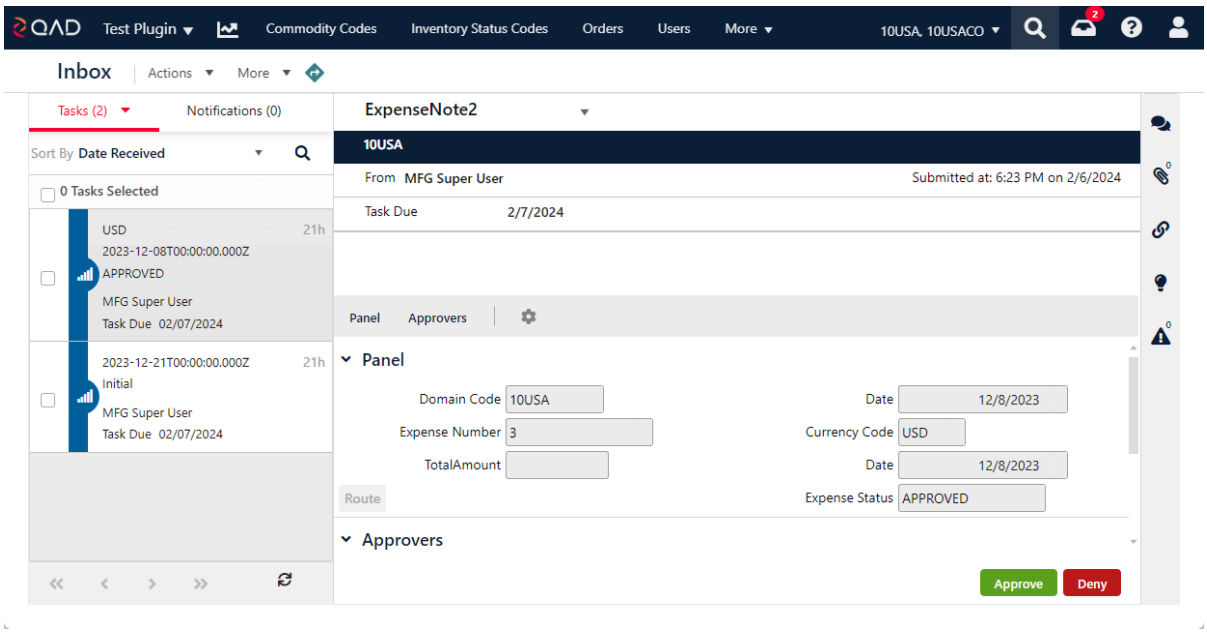
Multiselect

Use the Multiselect option to manage tasks and notifications in bulk. You can show or hide the Multiselect option from the More drop-down. Use the Select All checkbox to select all the notifications on the current page.



Tasks in Extended Inbox

On the Task tab in the Extended Inbox, the system displays all the active tasks by default. You can sort the task messages by Date Received, Due Date, From, or Domain. You can also filter the messages by clicking the lookup browse icon and using searching conditions.

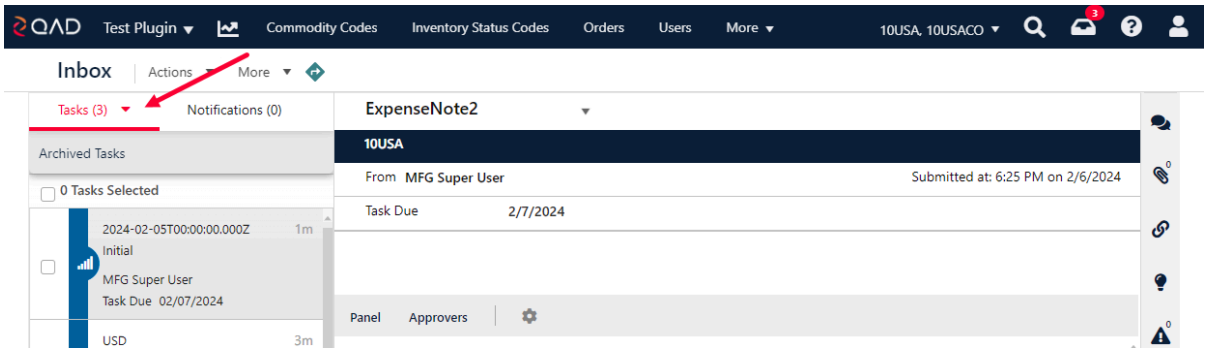


Click a specific task message in the list, and the system displays the approval screen to the right. You can approve, deny, or route the approval request. When you approve or deny a request, you can enter your comments on the request and also update the editable fields. To route

the approval request to another user, click the More button and choose Route. When you route the approval request, you can also enter your remarks.

You can approve or deny the approval requests in bulk. To do a mass approval, select the boxes at the beginning of all the listed approval requests that you want to handle. And then from Actions, choose Approve to approve all the selected requests. Choose Deny to deny all the selected requests. You can also give your comments when you approve or deny.

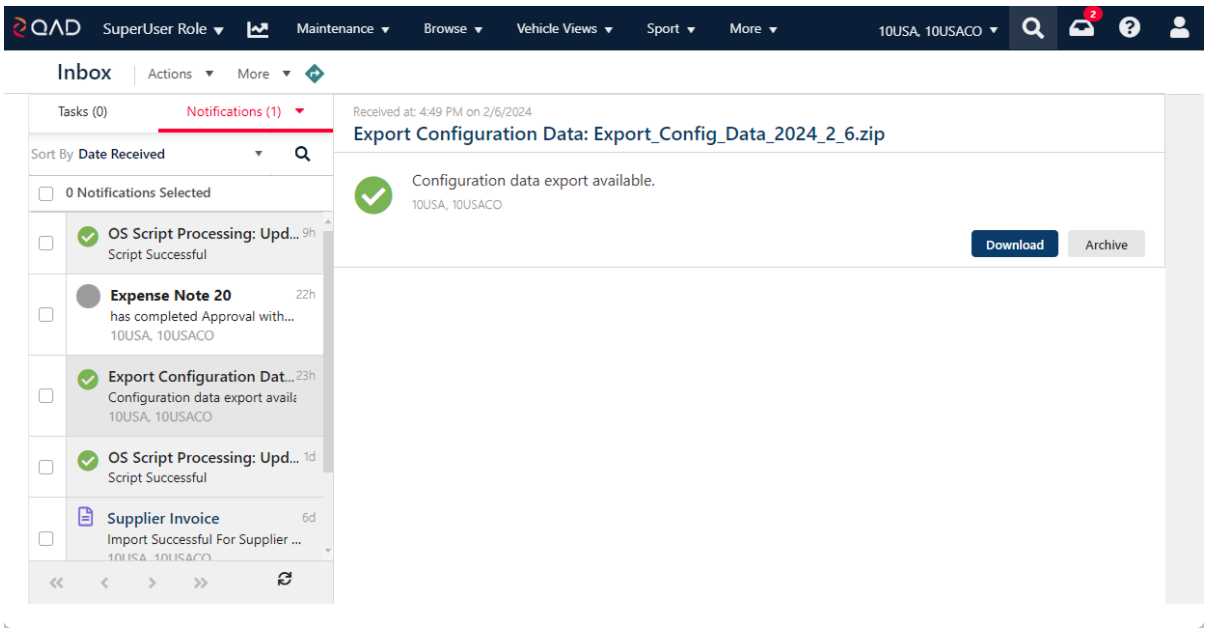
You can view archived tasks by clicking the small drop-down button beside Tasks. Archived tasks show all tasks that are completed by you. You cannot take any action on these tasks. To switch from Archived Tasks back to Active Tasks, click the small drop-down button beside Tasks.



Notifications in Extended Inbox

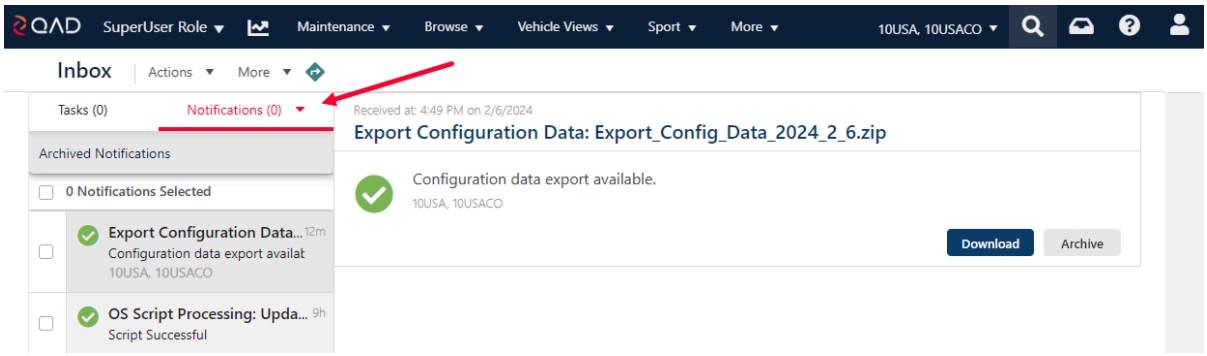
On the Notifications tab in the Extended Inbox, the system displays all the active notifications by default. You can sort the notification messages by Date Received or Workspace. You can also filter the messages by clicking the lookup browse icon and using searching conditions. For example, you can search by From and enter System to search the notifications that are not generated from user activities or comments. An example of this kind of notification is a notification to show that an approval request has been approved.

Click a specific notification message in the list, and the system displays the details page to the right. You can comment on the notification message or archive the notification message. If the notification is about a report for downloading, the system displays the Download button instead of the Comment button. To download the report, click Download.



You can archive notifications or mark notifications as read in bulk. To do a mass archive, select the boxes at the beginning of all the notifications that you want to handle. And then from Actions, choose Archive to archive all the selected notifications. Choose Mark as Read to mark as read all the selected notifications. You can also choose Archive All or Mark All as Read to handle all the notifications in the current Inbox.

You can view archived notifications by clicking the small drop-down button beside Notifications. Archived notifications show all notifications that are archived. You can still comment on the notifications. To switch from Archived Notifications back to Active Notifications, click the small drop-down button beside Notifications.



Menu Search

Use the menu search located on the menu bar on page 242 to find all the menu items available to you.

The menu search features “type ahead” behavior so you can easily find items with similar names.

Clicking a menu item launches a view, such as a browse, form, hybrid view (browse and form), or report viewer.

Icons indicate the view types:

 Browse

 Form and Hybrid View (Browse and Form)

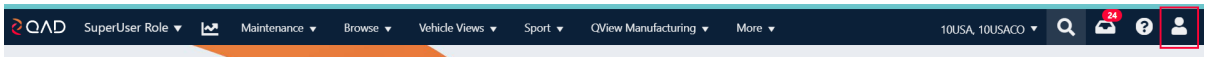
 Report Viewer

 Action Center

You can set a menu item as a favorite directly from the listing of menu items, and see which are already favorites.

User Options

The user options drop-down located on the menu bar on page 242 includes a variety of user settings.



Profile

Choose Profile from the menu bar user options drop-down to view your contact information and preferences, language and locale, and notification preferences.

For more information, see Profile on page 252.

Note that any profile changes will take effect the next time you sign in to QAD.

Display Density

Choose from two display density options:

- Comfortable — density displays with no change. This is the default setting.
- Compact — density displays browses, grids, and form fields at a smaller size, so more data can be displayed at once. It reduces the height of a lot of elements, such as grids, fields, buttons, and so on.

This setting is remembered per user.

Submitted Background Processing

Choose Submitted Background Processing from the user options drop-down to monitor and manage processes that you submitted.

Submitted Approvals

Choose Submitted Approvals from the user options drop-down to view submitted approvals.

Click the User Options icon and choose Submitted Approvals. The system displays a browse of all your submitted approvals. Double-click a record to open the Details page, and you can view all information of the approval, including:

- The date when the approval was submitted
- The type and status of the approval
- Its workspace and description
- Past approvers and next approvers
- Audit log

Future Approvals

Choose Future Approvals from the user options drop-down to view future approvals.

Approval Delegation

Choose Approval Delegation from the user options drop-down to delegate approvals.

Approvers can delegate approvals to other users. Use Approval Delegation to define to whom you want to delegate approvals, under what conditions, and when the delegation is effective.

Click the User Options icon and choose Approval Delegation. Set approval delegations on the Approval Delegation Settings window.

- **Delegation Active** Indicate whether to enable the Approval Delegation functionality for you.
- **Start Date** Specify the start date when the delegation starts to be effective.
- **End Date** Specify the end date when the delegation stops being effective.
- **Approver Type** Specify the approval type for which the delegation is set.
- **Route Code** Specify the route for which the delegation is set.
- **Approver** Specify the approver to whom you want to delegate approvals.

Favorites Setup

Choose Favorites Setup from the user options drop-down to manage your favorite menu items.

Set Current As Home

You can set any page to be your home page by choosing Set Current as Home from the user options drop-down.

Reset Home to Default

Choose Reset Home to Default to set your home page to the system default home page.

Sign Out

Choose Sign Out to exit the QAD Web UI.

Profile

The Profile screen displays a user's contact information, and provides options to select Language, Format Locale, and how the user will receive notifications for alerts.

Contact Information

User ID

The user's identifier.

Email Address

The user's email address.

Mobile App

Click Generate Code to generate a QR code. Scan the code using the camera on your mobile device to connect to the QAD Adaptive ERP Mobile App. See the mobile app online help for more details.

Languages

Language

The user's language. When the user signs in, the user interface will display in this language. From the drop-down, select from the supported languages.

When you open the online help from the Web UI, the online help opens in the user's language set up here. If the online help is not available in the specified language, it opens in English.

Format Locale

The user's locale. When the user signs in, the user interface will display data (such as dates) in a format appropriate for this locale. From the drop-down, select from the supported format locales.

Note that Language and Format Locale changes will take effect the next time the user signs in to QAD.

Notifications

Receive Notifications

Select whether the user will receive notifications.

Category Settings

Category Settings indicate whether notifications for each Notification Category (Activity, Alerts, and Reports) will be sent to the user's QAD Inbox or Email address.

Event Settings

A Notification Category can include various types of events. Event Settings indicate whether notifications for each Event Category will be sent to the user's QAD Inbox or Email address.

Calendar Integration

QAD CRM for Manufacturers allows you to create events that seamlessly integrate with your calendar client. A system administrator must configure both the company's calendar client and Adaptive UX before you can begin using this functionality. See the *Adaptive UX Implementation Guide* in the QAD Document Library for instructions on setting up calendar integration.

Enabled

Select the Enabled checkbox to allow calendar integration.

Calendar Type

Select your calendar type from the drop-down menu.

- Google Calendar. Select this option if your company uses Google as your email and calendar system. Changes to CRM Events are integrated immediately to the calendar and changes to calendar events are integrated immediately to QAD CRM for Manufacturers.
- Microsoft 365 (Outlook). Select this option if your company uses Microsoft 365 as your email and calendar system. Changes to CRM Events are integrated immediately to the calendar and changes to calendar events are integrated immediately to QAD CRM for Manufacturers.
- Other. Select this option if your company uses a client other than Google Calendar or Microsoft 365 as your email and calendar system. This option supports one-direction integration from CRM to the designated calendar. Changes to CRM Events are integrated to the calendar immediately but changes in the calendar cannot be integrated back into QAD CRM for Manufacturers.

Sync Start Date

All events created in the calendar after this date are integrated to QAD CRM for Manufacturers.

Exclude Private Events

When selected, calendar events marked as private do not sync to QAD CRM for Manufacturers.

Favorites

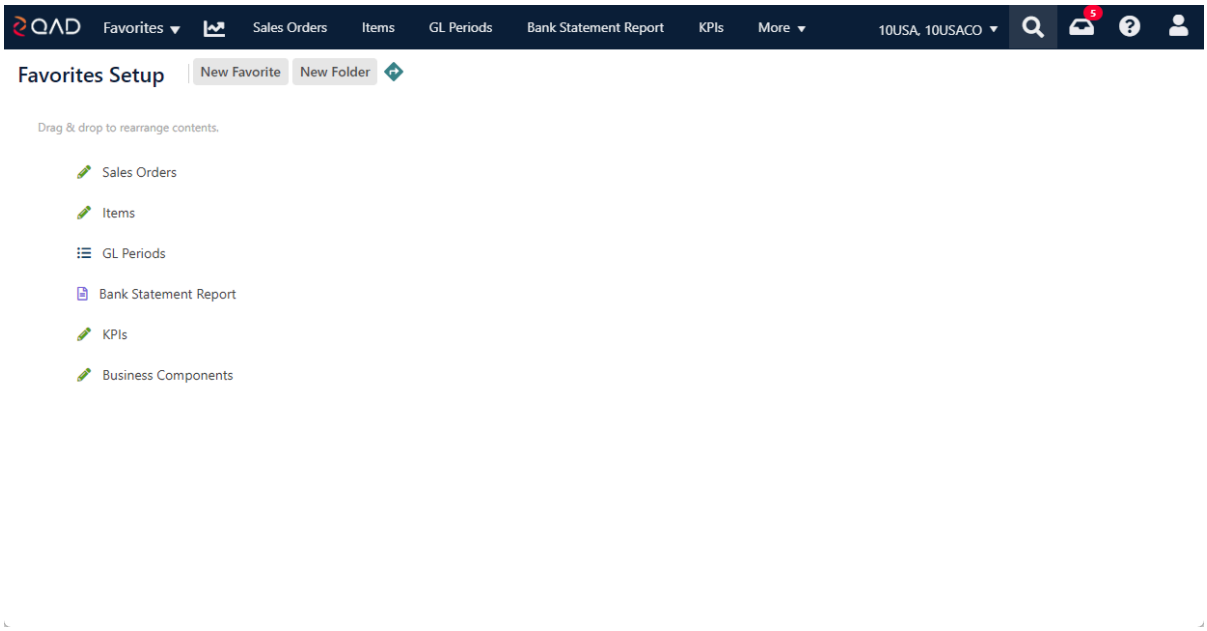
You can save a menu item as a favorite, and easily access your favorites from the menu bar.

On a menu item where you are currently working, you can save it as a favorite by selecting Favorite from the More drop-down.

You can also set a menu item as a favorite from the menu search results by clicking the star icon next to the menu item name.

Managing Favorites

To manage your Favorites, from the user options on page 250, select Favorites Setup to open the Favorites Setup form.



On Favorites Setup, use New Favorite to open the menu search and find menu items you want to set as favorites.

To create folders for your favorites, select New Folder. The folders you create will become the menu bar option drop-downs when you have the menu bar set to Favorites.

Views

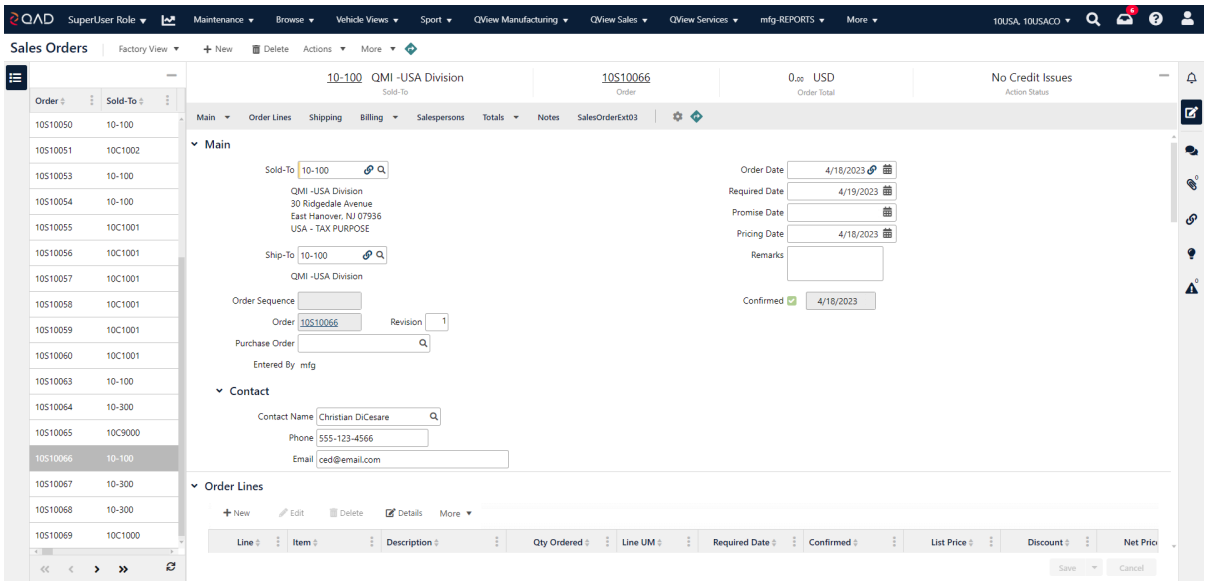
The QAD Web UI offers views in which you can interact with system resources.

Views that combine forms and browses are called hybrid views (or hybrid browses). Many views in the system are hybrid views, while some views consist of standalone browses, standalone forms, or other types of screens, such as the report viewer.

A hybrid view typically includes a toolbar along the top, a browse displaying a list of records (see Browses on page 257), and a form for completing work (see Forms on page 275).

The form organizes everything you need on a single page, typically starting with a summary panel at the top. Related fields and functions are grouped within panels. The navigation bar, located just below the summary panel, allows for quick access to any panel of interest. You can quickly access anything on the form by scrolling or using the navigation bar.

Here is an example of the Sales Orders hybrid view.



Some hybrid views include complete access, combining the grid with the form; other hybrid views only provide read-only access and display the grid. Hybrid views with complete access include the New and Edit buttons along the top.

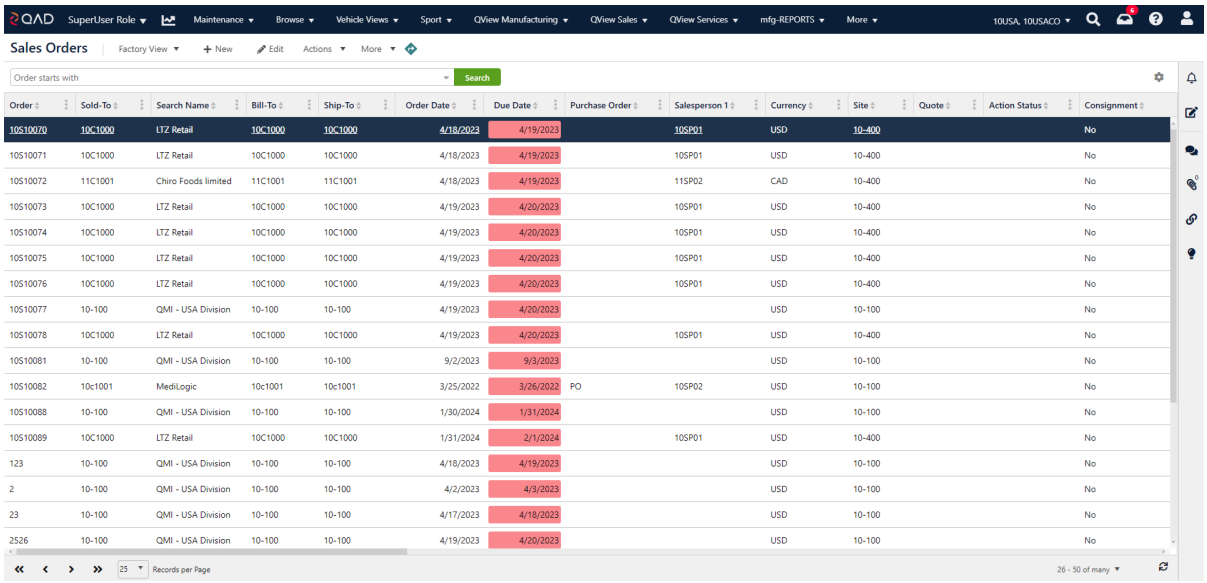
After the view name, the toolbar can include:

- **Stored View Selector** — from the drop-down, choose a stored view. The default is typically Factory View.
- **New** — on hybrid views, select New to open the form and create a new record.

- **Edit** — on hybrid views, select Edit to open the form and edit the current record.
- **Delete** — on hybrid views, select Delete to delete the current record.
- **Actions** — from the Actions drop-down, select actions you can take on one or more records.
- **More** — from the drop-down, you can select:
  - **Favorite** — to save the current view as a favorite.
  - **Share** — to get a link to the current form for sharing with other users.
  - **Design Layout** — to make changes to the form (if you have administrative access).
  - **Permissions** — to make changes to permissions (if you have administrative access).
  - **Show Group By** — to show the Group By feature that can display data hierarchically based on selected columns.
  - **Hide Group By** — to hide the Group By feature.
  - **Export** — to export and download data. Some views also include an **Import** option.
  - **Properties** — to explore the properties of the current screen that you are viewing.

Browses

When you first open a view, the browse grid displays records in rows.



Browses include the following features:

- Quickly search for data or add search conditions
- Sorting on columns
- Paging controls along the bottom of the grid display
- Hide and show columns with configuration settings
- Adjust sort order and priority of columns
- Save column configurations as part of a stored view
- Save search filters as stored searches

Browses that are part of a hybrid view include New and Edit buttons, so that you can create new records and edit existing records in the form. For these, you can also edit an existing record by double-clicking a row. Double-clicking a row opens the form for editing the selected record.

Searching and Filtering

You can filter the display using the Search function. In the Search box, enter the field you are searching on, along with an *operator* (such as “starts with” or “greater than”), and then a string or value. This describes a *search condition*.

For doing a quick search, enter a string to search based on the default condition for the grid. The default field is typically a key field for the grid.

You can use an asterisk (*) in browse search to broaden your search results. It indicates that you want to search for any number of characters. For example, to search for all items beginning with “abc”, select "equals",

and then type “abc*” in the search box. This functionality is available for the "equals" and "contains" operators in browse search, including the quick search and advanced search. It is also supported in any position of the string (in the beginning, middle, and end of a string) and is not limited to one wildcard: abc*, *abc, *abc*, a*b*, and so on.

You can enter as many search conditions as you need to display the results you want. Click the drop-down icon in the Search box to add, change, or remove search conditions.

The operators for defining search conditions include:

- Equals
- Does not equal
- Contains
- Range
- Starts with
- Greater than
- Less than
- Greater or equal to
- Less or equal to
- In list (only in fields with available lookup)
- Not in list (only in fields with available lookup)
- Ends with
- Is null
- Is not null

Use the “+” and “x” icons to add or remove conditions. To clear the values of the existing search conditions, click **Clear all**. With the March 2023 release, the Clear all option only deletes the condition line values, not the lines themselves.

For details on the advanced search, see Advanced Search on page 261.

Paging Controls

At the bottom of a grid, you have controls for paging through grid records and controlling the number of records per page.

The controls include:

- Buttons for Next and Previous pages.
- Buttons for First and Last pages.

- A drop-down to select number of **Records per Page** (1, 10, 25, 50, 100, 500, 1000, 2000, 5000, and 10000), with the default set at 25 records per page.
- Indication of number of records currently displayed so that you know the total number of records and your location within those records.
- Refresh button, so you can refresh the display to reflect the most recent data.

Important The 5000 and 10000 “records per page” settings should be used judiciously. Browsing with 10000 records may be slow, and the anticipated load time is up to 20 seconds.

Your administrator can disable the global browse record count and instead enable the browse record count only for specified browses. When the global browse record count is disabled, the total record count number in the bottom-right corner of the display is replaced by “many”, such as “1-200 of many”. By clicking “many”, you can select a “Count Records” option, which starts a record count. During the count, the display changes to “1-200 of [counting...]”. Once the count is complete, the record count number is then displayed, such as “1-200 of 23,510”.

Browse performance should be considered for “visible browse columns” and “records per page” settings.

- For optimal performance, we suggest to limit “visible browse columns” to 20 (see **Configuring Columns** below).
- Larger “records per page” settings could impact browse performance depending on the number of rows returned for the screen.

Configuring Columns

You can configure which columns display using the Configure Columns dialog. Open the dialog by clicking the gear icon in the upper-right corner above the browse grid, and select which columns you want the grid to display. For optimal performance, select 20 or fewer columns. You can select more columns, but this may impact performance. To select all columns, select the checkbox next to the Configure Columns label.

You can save the column configuration changes as part of a stored view.

Column Display Options

Each browse column header includes display options that you can choose from the vertical ellipsis icon. These options include **Sort Ascending**, **Sort Descending**, **Freeze**, and **Group By Column**.

You can sort columns in ascending or descending order by selecting Sort Ascending or Sort Descending. You can apply sorting to multiple columns at the same time; the sort priority of each is indicated by a small number next to the column title, with an upwards indicator for ascending order and a downwards indicator for descending order. You can toggle the indicators to adjust the sort priority and order of the columns.

With Group By Column, you can display data hierarchically based on selected columns. To group by a column, all records must display on one page. To do this, increase the records per page or apply filters that get fewer records on one page.

Note that applying Group By Column to a browse with many columns and more than 1,000 records can take a few seconds to complete. Browse performance should be considered for “visible browse columns” and “records per page” settings.

- For optimal performance, we suggest to limit “visible browse columns” to 20 (see **Configuring Columns** above).
- Larger “records per page” settings could impact browse performance depending on the number of rows returned for the screen.

For our high-volume customers, an optional server-side grouping feature can be enabled on request. This feature is helpful for browsing large datasets. It removes the prior limit on BC browses that required all records to load for grouping. However, to maintain system stability and speed, certain performance-risking features like “Expand Groups” are disabled when this feature is active. Note that an environment restart is required after enabling the server-side grouping feature.

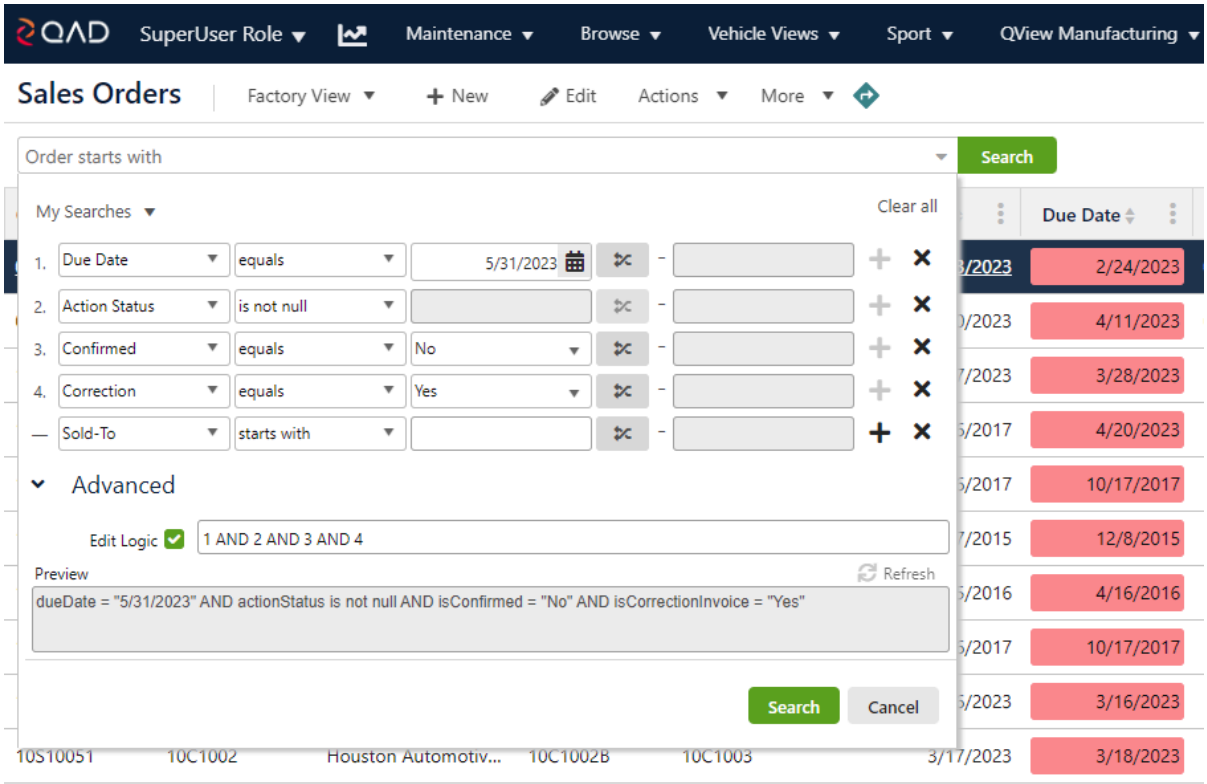
Export and Import

You can export and import browse data. See Exporting and Importing Data on page 266 for more information.

Advanced Search

It is possible to save the edited advanced search logic as stored views (View drop-down menu) and stored searches (My Searches drop-down menu). Stored views and stored searches allow you to save incomplete criteria lines as templates for searching.

If the search condition line is complete, the line has a number. Condition lines that are not complete have an em dash (—) instead of a number.



Advanced

Edit Logic

By default, the field is disabled and displays the standard browse search logic. Select the Edit Logic checkbox to change the logic statement in the text field. When you select the Edit Logic checkbox, all incomplete condition lines are deleted and only the last “+” icon for adding conditions is available.

In the Edit Logic text field, you can combine conditions with AND/OR terms and use the parentheses ( ) to group conditions for query execution. When you edit the logic statement, check the following:

- If you add new conditions, manually add the new condition numbers to the logic statement.

- If you delete conditions, manually remove the condition numbers from the logic statement.

The logic statement is validated for correct syntax. Also, all completed conditions must be included in the logic statement. Validation is done when you click the Refresh button in Preview or the Search button. If the condition line is not included in the logic statement, the line number is highlighted in orange. Also, syntax errors are returned if any.

To reset the search to the standard search logic, clear the Edit Logic checkbox.

Preview

Shows a preview of the browse search that is executed when you click the Search button for both standard browse searches and edit logic searches.

The preview is refreshed automatically for the standard browse search. If you edit the logic statement, you need to click Refresh to update the preview. The warning icon appears next to the preview if it needs to be updated.

Refresh

Click Refresh to update the preview if changes are made to the search conditions or the logic statement.

Search

Click Search to execute the advanced search logic and filter the browse.

You can view the full statement in the search field to check how the AND/OR terms as well as the ( ) are arranged. You can click the “x” icon to remove the search from the quick search bar, but you need to open the advanced search panel if you want to adjust the search conditions or logic.

Predefined Criteria

This panel is displayed if you select the Show in Advanced Search checkbox under Browses. For more details, see Business Components > Browses > Predefined Search Criteria.

Resource Properties

With the Properties screen, you can explore the properties of the current screen that you are viewing. You can view the properties for reports, hybrid browses, browse-only views (BC-based and legacy), and form-only views.

To access the properties, click More on the toolbar, and then click Properties. Use the Properties screen to quickly identify the underlying business component, Resource URI, App URI, Browse Label, Relationships, and more.

All users can access the Properties screen. It is read-only. Properties vary depending on the resource type (see the table below).

Panel	**Applies To**				
	Report	**Browse Only: BC-Based**	**Browse Only: Legacy**	**Form Only**	**Hybrid Browse**
Resources Fields: • Resource Label • Resource URI • Resource Type: Browse Only, Form Only, Hybrid Browse, and Report. • Eligible for Menu • App • App URI • Namespace • Secure URI	Yes	Yes	Yes	Yes	Yes
Browse Fields: • Browse Label • Browse URI • Business Component Browse • App	No	Yes	Yes	No	Yes

Panel	Applies To				
	Report	Browse Only: BC-Based	Browse Only: Legacy	Form Only	Hybrid Browse
• App URI • Namespace					
Business Component Fields: • Business Component • Business Component Label • Business Component URI • Not Extensible • App • App URI • Namespace • Business Document (fields are hidden if the checkbox is clear): Business Document Name, Business Document Label, Business Document URI, App, App URI, and Namespace.	No	Yes	No	Yes	Yes
Relationships Displays information from the Relationships panel in Business Components.					
Field Property System Overrides Displays information from the System Property Overrides panel in Business Components > Fields.					

Panel	Applies To				
	Report	Browse Only: BC-Based	Browse Only: Legacy	Form Only	Hybrid Browse
Field Property Domain Overrides Displays information from the Domain Property Overrides panel in Business Components > Fields.					

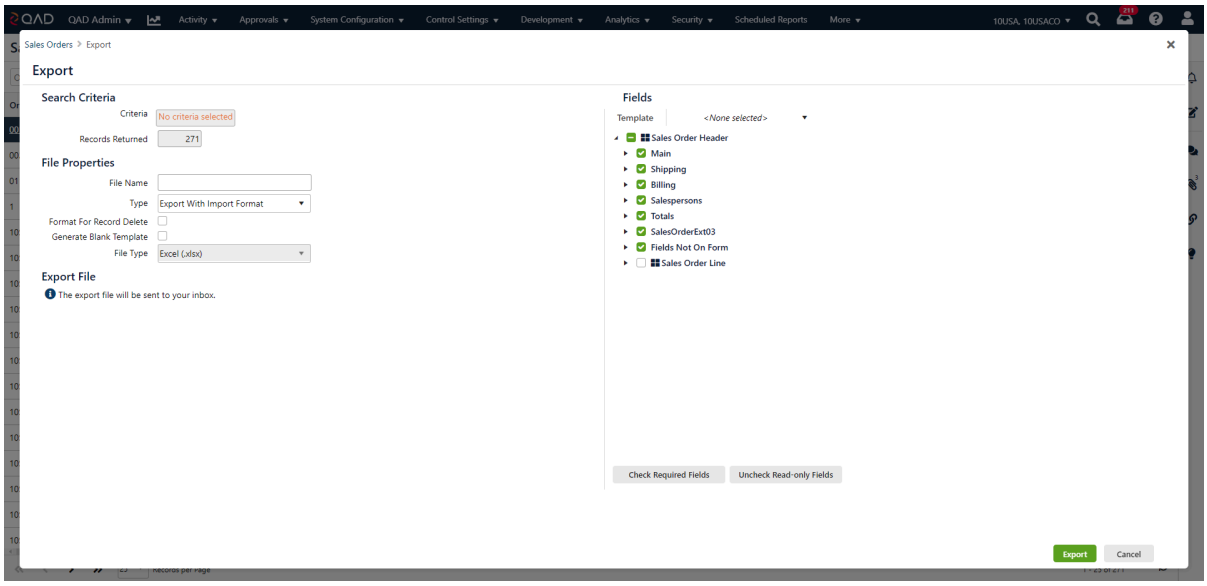
Exporting and Importing Data

You can export data from the system, and for selected screens, you can also import data.

For a domain or entity scoped business component or business document, you can only import data of the current domain or entity. If you leave the domain/entity value blank, the system will use the current domain/entity as the default value. If the domain/entity value in data is different from the current domain/entity, the data cannot be imported.

Exporting Data

To export data, from the More drop-down menu, choose Export. The Export screen opens, where you can export the data.



Criteria

Shows any browse search criteria that you have applied (disabled).

Records Returned

Shows the number of records returned according to the search criteria (disabled).

File Name

Enter the export file name.

Type

Select one of the following export types:

- Export Browse Columns — Export the data in the current browse.
- Export with Import Format — Select this option if you intend to edit existing records and import the changes, or add/delete records while importing.

Include in Export

(Only for Export Browse Columns) Export the current stored view columns (disabled).

Format for Record Delete

(Only for Export with Import Format) Select to export data and specify the records to delete upon import of the spreadsheet. This will add an additional column to the spreadsheet so you can specify which records are to be deleted.

Generate Blank Template

(Only for Export with Import Format) Generate a resulting Excel import template without any data populated.

File Type

For Export with Import Format, the file type is Excel (.xlsx).

For Export Browse Columns, select the file type. You can export browse data to a variety file formats, including:

- Excel (.xlsx)
- Comma-separated values (.csv)
- Tab-separated values (.tsv)
- Tab-delimited values (.txt)
- Portable Document Format (PDF) (.pdf)

Export File

When you click Export, the system will start to export data to the file, running in the background. You will receive a notification in the Inbox when the export is completed, and you can then click the link in the notification to download the file. The export process may take some time when there is a large amount of data. Please do not sign out when the export is running; instead, just close the web browser page if necessary. If you do sign out of the Web UI, the export will fail and you will not receive the notification.

Fields

(Only for Export with Import Format)

On the Fields panel, you can specify the fields to include in the import template. If the business component is enabled as a business document, you can select fields from both parent and child business components in the export configuration.

A child table in a business document is added to the import template only if the corresponding external grid is present in the design layout of the parent business component.

Some fields exist in multiple places on the layout, but they will be exported once if selected. You can see an information icon next to such fields.

Key Fields

All key fields for BCs and grids are included in the selection tree, no matter if they are on the form or not.

- If any business component/grid field is selected to be included in the import template, then the related key fields are selected and disabled (you cannot exclude them from the template).
- If all business component/grid fields are not selected, then you can exclude the entire business component/grid from the import template (you can exclude the entire grid and entire child business component).

Excluded Fields

Fields with "Hidden from UI" = Yes and fields that are not on the active layout of the user's current workspace are excluded from the import configuration tree and the resulting Excel template.

Initial Field Selection

Initially, only the parent business component in the tree list is selected. All child business components are not selected. You need to manually add any child business component fields needed for the export.

Legend



If a field has field security enabled and you do not have the read access, then the fields are displayed in the tree list, but are excluded from the generated Excel import template. If access to an entire field group is denied and all those fields are in the same panel, then the fields are not displayed and the panel shows the security icon.

Field labels have standard QAD Adaptive text.

Check Required Fields

Select all required fields in the business component or business document.

Uncheck Read-only Fields

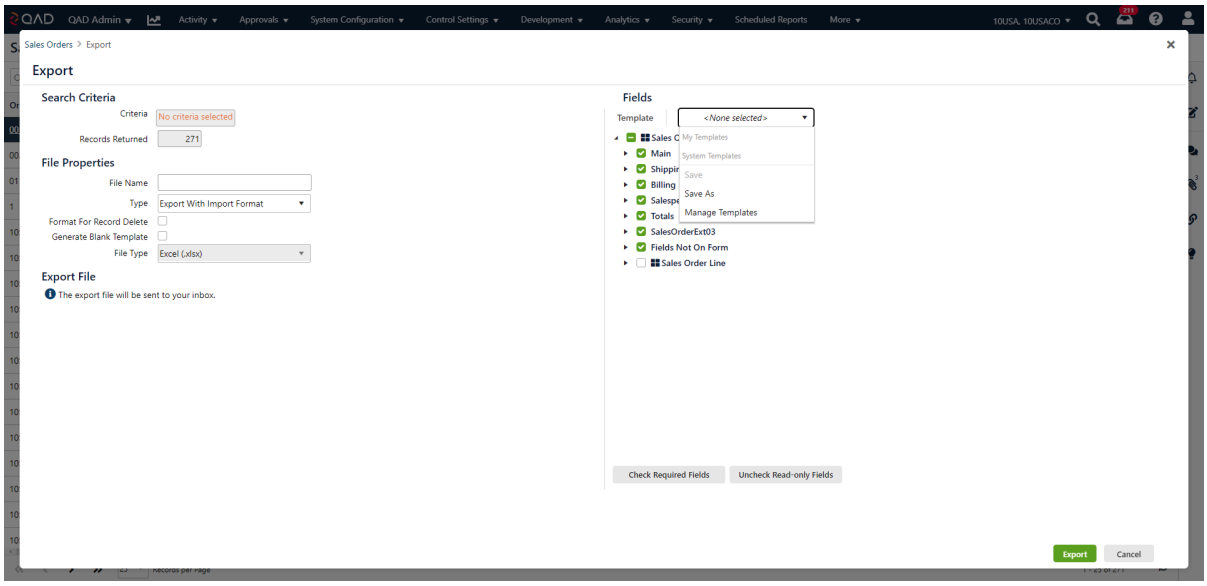
Unselect all read-only fields in the business component or business document.

Template

(Only for Export with Import Format)

Use the Template menu to create new import templates and update the existing import templates. From the Template menu, you can view system-wide and your personal import templates. Inactive import templates are hidden from the Template menu.

Excel import templates are saved as configuration data, and you can move them across environments.



Save

Update the selected import template with the new field configuration.

Save As

Open the Import Templates screen where you can create a new import template. See Import Templates on page 6974.

Manage Templates

Open the Import Templates screen where you can update existing import templates. See Import Templates on page 6974.

Exporting Data for Business Documents Based on Business Component Structure

Export configuration can be based on the active layout or the business component structure. This option is configured in BrowseView of ViewResourceMetadata for a business component that is enabled for business document.

For some business documents, the active layout does not match the business document definition, so the Excel import template does not include all needed fields. You can use an alternate option to access all fields from a business document to cover these special business document cases. This alternate template is constructed from the business component structure definition instead of the active layout, and you can select all fields for the import template.

When the alternate export configuration is used, you can view the following information message at the bottom of the Export screen: "Fields are organized by the business component structure, not the form layout."

The import template structure uses the business component field groups to organize the fields.

- If a field belongs to a field group, then the field is included in the defined field group. Note that field groups also enable field security for any field in a field group.
- User Defined Fields — If a field is User Defined and is not included in a field group, then the field is included in the User Defined Fields group.
- Default — All fields without a field group are included in the Default group.
- If a field is excluded from the template or is Hidden From UI, then the field is excluded from the template structure.

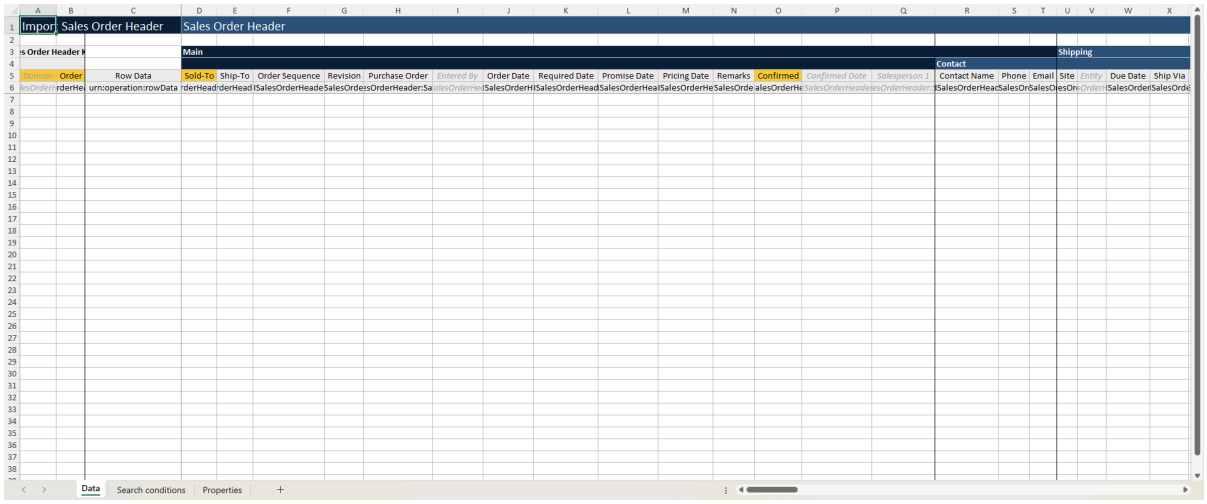
Internal grids and embedded business components are included in the template structure automatically. Internal grids appear in the template as another level in the structure and are denoted as a grid. Embedded business components with a One-to-one relationship are included in the template structure according to field groups. Embedded business components with a Many-to-one relationship appear as internal grids.

In the Excel import template, field groups are displayed as panels and can be ordered.

Excel Import Template

To import data, export the data to Excel or generate a blank template, make changes, and then re-import data from Excel. When you format the data for later import, the data is exported to Excel in a format that is based on the underlying (Business Component or Business Document) data structure, which can be different than the browse column order. In Excel, the order of the columns might not match the order of the columns in the browse grid you are currently working with in the QAD Web UI.

You can only import data that is in a proper Excel file format based on the underlying Business Component (or Business Document) data structure. You can then remove the data rows and enter new data rows, following the format in the Excel file. See the Entering Data section below.



Frozen Columns

The following columns are always frozen so you never lose the context of the data you are working with:

- Delete Record Column
- Parent Business Component Key Field Columns
- Child Business Component Key Field Columns
- Row Data Column

Frozen Rows

The following rows are always frozen:

- Row 1: Import Title
- Row 2: Blank row for you to leave ad-hoc notes in the template
- Row 3: Panel names and merged fields by panel for field context and easy navigation
- Row 4: Grid names and merged fields for grid context
- Row 5: Translated field labels (uses the language of a user who executed the export)
- Row 6: Field URI

Field Order

Fields are ordered first by business component, then by form panel, in the order they appear on the design layout. Business document child business components are also organized according to their own form layout in their appropriate section of the template.

Row Data Column Codification

The Row Data helps you understand what data the row represents. This code is calculated as part of the template so when you add a new row to the sheet, the code is correctly defaulted depending on

whether values exist in the various element key fields. If a value exists in multiple element key fields, then the row field indicates an error. The row code shows the element label.

Legend

- Required fields
- The Yes mark in the Delete Record column
- Read-only fields are displayed in italics with gray font color
- To help with navigation and data orientation, business component and panel labels alternate colors
- To differentiate fields from grid columns, grid labels alternate colors

To differentiate fields from one panel to another, a standard black vertical column border separates the panels and/or grid levels.

Business documents typically contain multiple related business components. To differentiate fields from one business component to another, a bold standard black column border is displayed between the panels of one business component and another.

Entering Data

You can further customize the spreadsheet by deleting some columns, as long as it is not a key field.

You need to enter data in corresponding rows specified by the Row Data value.

Field property validation is available in the Excel import template, such as data type, field length, and maximum/minimum values.

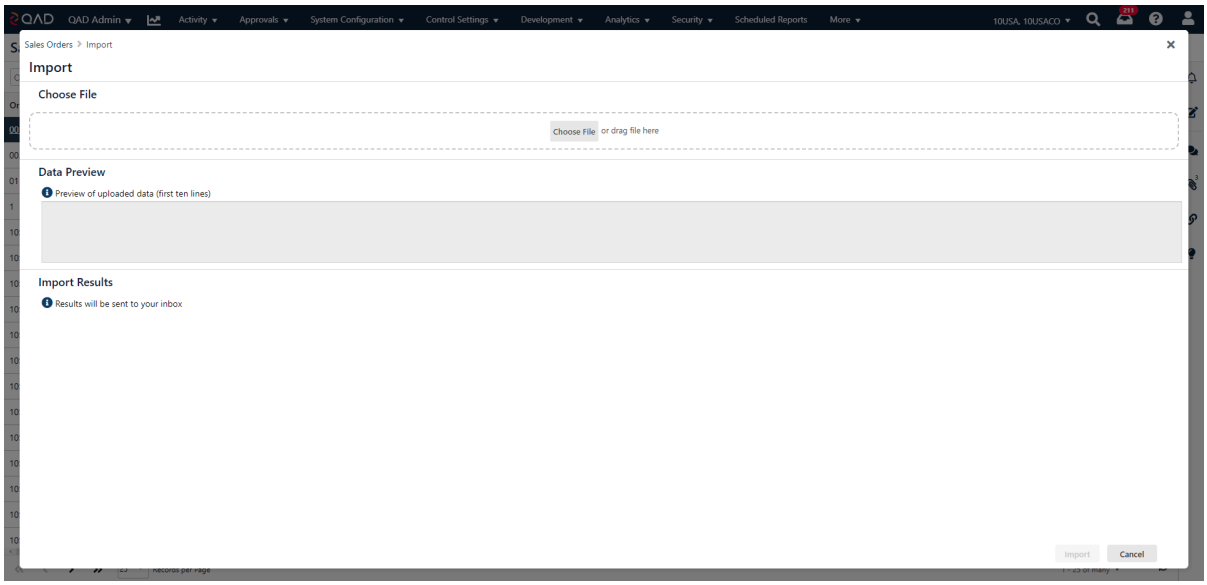
If you leave a cell blank, the value of the field will not be updated.

Use curly brackets {} to update a field value with blank, this is only needed for character fields with default value logic; create and update.

Importing Data

To import data, from the More drop-down menu, choose Import. The Import screen opens, where you can import the data.

Note that importing from Microsoft OneDrive is not supported.



Choose File

On the Choose File panel, choose a file or drag a file to the screen.

Data Preview

To help make sure you have selected the file you want, the Data Preview panel shows the first ten lines of the file.

Import Results

When you click Import, the system will start to import the data into the system, running in the background. You will receive a notification in the Inbox when the import is completed, and you can then click the link in the notification to open the background processing details of the import. If the import failed, you can check the errors using Background Processing on page 7029, or go to attachments and download the results file.

The import may take some time when there is a large amount of data. Please do not sign out when the import is running; instead, just close the web browser page. If you do sign out, the import will fail and you will not receive the results notification.

Excel Import Results File

The system provides the import results in an Excel file. The file includes four sheets: Log, Error, Failed, and Properties.

You can identify fields with error or warning/information messages by the following cell highlighting:

-  Error
-  Warning
-  Information message

You can use the Excel import results file to import the failed records. To do this, fix the errors in the Failed sheet, delete the Errors column, and then move the sheet to first in the file.

Log

The Log sheet provides the import status for each row (error or success).

Error

For failed rows, the Error sheet provides further information about each failed row.

Failed

The Failed sheet provides the row data so that you can correct it and re-import.

If the row contains errors, the Errors column shows Error (#), where # is the count of errors and warnings for the row. Hover over the cell to view a comment listing the errors and warnings for the row. Information messages are also displayed in a comment.

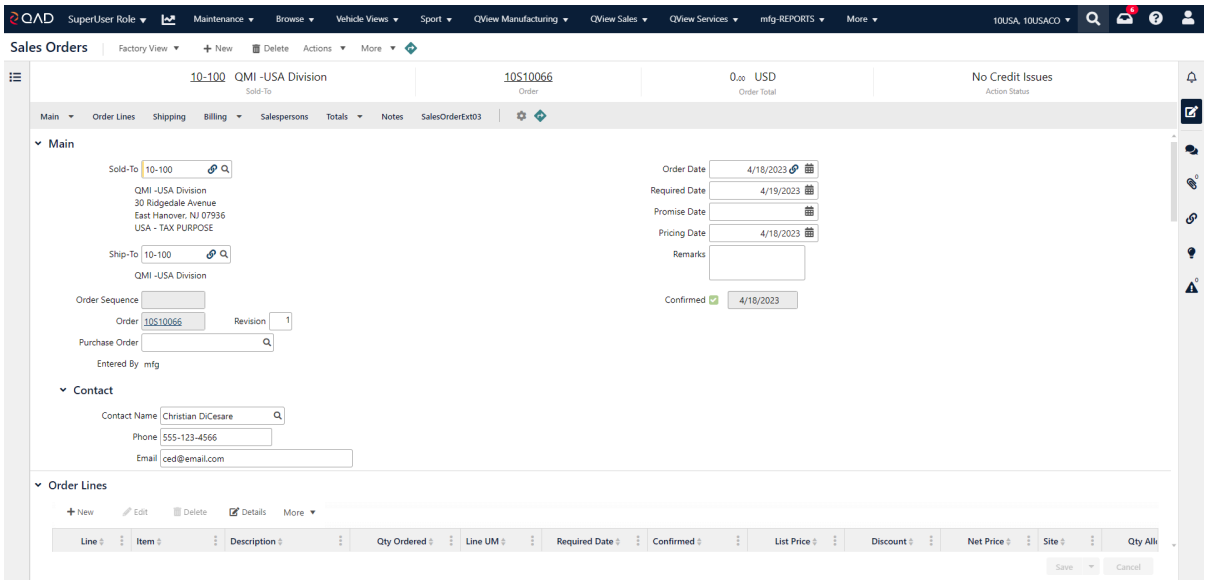
Properties

The Properties sheet includes summary information about the report, including the user ID, time, time zone, and import file name.

Forms

In a form, you create, view, edit, and delete record data. The form displays when you double-click on a row (a record) in an editable view grid. A form includes a summary panel, navigation bar, main panel, and then various detail panels and sub-panels.

Here is an example of a Sales Orders form.



Typically, forms are part of a hybrid view that also includes a browse (see Views on page 255).

Summary Panel

The summary panel displays key data about the form so that you know what you're working on at a glance. The summary panel is located between the menu bar and the navigation bar. These fields are read-only and stay in view even when you scroll down the page.

Navigation Bar

Just below the summary panel, you have a navigation bar. The navigation bar summarizes the various panels on the form, with drop-downs for the sub-panels. Use the navigation bar to jump quickly to any panel on the form. This is particularly useful when you are working with a long form and want to quickly go to a particular panel.

Configuring Panels

The gear icon on the navigation bar opens a Configure Panels dialog where you can control the display of panels and fields on a form.

Select a checkbox to display a panel or field; clear a checkbox to hide a panel or field.

Fields that do not have labels are indicated by their identifiers within square brackets.

You can use this feature to simplify the form as needed so that you just see the panels and fields you need for a particular task.

You can save these configuration settings as part of a stored view.

Note

In the Configure Panels dialog, panels and fields hidden by business logic are indicated by the "eye-slash" icon. Panels or fields hidden by business logic will not display on the form, even if the display checkbox is selected. Conversely, any panels and fields hidden because the display checkbox was cleared will not display even if the business logic requests that they be displayed.

Main and Detail Panels

The main panel is the first panel located below the navigation bar. The main panel includes the most pertinent fields and controls, while the detail panels (and sub-panels) bring together various supporting fields and controls.

Each panel is represented on the navigation bar, with sub-panels included as drop-downs on the menu bar. You can quickly navigate to any panel or sub-panel from the navigation bar. You can also simply scroll up and down the form to access anything on the form. Everything you need is directly available on the page.

Form Grids

A form grid is a grid located in a form’s panel that lists data you can edit. The QAD Web UI includes the following types of form grids:

- Single-row edit grid — only one row at a time can be edited.
- Multi-row edit grid — multiple rows can be edited at a time.
- Hierarchical grid — for hierarchical, parent-child data relationships.

If the amount of data exceeds the system's capacity, you can see an error message above the grid indicating that it encountered an error loading its data.

Single-Row Edit Grids

With single-row edit grids, you can only edit one row at a time. For example, you can only make changes to one sales order line at a time. In the Order Lines panel on Sales Order, changes to any one particular line must be saved before you can make changes to another line, so the Order Lines panel includes a single-row edit grid.

Under the line you are currently editing, three buttons display: Next Line, Done, and Cancel.

- **Next Line / New Line** — saves the current grid row and moves your focus to the next row. If you are on the last row, a new row is started for you.
- **Done** — indicates you are done with making changes. Clicking Done saves the current row and moves your focus to the next field on the form.
- **Cancel** — discards your changes.

Single-row edit grids make quick entry easy, but on occasion you might need to see additional fields. Next to the New, Edit, and Delete buttons, you have the Details option.

Clicking **Details** opens a form as a modal dialog. In the details form, a breadcrumb in the upper-left corner provides context about your path. When you are done, at the bottom of the form, click Save and then Close to close the dialog.

Multi-Row Edit Grids

With multi-row edit grids, you can edit any field in the grid, and then the changes are saved when you click the Save button located at the bottom of the form page.

Multi-row edit grids are for situations where the individual rows do not need to be saved independently. Multi-row edit grids do not include the Next Line, Done, and Cancel buttons that are included on single-row edit grids. In multi-row edit grids, the changes can be made directly, but the changes are not saved until you click the form's Save button.

Hierarchical Grids

Some form grids can display data hierarchically, showing the parent-child relationships of the data.

Hierarchical grids whose rows include child data include a toggle for opening and closing the child grids.

On hierarchical grids, the Group By feature is available for the top level of the hierarchy only.

Group By

With the Group By feature, you can display data hierarchically based on selected columns.

To use Group By on a form grid, from the More drop-down, select Show Group By. With Group By active, you can drag-and-drop column headers to group the display by the selected columns. You can then expand or collapse the groupings as needed. When done, from the More drop-down, select Hide Group By.

For hierarchical grids, the Group By feature is available only for the top level of the hierarchy.

Export to Excel

With the Export to Excel feature, you can export the contents of a form grid to Excel and analyze data offline. This feature is available from the More drop-down. Only the visible columns are exported, along with treelist index fields. The maximum export limit is configurable and defaults to 50000. The import is not supported from the grid.

For hierarchical grids, you can export only the first level of the grid.

Form Grid Column Controls

Each column includes controls to do the following:

- Sort Ascending
- Sort Descending
- Summary: None, Sum, Average, Minimum, Maximum, Count
- Filter: Equals, Does Not Equal, Greater Or Equal To, Greater Than, Less Or Equal To, Less Than, Is Null, Is Not Null, Range, In List, Not In List

Form Footer Controls

The form footer is the area along the bottom of the form.

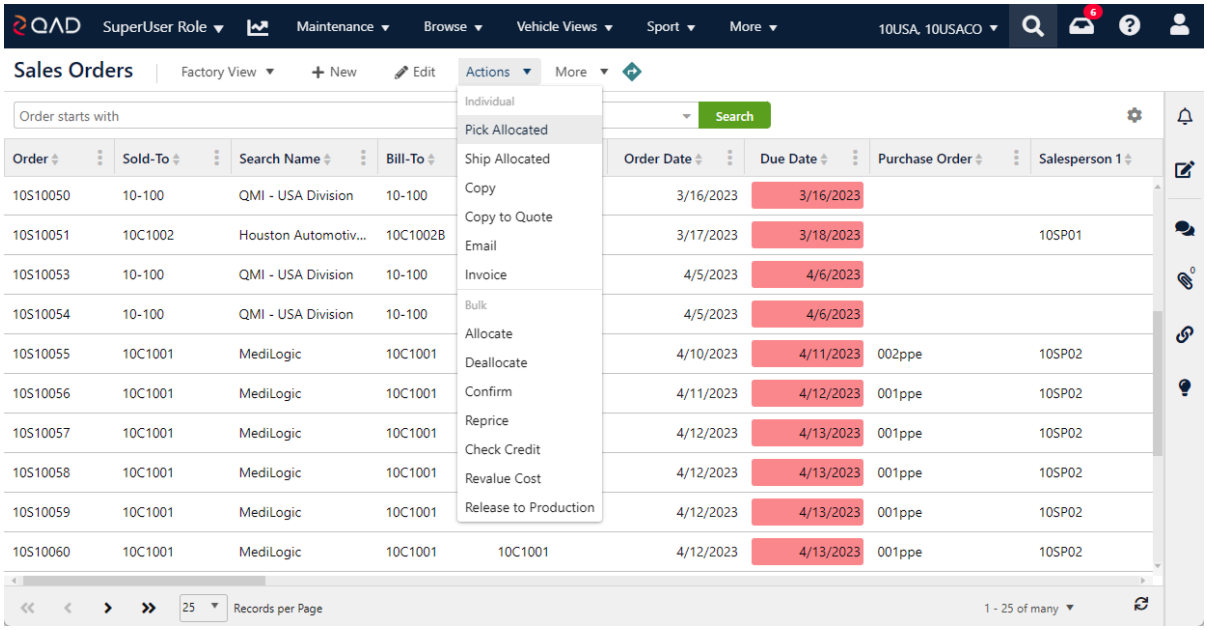
This area includes buttons for saving and canceling changes you have made on the form:

- **Save** — the Save button becomes active when you have made changes that need saving.
- **Save & Next** — for efficient data entry, you can select Save & Next instead of Save. Clicking Save & Next saves the current form and then moves to the next grid (browse) record so you can continue your work without having to explicitly open the next record. If you are on the last record, clicking Save & Next will start a new record for you.

Actions

On some browses and hybrid views, the Actions drop-down can offer actions you can take for a record or set of records. Individual actions are for individual records and bulk actions are for sets of records that are selected based on browse search criteria and automatically applied required criteria. Bulk actions are only available for a hybrid view in browse-only view mode. For bulk actions, once you select the records and set options, you can simulate or submit the action.

Here is an example of actions available from Sales Orders.



1. For bulk actions, make sure the hybrid view is in browse-only view mode.
2. Use the Search box to build criteria for filtering.
3. From the Actions drop-down, choose a bulk action.
4. In the modal dialog, set any options in the Options panel and click Continue.

   The system displays all the records (or lines) that meet the criteria.
5. Select the specific records (or lines) to which you want to apply the action.

   Note

   The displayed records (or lines) not only meet the search criteria, but also meet the required criteria for the action.
6. If you want to do simulation before confirming the action, click the Simulate button.

   The system displays the simulation results.

- If the simulation results show a success, all the selected records (or lines) are confirmed for the action.
- When conflicts exist, the system displays an error message.

7. To submit the action, click the Submit button.

The action is now applied to the selected records (or lines).

Stored Views

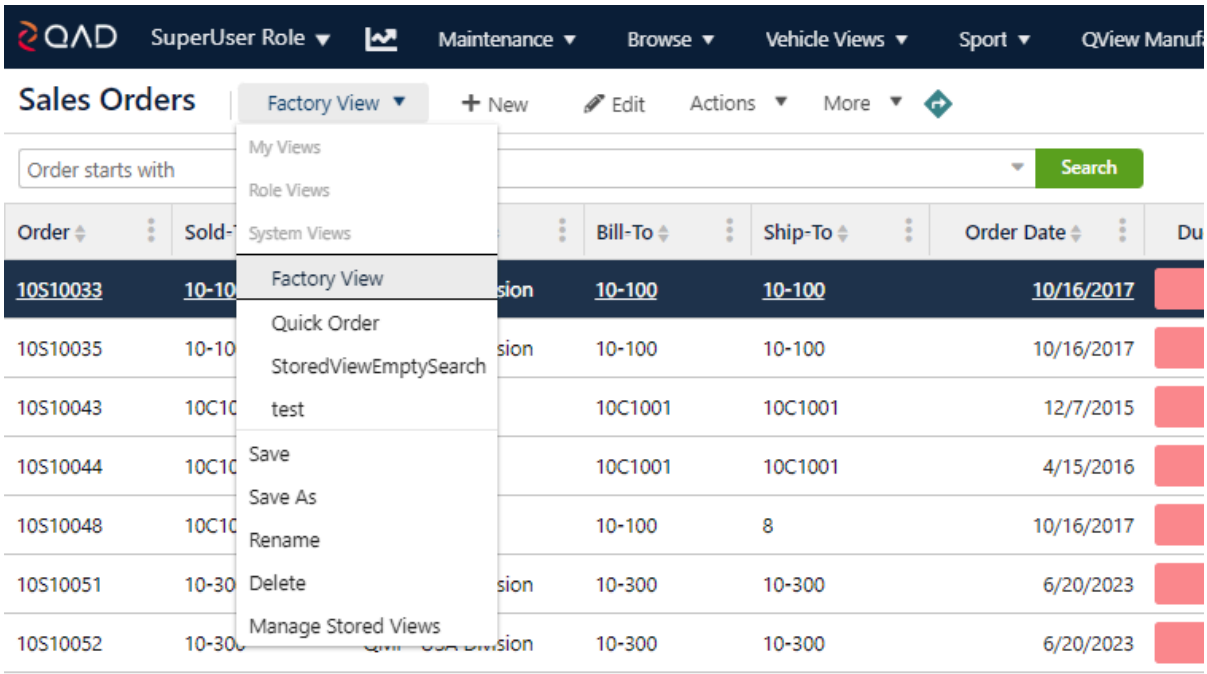
You can modify a view and then save it for later use as a stored view.

With the September 2021 release, the Default View is renamed to No Stored View or Factory View:

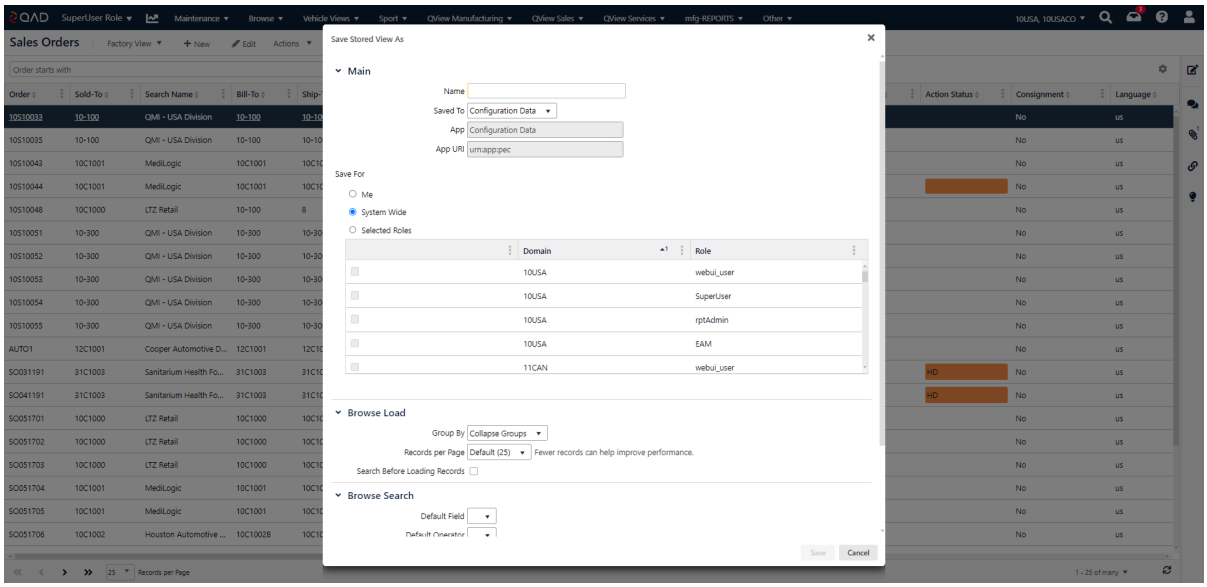
- For existing screens where the Default View has no stored view changes to it, No Stored View is displayed.
- For existing screens where the Default View has stored view changes to it, it is renamed to Factory View.

No Stored View appears for screens with no stored views or no active stored views.

The views drop-down displays the name of the currently selected stored view. From this drop-down, you can select other stored views.



By selecting Save As, you can save a new stored view and select options for it.



Aspects of a view you can modify include the grid columns, grid search conditions, form panels, and form grid filters. After configuring a view, you can save it as a stored view from the drop-down located next to the view title. Once saved, a stored view is available from that drop-down.

Some views include predefined views for your convenience. For example, Sales Orders includes a "Quick Order" view that includes only the panels that contain required fields, while the "Factory View" includes all of the panels. Security settings can limit what a given role can see and update, tailoring access for that role.

Note that on QAD-provided screens, the "Factory View" can only be changed by QAD. However, the "Factory View" of screens created by a QAD customer can be modified by the QAD customer's users.

When you access the system, the last-used stored view will be displayed. However, when you first access the system, the stored view set as default is displayed.

If you customize a view (such as the sort, search, columns, and panels), the customized view will be used only for the single use unless the view is saved as a stored view.

Main

Name

Enter the name for this stored view.

Saved To

Choose to save your data to the App Data or Configuration Data:

- App Data. The data is saved to the user's Active App set in My Developer Settings and displays the active app name and the corresponding App URI.

- Configuration Data. The data is saved to Configuration Data and displays Configuration Data as the app and the corresponding Configuration Data URI.

See Configuration Data Overview on page 7153 for detailed information.

App

Indicates the app for the stored view.

App URI

Indicates the uniform resource identifier (URI) for the app.

Save For

Choose to save your stored view as available just to you, shared with selected roles, or available to everyone on the system.

Browse Load

The Browse Load settings include:

Group By

From the drop-down, choose Collapse Groups to have any groupings collapsed by default. Choose Expand Groups to have any groupings expanded by default.

Records per page

Specify the records per page that will be loaded for the browse (default is 25). You can increase the number, but fewer records can help improve performance.

Search Before Loading Records

Select to have the browse search panel open automatically so that you can quickly enter search criteria. The browse records will not appear automatically until you perform a search. You will need to click search to see records.

In the case of browses with many records to which you will want to apply your own search criteria, you might want to select this option so that no records are loaded initially. Otherwise, the search is not opened and the records are loaded automatically.

Browse Search

You can specify search settings and include those settings as part of a stored view. The search criteria are then applied to the browse automatically for the stored view.

Default Field

Select the field to include in the stored view's browse search.

Default Operator

Select the default search operator to include in the stored view's browse search.

Include Search Criteria

Select to include the current search criteria on the browse as the default search criteria for the stored view.

Note that including search criteria only applies when the browse is accessed directly from the menu. Including search criteria does not apply when the browse is accessed from lookups, drilldowns, the Inbox, or a custom button on a form.

Mobile App

Display in App

Select to have the stored view be available in the QAD Mobile App.

Number of Browse Fields Displayed

From the drop-down, select the number of browse fields (that is, columns) to be displayed in the context of the QAD Mobile App. You can set this to a number between 1 and 6 (default is 4).

Manage Stored Views

You can set the initial default stored view and hide QAD-provided views by using the Manage Stored Views option under the views drop-down.

Note that you need to have permissions to access the Manage Stored Views option and see the grids.

System Views

In the System Views grid, you can activate and deactivate system stored views. When not active, a stored view does not display in the views drop-down, but always displays in the grid. It is possible to deactivate all views.

You can also choose a default stored view that is selected the very first time a user opens a screen. This allows you to set the initial stored view for your users rather than use a QAD-provided view. This setting is only used when the user opens a screen for the very first time, each time after that the last-used stored view is selected.

Once the first system stored view is created, it automatically becomes the default view. The default stored view is always active.

Role Views

In the Role Views grid, you can activate and deactivate role stored views. When not active, a stored view does not display in the views drop-down, but always displays in the grid. It is possible to deactivate all views.

My Views

In the My Views grid, you can activate and deactivate user's stored views. When not active, a stored view does not display in the views drop-down, but always displays in the grid. It is possible to deactivate all views.

Activity

With the Activity panel, you can see a history of transactions and follow the things you care about most. You can comment on activity in a way similar to how you use social media. The system can track field changes, comments, and attachment uploads for the current record (such as a field) and displays them in the Activity panel.

From a form, open the Activity panel on the right side of the page by clicking on the Activity icon along the right.

The Activity panel is contextual if enabled by your administrator. As you click between the header and different child grids on a form, the content updates according to the current selection. If you have focus in the header, activity appears for the header record. If you select a grid row on the form, activity appears for the child record. The panel banner shows the name of the item for which the content is displayed.

You can add comments directly to the activity feeds, and mention the names of other users in the comments by first typing the @ symbol.

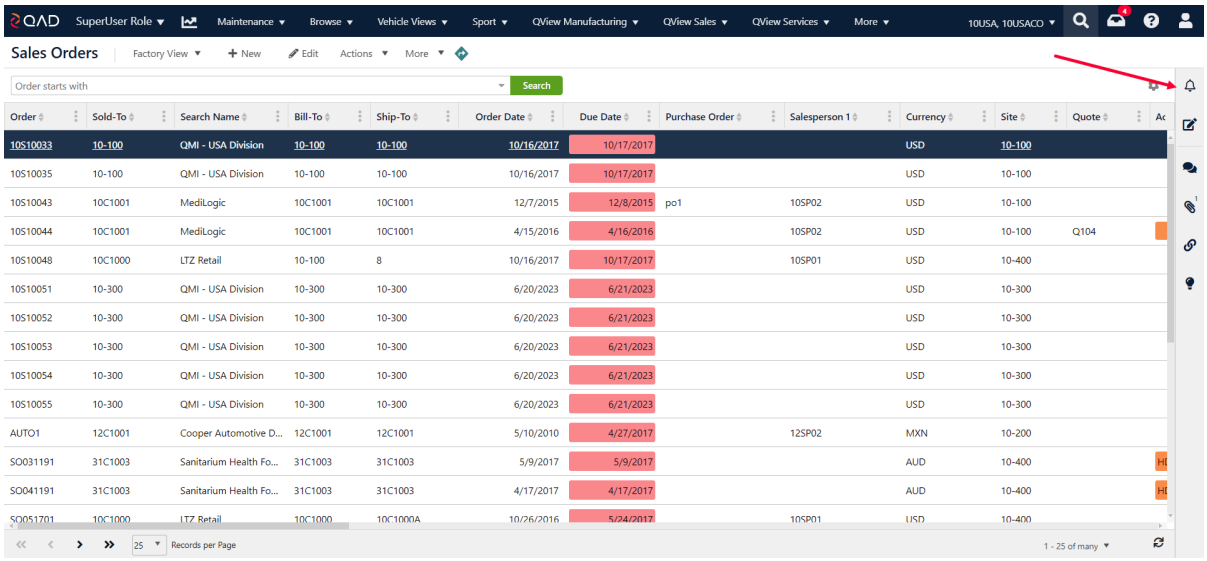
Tagged users receive Inbox notifications that they have been mentioned in a comment. They can then go directly from their Inbox notification to the form and its activity panel. They can also simply reply to the Inbox notification.

You can see these event changes in a view form’s Activity panel and also in the Inbox.

An administrator can configure which fields can be tracked using the Administration > Activity Tracking menu.

Alerts

You can create alerts for selected views, as defined by your administrator. A bell icon is included in the upper right corner of views that are enabled for alerts. Click the bell icon to open the Alerts pop-up window, where you can create your own alerts.



Note An administrator can also create alerts and assign them to you. To modify these alerts, contact your administrator. For administrators, see Alerts on page 7001.

You can select whether you want to receive alert notifications from your user profile settings. See Profile on page 252.

The Alerts pop-up window includes the following:

Main

Alert

Enter a brief description for the alert.

Saved To

Indicates that all alert data is saved as Configuration Data.

Conditions

You can have alerts about changes to fields, and alerts for when specific conditions are met. On the Conditions panel, define the conditions to use for getting alerts when conditions are met.

Click New to define a condition.

Field: select a field from the business component.

Operator: select from the following: equals, does not equal, contains, not contains, range, greater than, greater or equal to, less than, less or equal to, in list, not in list, starts with, ends with, is null, is not null. For

operators that require one value, use the Value1 field. For operators that require two values, use the Value1 and Value2 fields.

Value1: Select from available values or click the toggle icon to select available fields.

Value2: For operators that require two values, select from available values or click the toggle icon to select available fields.

Send alerts about changes to fields

From the drop-down, select *Send alerts about changes to fields* to get notifications about any changes to field values. On the Field Changes panel, select the fields for which you want notifications.

Send alert when conditions are met

From the drop-down, select *Send alert when conditions are met* to define a message and notification delivery options for when the specified conditions are met. On the Message and Notification Options panels, define the message content and delivery options.

Field Changes

This panel is displayed if *Send alerts about changes to fields* is selected.

In the grid, select the fields for which you want alerts for field value changes. Note that only fields that are enabled for activity tracking are available for alerts.

Message

This panel is displayed if *Send alert when conditions are met* is selected.

Enter the message text. To include field values, click Include Field and select from the business component fields. The field is included in the text in the ${field_name} format, indicating that the field value will be included in the text.

Notification Options

This panel is displayed if *Send alert when conditions are met* is selected.

Delivery

Select *Immediate (when conditions are first met)* to have the notification sent as soon as conditions are met. Select Repeat to repeat sending the notification while the conditions are met for the specified Days, Hours, and Minutes.

Select *Delayed (after conditions are still true)* to have the message sent after a time delay (in Days, Hours, and Minutes). Select Repeat to repeat sending the notification while the conditions are still true for the specified Days, Hours, and Minutes.

Select *Relative (before or after a variable date)* to have the notification sent within a specified time (in Days, Hours, and Minutes) before or

after a date provided by selected date field. Select Repeat to repeat sending the notification for the specified Days, Hours, and Minutes.

Repeat

Depending on the Delivery option selected, specify how often to repeat the notification in Days, Hours, and Minutes.

Consolidate Alerts

Select to bundle multiple instances of the same alert into one notification, and provide a label for the consolidated alert notifications. You can also specify how often to send the notification in Days, Hours, and Minutes. For consolidated alerts, the timer starts when the first alert is triggered.

Attachments

You can attach files to views to keep relevant documents together for a particular transaction. For example, you can attach spreadsheets, Word documents, and PDFs to a sales order. You can also attach your notes and add links (to a google doc or slide presentation) as well as preview your PDF and image files.

The Attachments panel is contextual if enabled by your administrator. As you click between the header and different child grids on a form, the content updates according to the current selection. If you have focus in the header, attachments appear for the header record. If you select a grid row on the form, attachments appear for the child record. The panel banner shows the name of the item for which the content is displayed.

Attaching Files

- To attach a file, open the Attachments panel by clicking the paper clip icon. Drag the attachment to the attachments drop area or click **Attach**, and then click **File** to select the attachments.
- To preview an attachment, in the Attachments panel, click the attachment title to open the file in the Preview pop-up window.
- Click the ellipsis button next to the file to preview or delete the attachment, view its properties, or download it.
- Attached files appear under the Files section of the Attachments panel.

Attaching Links

- To add a link, open the Attachments panel by clicking the paper clip icon. Click **Attach**, and then click **Link** to open the Attach Link dialog. Enter your URL and URL display text.
- To open a link, in the Attachments panel, click the link title. The link will always open in a new tab.
- Click the ellipsis button next to the link to open or delete the link, view its properties, or edit it.
- Attached links appear under the External Links section of the Attachments panel.

Attaching Text

- To add a note or any text, open the Attachments panel by clicking the paper clip icon. Click **Attach**, and then click **Text** to open the Attach Text dialog. Enter your text and format it as needed.
- Click the ellipsis button next to the file to edit or delete the text or view its properties.
- The attached text appears under the Files section of the Attachments panel.

Including Attachments in Reports

You can configure QRF reports with an option to include attachments as part of the print report output.

To make an attachment eligible for inclusion, select the Include on Reports checkbox when you upload the attachment to a record—in **Items** or **Sales Orders**, for example.

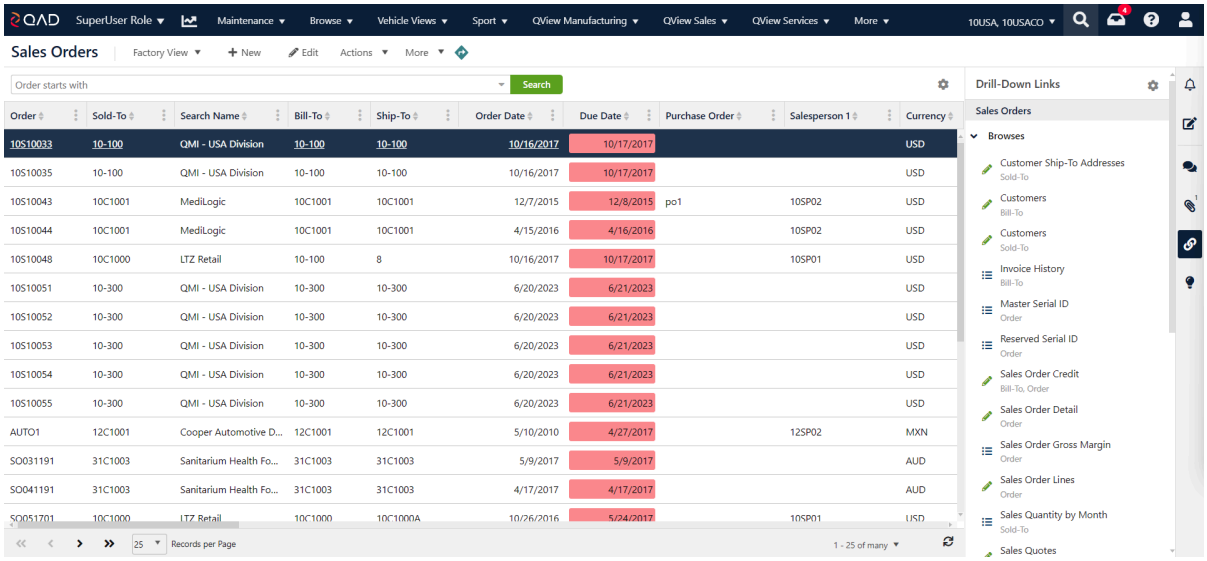
Note Individual QRF reports must be specifically configured to include attachments; changes to the Progress source code are required. For information, see Print Attachments on Reports on page 299.

File Types

Valid attachment file types are defined in whitelists or blacklists. Changes to these lists require a QAD support ticket. Contact your system administrator.

Drill-Down Links

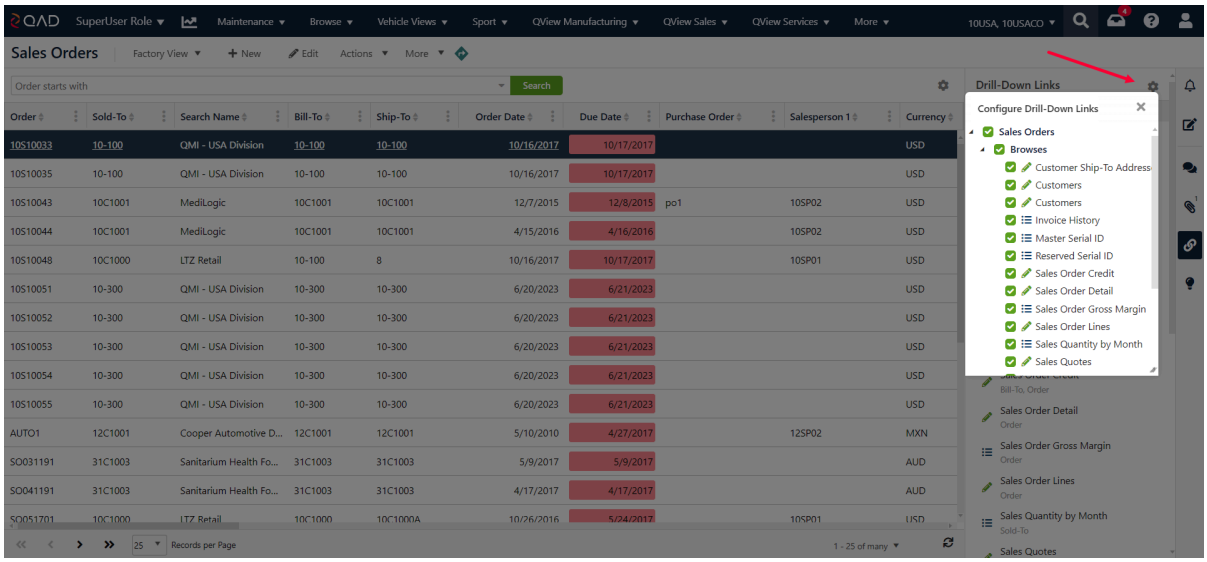
Selected views such as Sales Orders include the drill-down links feature. By clicking on the link icon located on the right side of the view, you can quickly access browses related to data in the currently selected record.



For example, if you have Sales Orders open and are working on a particular order, you can quickly open the Sales Order Credit browse for the Bill-To of the order.

The Drill-Down Links panel is contextual if enabled by your administrator. As you click between the header and different child grids on a form, the content updates according to the current selection. If you have focus in the header, drill-down links appear for the header record. If you select a grid row on the form, drill-down links appear for the child record. The panel banner shows the name of the item for which the content is displayed.

You can configure which drill-down links display by clicking the gear icon, and then removing (or adding) from the available drill-down links.



Hyperlink Drill Downs on Fields

You can view hyperlink drill downs on form fields, form grids, and on browses. Editable fields have a blue drill-down icon for links, but non-editable fields (read-only and disabled) have a hypertext link. Hovering over the link (icon or text) brings up a hover with a list of drill downs on the field. Hyperlinks on browses and form grids are only available on the selected record row.

Hyperlinks appear automatically without setup.

Insights

The Insights panel provides key insights about a record (for example, a sales order or a customer) to make quick and informed decisions. The Insights panel contains visuals created through the action center/analytics functionality. Instead of going to a particular action center, you can access visuals directly from the Insights panel. You have quick access to KPI visual information for your current record context. For example, while in the Sales Orders view, you can see visuals with sales order data, and when looking at a customer, you can review overdue order lines, top items, or invoiced value by month.

The Insights panel is contextual if enabled by your administrator. As you click between the header and different child grids on a form, the content updates according to the current selection. If you have focus in the header, insights appear for the header record. If you select a grid row on the form, insights appear for the child record. The panel banner shows the name of the item for which the content is displayed.

While working in a form, you can quickly access the Insights panel by clicking the light bulb icon located on the right side of the view.

You can use the following options on the chart:

- Expand icon — Opens the visual in a pop-up window, in read-only mode. If you have the edit access to this visual, the gear icon in the upper-right corner appears. Clicking the gear icon, opens the visual builder, where you can make changes to this visual and save it. This may be helpful for an admin who decides to make changes to the chart on the fly so that it fits better or displays the data better for the context.
- Timestamp and Refresh — Shows the timestamp for the visual data, with the option to refresh for the latest data.
- Drill-out icon — Drills directly to the data source, for example, opens the browse with a filter for the customer 10-C1001.

The Insights panel can also be resized.

You can configure which insights display by clicking the gear icon, and then removing (or adding) from the available insights.

Limitations

Insights are not available for embedded business components or internal grids that do not link to a business component (usually the grids in Actions).

Messages

Information and warning messages are displayed on the Messages panel, which reduces potential distractions during data entry while improving message accessibility.

While working in a form, you can quickly access the Messages panel by clicking the messages icon on the right side of the view.

To reduce clutter, you can clear currently displayed messages by clicking the Clear All button in the lower-right corner of the Messages panel.

Note that error messages are displayed at the bottom of the form, in an Errors grid.

Reports

You can run reports using the report viewer. When you first open a report, complete the Settings and Filter panels. Next, click Run to run the report right away or schedule the report to be delivered in your Inbox using Schedule > Schedule Report.

Some reports can take time to render. After you hit the Run button, a new tab opens with options for viewing the report or having the report sent to your Inbox.

You can save a report with its current settings and filter by using the stored reports drop-down option.

Report Viewer Settings and Filter

The Settings panel, which is closed by default, includes sub-panels for Format, Date Display, and Decimal Display settings.

The following fields can be edited in the Format sub-panel:

File Type

Specify the file type that you want to use for the report. The default file type is PDF but you can specify from PDF, Excel, Plain Text, PDF Read-only, XML, and XMl with schema.

Filter Criteria Display

Specify whether you want to display the criteria of the report at the header, the footer, or not at all.

Language

Specify the language that you want to display the report in.

Layout

Select the layout for the report if multiple layouts have been specified in the report definition. Otherwise, this field is not active.

Date Display

The following fields can be edited in the Date Display sub-panel:

Short Date Format

Specify the format in which you want to display the date in the report.

Date Separator

Specify the separator you want to use for the date in the report.

Decimal Display

The following fields can be edited in the Decimal Display sub-panel:

Decimal Separator

Choose a decimal separator to be used in the report.

Decimal Digits

Choose the number of decimal digits to display in the report.

Grouping Separator

Choose a grouping separator to be used in in the report.

Digit Grouping

Choose the method of grouping you want to use in the report.

Filter

The **Filter** panel, which is open by default, includes the criteria for the data used in the report.

If a report has been configured to include attachments as part of the output, you may see that option as one of the filters. For information on this topic, see Print Attachments on Reports on page 299.

Stored Reports

For the currently selected report, you can save it as a stored report in a similar way that you can use the stored views feature.

Initially, when you select a report, the Default Report is displayed.

After configuring a report, you can save it as a stored report by selecting Save As and completing the following modal dialog.

Once saved, a stored report is available from a pull-down menu located next to the report title. Stored reports can be made available just to you or can be made available to everyone on the system.

For first time access, the Default Report is displayed. If there is more than one QAD default report, the first report in alphabetical order is displayed.

Scheduled Reports

To schedule a report run, choose Schedule > Schedule Report and specify the settings in the Schedule Report modal window.

Also, from the Schedule drop-down, you can choose View Schedule and View History:

- View Schedule opens the Scheduled Report browse grid in an overlay window displaying on the Scheduled Report browse.
- View History opens the Scheduled Report History browse grid in an overlay window displaying the Scheduled Report History browse.

General

The following fields can be edited in the General sub-panel:

Description

Enter a short, meaningful description of the scheduled report for batch delivery.

Batch ID

Specify the batch ID that you want to use for the scheduled report.

Printer

Select a printer to send the report to if you want to have the scheduled report printed.

Report Output

The following fields can be edited in the Report Output sub-panel:

Save Report Output

Select this option to direct the report server to send the scheduled report output file to the document service. This option is selected by default.

Output File Path

This field specifies the output file path on the server.

Output File Name

This field specifies the output file name.

Run Only Once

Select this field if you want the scheduled report to be run only once. This field is selected by default.

Report Delivery

The following fields can be edited in the Report Delivery sub-panel:

Send To

Enter a comma-separated list of email addresses and user IDs. Reports are delivered to your Inbox as messages including links to reports.

Send As Attachment

Specify whether you want to include the report file as an attachment. The notification message will always include a link to the report located on the server. For security reasons, QAD recommends just sending a link to the report, but some situations can require that the report be included as an attachment.

Print Attachments on Reports

You can configure QRF reports with an option to include attachments as part of the print report output. A few potential examples include:

- A printed maintenance work order can include technical information for the maintenance technician to use while working on a machine.
- You can send the supplier a purchase order that includes the quote.
- You can include a copy of the packing slip when printing a purchase order. A sales order confirmation can include detailed order conditions.
- A pre-shipper/shipper can provide special instructions related to the item or container. An RMA report can include a photo of the defective item.
- A printed service contract report can include a PDF of the actual contract.

These scenarios are provided as potential applications for the report attachment feature. None have been tested or implemented in the delivered version of QAD Adaptive.

Important The configuration process requires changes to the Progress source code (.p) for the QRF report.

Configuring a Report to Include Attachments

This section describes key steps in the configuration process, including:

- Add the attached file temp tables to the reports dataset sent to QRF.
- Ensure that the temp tables are empty.
- Define the attachments to be added to the dataset.
- Include attachments report filters.

So that you can see the code updates in context, sample programs—as well as some reference material—are provided as links:

- Item Inventory Type Report Example on page 318
- Item Cost Report Example on page 303
- Dataset Reporting Utility on page 337

CONFIGURATION TASKS

Code snippets in these steps refer to the Item Inventory Type Report Example on page 318.

1. Define Dataset

The first requirement is to add the attached file temp tables to the reports dataset sent to QRF, as shown in this sample code:

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

```
define dataset dsReportResults for

  ttReportOptions, tt_report_results, tt_logical_expression

  &IF DEFINED(supportsAttachments) &THEN

  , {com/qad/shell/report/addAttachedFileTables.i}

  &ENDIF

  .
```

Note The "IF DEFINED" takes care of the case where the underlying .NET UI code does not have the required definitions.

2. Empty Temp Tables

The next step is to ensure that the tables are empty. Add this to the startup where the other temp tables are emptied:

```
&IF DEFINED(supportsAttachments) &THEN

empty temp-table ttAttachedFileInfo  no-error.

empty temp-table ttAttachedFileHeader  no-error.

empty temp-table ttAttachedFileDetails no-error.

&ENDIF
```

3. Add to Tables

The following code adds the **Sites** attachments, if any, to the dataset:

```
for first si_mstr no-lock

  where si_mstr.si_domain = global_domain

    and   si_mstr.si_site = in_mstr.in_site:

end.


if available si_mstr

then do:
```

```
  &IF DEFINED(supportsAttachments) &THEN

  attachedFileURN = "urn:be:com.qad.base.item.isite:" + global_domain + "." +
si_mstr.si_site.

  /* Header and Values are read in as list of display names and database values */

  run AddAttachedFiles(

  attachedFileURN,

  "qad-erp-base",

  "Site|Description",

  si_mstr.si_site + "|" + si_mstr.si_desc).

  &ENDIF
```

Where:

- attachedFileURN = "urn:be:com.qad.base.item.isite:" + global_domain + "." + si_mstr.si_site.

  is the URN used by the Web UI to find the attached files. As a developer you should know what this is. But in fact few people actually deal with this level of detail.
  To find it, go to Web UI and attach a file. Look at the Network trace. In this case, a site is defined by domain and siteid.

- "qad-erp-base"

  is the plug-in ID which controls this attachment key mapping. Some plug-ins use a key mapping service, and this is important.

- "Site|Description"

  is a pipe-delimited list of key fields. These are displayed on the final PDF.
  This is the value for the above: si_mstr.si_site + "|" + si_mstr.si_desc.

Another example illustrates how to add **Product Lines** attachments to the dataset:

```
if available pl_mstr then do:

  ttSiteInfo.pldesc = pl_mstr.pl_desc.



  &IF DEFINED(supportsAttachments) &THEN
```

```
  run AddAttachedFiles in reportHandle(

  "urn:be:com.qad.base.item.IProductLine:" + global_domain + "." +
pl_mstr.pl_prod_line,

  "qad-erp-base",

  "Product Line|Description", pl_mstr.pl_prod_line + "|" + pl_mstr.pl_desc).

  &ENDIF



end.
```

Your code may require additional logic. See the "AddAttachedFiles" method in Dataset Reporting Utility on page 337.

4. Add Attachment Filters

When including attachments in reports, include a report filter for each table from which you are querying attachments. For example, if you are including item attachments in an Item report, add a filter called "Include Attachments" with the default set to No. If you are including attachments from both items and sites, add two separate filters "Include Item Attachments" and "Include Site Attachments". This gives the user the flexibility to include all attachments, no attachments, just item attachments, or just site attachments.

Some guidelines to keep in mind when you are designing filters:

- If the report context matches the attachment context (Items report pulling Item attachments), use filter label: "Include Attachments".
- If the report context is different from the attachment context (Work Order report pulling Equipment attachments), use filter label "Include [table name] Attachments", where table name specifies where the attachments are coming from; for example, "Include Equipment Attachments".
- If including attachments from more than one table, include a filter for each table and specify the name in the filter; for example, "Include Item Attachments" and "Include Site Attachments".
- Placement of the filter can vary depending on the business scenario and attachment importance. Group with other 'Include' filters, if there are any.
- Default = No. This is intended to minimize mistakes and accidental inclusion of long attachments.

Indicating Attachments To Be Included on Reports

To enable a modified QRF report to include attachments from related records, select the Include on Reports checkbox when you add an attachment to a record, such as an Item or a Sales Order. For example, if the QRF report concerns item data, select Include on Reports for any attachments defined in Items that you want to include in report output. The field is unselected by default to help prevent unintended consequences; for example, internal documents included in report outputs that go to an external audience.

Note The field displays whenever you add an attachment. However, the attachment is included in a report only when one has been configured to support attachments to records of a specific type. If it is not, selecting Include on Reports has no effect.

Other Considerations

- Only reports rendered to PDF include attachment content. The feature is not supported for reports output as RTF, XLS, PlainText, and so on.
- You can only include a specific attachment on all reports that contain attachments from that screen. For instance, you cannot select a particular item drawing attachment to be included solely on a Production Orders Report, and not on a Sales Orders report.
- Reports either include all attachments or none at all, without any selective inclusion. To indicate which attachments should be included, you can utilize the 'Include in Reports' attachment property.
- Supported attachment file types are PDF, images (jpg, png, svg, gif, bmp, cgm, dicom, emf, and tiff), and text files (in any of its variations like html, xml, PlainText, and so on).
- If images are included, they will be scaled down as needed (but never up) to fit the report page size.

Item Cost Report Example

```
/* ACME_ItemCostReport.p - Item Cost Report                          */

/* Copyright 1986-2011 QAD Inc., Santa Barbara, CA, USA.             */

/* All rights reserved worldwide.  This is an unpublished work.      */

/* $Id$ */

/*                                                                   */

/* Report matches legacy ppptrp13.p for use by QAD Reporting Framework.
 */

/*                                                                   */
```

```
{mfsubdirs.i}

{{&US_BBI}mfdeclre.i}

{{&US_BBI}gplabel.i}

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

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

/
*============================================================================*/

/* ***************************  TEMP TABLES
 ****************************** */

/
*============================================================================*/

/* TEMP-TABLE DEFINITION BLOCK */

define temp-table ttSiteInfo no-undo

  field si_site      like si_mstr.si_site

  field in_site      like in_mstr.in_site

  field si_desc      like si_mstr.si_desc

  field pt_prod_line like pt_mstr.pt_prod_line

  field pldesc       like pl_mstr.pl_desc

  field uniqueid     as   integer

  field pageBreak    like mfc_logical

.

define temp-table tt_pt_mstr no-undo like pt_mstr

  field referenceid  as   integer
```

```
  field parentID     as   integer
  index my_index parentID
.

define temp-table tt_sct_det no-undo like sct_det
  field spt_sim like spt_det.spt_sim
  field referenceid  as   integer
  field sct_mtl like sct_det.sct_mtl_tl
  field sct_lbr like sct_det.sct_lbr_tl
  field sct_bdn like sct_det.sct_bdn_tl
  field sct_ovh like sct_det.sct_ovh_tl
  field sct_sub like sct_det.sct_sub_tl
  index my_index referenceid
.

/
*============================================================================*/

/* ***************************  MAIN BLOCK
 *************************************** */

/
*============================================================================*/

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

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

```
&IF DEFINED(supportsAttachments1) &THEN

/* Attachment links data table defintions */

{com/qad/shell/report/ttAttachedFiles.i &FIELDNAME="si_site"
 &FIELDTYPE="character"}

{com/qad/shell/report/ttAttachedFiles.i &FIELDNAME="pt_prod_line"
 &AINDEX="2"}

&ENDIF



message "Running Item cost".

define dataset dsReportResults for ttSiteInfo, tt_pt_mstr, tt_sct_det


&IF DEFINED(supportsAttachments1) &THEN

  /* Put attachment tables into Report dataset  */

  , {com/qad/shell/report/addAttachedFileTables.i}

  , {com/qad/shell/report/addAttachedFileTables.i &AINDEX="2"}

&ENDIF


&IF DEFINED(supportsAttachments1) &THEN

  /* Attachment links dataset relations */

  {com/qad/shell/report/AddAttachedDataRelations.i
&TOP_TABLE="ttSiteInfo" &FIELDNAME="si_site"}

  {com/qad/shell/report/AddAttachedDataRelations.i
&TOP_TABLE="ttSiteInfo" &FIELDNAME="pt_prod_line" &AINDEX="2"}

&ENDIF


 .


{com/qad/shell/report/reports/QAD_HelperFunctions.i}
```

```
define variable vhDS as handle    no-undo.

empty temp-table ttSiteInfo no-error.
empty temp-table tt_pt_mstr no-error.
empty temp-table tt_sct_det no-error.

&IF DEFINED(supportsAttachments1) &THEN
empty temp-table ttAttachedFileInfo     no-error.
empty temp-table ttAttachedFileHeader   no-error.
empty temp-table ttAttachedFileDetails  no-error.

empty temp-table ttAttachedFileInfo_2    no-error.
empty temp-table ttAttachedFileHeader_2  no-error.
empty temp-table ttAttachedFileDetails_2 no-error.
&ENDIF

for first ttReportRequest no-lock:

  vhDS = dataset dsReportResults:handle.
  run FillMetaData in reportHandle(
    vhDS, "ACME_ItemCostReport.meta.xml").

  run SetReportMetaData (input reportHandle).

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

```
  delete object phReportResults no-error.

end.

PROCEDURE SetReportMetaData:

  define input parameter  ipReportHandle as handle.

  run SetFieldMetaParameter in ipReportHandle(
       "ttSiteInfo", "in_site", "DefaultValue", global_site).

END PROCEDURE. /* PROCEDURE ModifyMetaData */

/* DATA RETRIEVING LOGIC BLOCK */
PROCEDURE RunReport:

  define output parameter dataset-handle phReportResults.

  /* RETRIEVE THE DATA FROM DATABASE */
  define variable queryString   as character no-undo.
  define variable hMainQuery    as handle    no-undo.
  define query    MainQuery     for pt_mstr, in_mstr.
  define variable l_exist       like mfc_logical no-undo.
  define variable parentSite    as integer   no-undo.
  define variable itemID        as integer   no-undo.
  define variable CostSetRange  like mfc_logical no-undo.

  assign
```

```
   hMainQuery          = query MainQuery:handle.

  for first icc_ctrl where icc_ctrl.icc_domain = global_domain
  no-lock:
  end.

  queryString = "for each pt_mstr no-lock use-index pt_prod_part" +
  " where pt_mstr.pt_domain = " + quoter(global_domain).

  run AddAllConditions in reportHandle
    ("pt_mstr", input-output queryString).

  queryString = queryString + ",each in_mstr no-lock where in_mstr.in_domain"
+
  " = " + quoter(global_domain).

  run AddSpecificConditions in reportHandle
    ("in_mstr",
     "in_site",
     input-output queryString).

  queryString = queryString + " and in_part = pt_part"
                        + " break by in_site by pt_prod_line by pt_part"
                        + ":".
  hMainQuery:query-prepare(queryString).
  hMainQuery:query-open().
  hMainQuery:get-next().

  repeat while not hMainQuery:query-off-end:
```

```
for first si_mstr where si_mstr.si_domain = global_domain
  and si_mstr.si_site = in_mstr.in_site no-lock:
end.
if si_db <> global_db
then do:
  hMainQuery:get-next().
  next.
end.

for first ptp_det where ptp_det.ptp_domain = global_domain and ptp_part =
 pt_mstr.pt_part and ptp_site = si_mstr.si_site no-lock:
end.
if available ptp_det
then do:
  if not ValuateInputCharacter(ptp_buyer,"tt_pt_mstr","pt_buyer")
  then do:
    hMainQuery:get-next().
    next.
  end.
end.
else if not ValuateInputCharacter(pt_buyer,"tt_pt_mstr","pt_buyer")
then do:
  hMainQuery:get-next().
  next.
end.
```

```
l_exist = no.
for each sct_det
   where sct_det.sct_domain = global_domain
   and   sct_part           = pt_mstr.pt_part
   and   sct_site           = si_mstr.si_site
no-lock:
   if not ValuateInputCharacter(sct_sim,"tt_sct_det","spt_sim")
   then do:
      next.
   end.
   l_exist = yes.
   leave.
end. /* FOR EACH sct_det */

if not l_exist
then do:
   hMainQuery:get-next().
   next.
end.

&IF DEFINED(supportsAttachments1) &THEN
run AddAttachedFiles(
si_mstr.si_site,
"urn:be:com.qad.base.item.isite:" + global_domain + "." + si_mstr.si_site,
"qad-erp-base",
```

```
    "Site|Description",
    si_mstr.si_site + "|" + si_mstr.si_desc).
    &ENDIF

    create ttSiteInfo.
    assign
      parentSite          = parentSite + 1
      ttSiteInfo.uniqueid  = parentSite
      ttSiteInfo.si_site   = si_mstr.si_site
      ttSiteInfo.si_desc   = si_mstr.si_desc
      ttSiteInfo.pageBreak = true
    .

    for first pl_mstr where pl_mstr.pl_domain = global_domain and
    pl_prod_line = pt_mstr.pt_prod_line no-lock:
    end.

    ttSiteInfo.pt_prod_line = pt_mstr.pt_prod_line.

    if available pl_mstr then do:
      ttSiteInfo.pldesc = pl_mstr.pl_desc.

      &IF DEFINED(supportsAttachments1) &THEN
      run AddAttachedFiles_2(
      pl_mstr.pl_prod_line,
      "urn:be:com.qad.base.item.IProductLine:" + global_domain + "." +
pl_mstr.pl_prod_line,
```

```
  "qad-erp-base",
  "Product Line|Description",
  pl_mstr.pl_prod_line + "|" + pl_mstr.pl_desc).
  &ENDIF

end.

create tt_pt_mstr.
assign
  itemID               = itemID + 1
  tt_pt_mstr.parentID  = itemID
  tt_pt_mstr.referenceid = parentSite
  tt_pt_mstr.pt_part   = pt_mstr.pt_part
  tt_pt_mstr.pt_um     = pt_mstr.pt_um
  tt_pt_mstr.pt_desc1  = pt_mstr.pt_desc1
  tt_pt_mstr.pt_desc2  = pt_mstr.pt_desc2
.

if in_cur_set <> ""
then do:
  for first sct_det where sct_det.sct_domain = global_domain
    and sct_part = pt_mstr.pt_part and sct_det.sct_sim = in_cur_set
    and sct_site = si_mstr.si_site no-lock:
  end.
end.
else do:
```

```
  for first sct_det where sct_det.sct_domain = global_domain
    and sct_part = pt_mstr.pt_part and sct_det.sct_sim = icc_cur_set
    and sct_site = si_mstr.si_site no-lock:
  end.
end.

if available sct_det
  and ValuateInputCharacter(sct_sim,"tt_sct_det","spt_sim")
then do:
  create tt_sct_det.
  assign
    tt_sct_det.referenceid  = itemID
    tt_sct_det.spt_sim      = sct_det.sct_sim
    tt_sct_det.sct_mtl      = sct_det.sct_mtl_tl + sct_det.sct_mtl_ll
    tt_sct_det.sct_lbr      = sct_det.sct_lbr_tl + sct_det.sct_lbr_ll
    tt_sct_det.sct_bdn      = sct_det.sct_bdn_tl + sct_det.sct_bdn_ll
    tt_sct_det.sct_ovh      = sct_det.sct_ovh_tl + sct_det.sct_ovh_ll
    tt_sct_det.sct_sub      = sct_det.sct_sub_tl + sct_det.sct_sub_ll
    tt_sct_det.sct_cst_tot  = sct_det.sct_cst_tot
    tt_sct_det.sct_cst_date = sct_det.sct_cst_date
  .
end.

if in_gl_set <> ""
then do:
  for first sct_det where sct_det.sct_domain = global_domain
```

```
      and sct_det.sct_part = pt_mstr.pt_part
      and sct_det.sct_sim = in_gl_set
      and sct_det.sct_site = in_gl_cost_site no-lock:
    end.
  end.
  else do:
    for first sct_det where sct_det.sct_domain = global_domain
      and sct_det.sct_part = pt_mstr.pt_part
      and sct_det.sct_sim  = icc_gl_set
      and sct_det.sct_site = in_gl_cost_site no-lock:
    end.
  end.
  if available sct_det
    and ValuateInputCharacter(sct_det.sct_sim,"tt_sct_det","spt_sim")
  then do:
    create tt_sct_det.
    assign
      tt_sct_det.referenceid  = itemID
      tt_sct_det.spt_sim      = sct_det.sct_sim
      tt_sct_det.sct_mtl      = sct_det.sct_mtl_tl + sct_det.sct_mtl_ll
      tt_sct_det.sct_lbr      = sct_det.sct_lbr_tl + sct_det.sct_lbr_ll
      tt_sct_det.sct_bdn      = sct_det.sct_bdn_tl + sct_det.sct_bdn_ll
      tt_sct_det.sct_ovh      = sct_det.sct_ovh_tl + sct_det.sct_ovh_ll
      tt_sct_det.sct_sub      = sct_det.sct_sub_tl + sct_det.sct_sub_ll
```

```
      tt_sct_det.sct_cst_tot  = sct_det.sct_cst_tot
      tt_sct_det.sct_cst_date = sct_det.sct_cst_date
    .
  end.


  for each sct_det
    where  sct_det.sct_domain = global_domain
    and    sct_det.sct_part   = pt_mstr.pt_part
    and    sct_det.sct_site   = si_mstr.si_site
    and    sct_det.sct_sim   <> si_gl_set
    and   (si_gl_set          = ""
    and    sct_det.sct_sim   <> icc_gl_set)
    and    sct_det.sct_sim   <> si_cur_set
    and   (si_cur_set         = ""
    and    sct_det.sct_sim   <> icc_cur_set)
  no-lock:
    CostSetRange =
ValuateInputCharacter(sct_det.sct_sim,"tt_sct_det","spt_sim").
    if CostSetRange
    then do :
      create tt_sct_det.
      assign
        tt_sct_det.referenceid  = itemID
        tt_sct_det.spt_sim      = sct_det.sct_sim
        tt_sct_det.sct_mtl      = sct_det.sct_mtl_tl + sct_det.sct_mtl_ll
```

```
            tt_sct_det.sct_lbr      = sct_det.sct_lbr_tl + sct_det.sct_lbr_ll
            tt_sct_det.sct_bdn      = sct_det.sct_bdn_tl + sct_det.sct_bdn_ll
            tt_sct_det.sct_ovh      = sct_det.sct_ovh_tl + sct_det.sct_ovh_ll
            tt_sct_det.sct_sub      = sct_det.sct_sub_tl + sct_det.sct_sub_ll
            tt_sct_det.sct_cst_tot  = sct_det.sct_cst_tot
            tt_sct_det.sct_cst_date = sct_det.sct_cst_date
          .
        end. /* IF CostSetRange */
      end.

      hMainQuery:get-next().
    end. /* REPEAT QUERY */


    hMainQuery:query-close().
    delete object hMainQuery.
    ttSiteInfo.pageBreak = false.
    phReportResults = dataset dsReportResults:handle.
    delete object phReportResults no-error.

END PROCEDURE.

&IF DEFINED(supportsAttachments1) &THEN
{com/qad/shell/report/AddAttachedFiles.i &FIELDNAME="si_site"}
{com/qad/shell/report/AddAttachedFiles.i &FIELDNAME="pt_prod_line"
 &AINDEX="2"}
&ENDIF
```

Item Inventory Type Report Example

```
/* ACME_ItemInventoryTypeReport.p - ITEM INVENTORY TYPE REPORT
          */

/* Copyright 1986-2011 QAD Inc., Santa Barbara, CA, USA.                  */

/* All rights reserved worldwide.  This is an unpublished work.           */

/* $Id$ */

/*                                                                        */

/* Report matches legacy icobirp.p for use by QAD Reporting Framework.
 */

/*                                                                        */


{mfsubdirs.i}

{{&US_BBI}mfdeclre.i}

{{&US_BBI}gplabel.i}


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

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


{{&US_IC}icobvrit.i}

{{&US_IC}icobwtlo.i "new shared"}


{{&US_PX}pxmaint.i}

{{&US_PX}pxphdef.i ppitxr}

{{&US_GP}gprunpdf.i "ppccxr1" "p"}

{{&US_GP}gprunpdf.i "icobxr" "p" }


/
*============================================================================*/
```

```
/* ***************************  TEMP TABLES
 ****************************** */

/
*============================================================================*/

/* Temp-table definition block */

define temp-table ttReportOptions no-undo

  field tt_conflicts_only as logical

  field tt_show_logic     as logical

  field tt_incl_site_attach as logical

.

define temp-table tt_report_results no-undo

  field unique_id      as   integer

  field in_part        like in_mstr.in_part

  field pt_desc1       like pt_mstr.pt_desc1

  field in_site        like in_mstr.in_site

  field tt_inven_type  like iat_mstr.iat_inven_type

  field pt_prod_line   like pt_mstr.pt_prod_line

  field pt_part_type   like pt_mstr.pt_part_type

  field pt_group       like pt_mstr.pt_group

  field comd_comm_code like comd_det.comd_comm_code

  field pti_abc        like pti_det.pti_abc

  field pt_pm_code     like pt_mstr.pt_pm_code

  field pt_buyer       like pt_mstr.pt_buyer

  field tt_conflict    as   character
```

```
.

define temp-table tt_logical_expression no-undo
  field ref_id         as integer
  field tt_logic_line  as character
.

define temp-table ttInvenType no-undo
  field tt_inven_type1 like iat_mstr.iat_inven_type
  field tt_inven_type2 like iat_mstr.iat_inven_type
.

/
*============================================================================*/

/* ***************************  MAIN BLOCK
 *************************** */

/
*============================================================================*/

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

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

define variable p-incl-site-attach like mfc_logical no-undo.
&IF DEFINED(supportsAttachments1) &THEN
      if p-incl-site-attach
      then do:
```

```
  /* Attachment links data table defintions */

  {com/qad/shell/report/ttAttachedFiles.i &FIELDNAME="in_site"
&FIELDTYPE="character"}

      end.

&ENDIF


define dataset dsReportResults for

  ttReportOptions, tt_report_results, tt_logical_expression

&IF DEFINED(supportsAttachments1) &THEN

  /* Put attachment tables into Report dataset  */

  , {com/qad/shell/report/addAttachedFileTables.i}

&ENDIF


&IF DEFINED(supportsAttachments1) &THEN

  /* Attachment links dataset relations */

  {com/qad/shell/report/AddAttachedDataRelations.i
&TOP_TABLE="tt_report_results" &FIELDNAME="in_site"}

&ENDIF

  .


define variable vhDS            as handle no-undo.


empty temp-table ttReportOptions       no-error.

empty temp-table tt_report_results     no-error.

empty temp-table tt_logical_expression no-error.

empty temp-table ttInvenType           no-error.


&IF DEFINED(supportsAttachments1) &THEN
```

```
empty temp-table ttAttachedFileInfo    no-error.
empty temp-table ttAttachedFileHeader  no-error.
empty temp-table ttAttachedFileDetails no-error.
&ENDIF

for first ttReportRequest no-lock:

  vhDS = dataset dsReportResults:handle.
  run FillMetaData in reportHandle(
    vhDS, "ACME_ItemInventoryTypeReport.meta.xml").
  run SetReportMetaData (input reportHandle).
  if runReport then do:
    run RunReport
      (output dataset-handle phReportResults).
  end.

  delete object phReportResults no-error.

end.
PROCEDURE SetReportMetaData:

  define input parameter  ipReportHandle as handle.

  run SetFieldMetaParameter in ipReportHandle(
          "tt_report_results", "in_site", "DefaultValue", global_site).

END PROCEDURE. /* PROCEDURE ModifyMetaData */
/* Data retrieving logic block */
```

```
PROCEDURE RunReport:

  define output parameter dataset-handle phReportResults.
  /* Retrieve the data from database */
  define variable queryString      as   character   no-undo.
  define variable hINQuery         as   handle      no-undo.
  define query    INQuery          for  in_mstr.
  define variable old_date_fmt     as   character   no-undo.
  define variable p-inven-type     like iat_inven_type no-undo.
  define variable p-inven-type1    like iat_inven_type no-undo.
  define variable p-conflicts-only like mfc_logical no-undo.
  define variable p-show-logic     like mfc_logical no-undo.
  define variable l-inven-type     like iat_inven_type no-undo.
  define variable isRange          as   logical     initial false no-undo.
  define variable l-match          as   logical     no-undo.
  define variable i1               as   integer     no-undo.
  define variable i2               as   integer     no-undo.
  define variable unique_id        as   integer     initial 0     no-undo.
  define variable attachedFileURN  as   character   no-undo.

  for each ttFilterCondition no-lock
    where ttFilterCondition.bufferName  = "tt_report_results"
      and ttFilterCondition.bufferField = "tt_inven_type"
  break by ttFilterCondition.filterSequence1
  by ttFilterCondition.filterSequence2:
```

```
if first-of(filterSequence2)
then do:
  isRange = false.
  CASE filterOperator:
    when '=' then do:
      create ttInvenType.
      assign
        tt_inven_type1  = filterValue
        tt_inven_type2  = filterValue.
    end.
    when '<>' then do:
      create ttInvenType.
      assign
        tt_inven_type1  = ""
        tt_inven_type2  = filterValue.
      create ttInvenType.
      assign
        tt_inven_type1  = filterValue
        tt_inven_type2  = hi_char.
    end.
    when '>' then do:
      create ttInvenType.
      assign
        tt_inven_type1  = filterValue
```

```
        tt_inven_type2  = hi_char.
    end.
    when '<' then do:
      create ttInvenType.
      assign
        tt_inven_type1  = ""
        tt_inven_type2  = filterValue.
    end.
    when '>=' then do:
      isRange = true.
      create ttInvenType.
      assign
        tt_inven_type1  = filterValue
        tt_inven_type2  = hi_char.
      next.
    end.
    when '<=' then do:
      isRange = true.
      create ttInvenType.
      assign
        tt_inven_type1  = ""
        tt_inven_type2  = filterValue.
      next.
    end.
  END CASE.
```

```
    end.
    if isRange
    then do:
      CASE filterOperator:
        when '<=' then do:
          tt_inven_type2  = filterValue.
        end.
        when '>=' then do:
          tt_inven_type1  = filterValue.
        end.
      END CASE.
      isRange = false.
    end.
  end.

  assign
    p-conflicts-only     =
                IsTrueValue("ttReportOptions", "tt_conflicts_only", false)
    p-show-logic         =
                IsTrueValue("ttReportOptions", "tt_show_logic", false)
    p-incl-site-attach   =
                IsTrueValue("ttReportOptions", "tt_incl_site_attach", true)
    hINQuery             =  query INQuery:handle
    old_date_fmt         =  session:date-format
    session:date-format  =  "ymd"
```

```
  queryString         =  "for each in_mstr no-lock where in_domain = "
                      + quoter(global_domain).

run AddAllConditions in reportHandle
  ("in_mstr", input-output queryString).

queryString = queryString + ":".
for first ttInvenType no-lock:
end.
if not available ttInvenType
then do:
  create ttInvenType.
  assign
    ttInvenType.tt_inven_type1 = ""
    ttInvenType.tt_inven_type2 = hi_char.
end.
for each ttInvenType no-lock:
  assign
    p-inven-type  = ttInvenType.tt_inven_type1
    p-inven-type1 = ttInvenType.tt_inven_type2.

hINQuery:query-prepare(queryString).
hINQuery:query-open().
session:date-format = old_date_fmt.
hINQuery:get-next().
```

```
unique_id = 0.
item-master:
repeat while not hINQuery:query-off-end:
  for first pt_mstr no-lock
    where pt_mstr.pt_domain = global_domain
     and   pt_mstr.pt_part = in_mstr.in_part:
    for first ptp_det no-lock
      where ptp_det.ptp_domain = global_domain
       and   ptp_det.ptp_part = in_mstr.in_part
       and   ptp_det.ptp_site = in_mstr.in_site:
    end.

    for first si_mstr no-lock
      where si_mstr.si_domain = global_domain
       and   si_mstr.si_site = in_mstr.in_site:
    end.

    if available si_mstr
    then do:
      &IF DEFINED(supportsAttachments1) &THEN
        if p-incl-site-attach
        then do:
        attachedFileURN = "urn:be:com.qad.base.item.isite:" + global_domain
+ "." + si_mstr.si_site.
        /* Header and Values are read in as list of diplay names and database
values */
        run AddAttachedFiles(
```

```
            si_mstr.si_site,
            attachedFileURN,
            "qad-erp-base",
            "Site|Description",
        si_mstr.si_site + "|" + si_mstr.si_desc
                ).
            end.
          &ENDIF


        {{&US_PX}pxrun.i &PROC  = 'readInventoryDetail' &PROGRAM =
'ppitxr.p'
            &HANDLE=ph_ppitxr
            &PARAM = "(input  pt_mstr.oid_pt_mstr,
                    input  si_mstr.oid_si_mstr,
                    buffer pti_det)"
            &NOAPPERROR = true
            &CATCHERROR = true
          }
        end.

        /* GET COMMODITY CODE */
        {{&US_GP}gprunp.i "ppccxr1" "p" "processRead"
        "(input pt_part,
          buffer comd_det,
          no,
```

```
yes)"}

/* DETERMINE INVENTORY TYPE */
l-fields
= pt_mstr.pt_part
+ ";"
+ pt_mstr.pt_prod_line
+ ";"
+ pt_mstr.pt_part_type
+ ";"
+ pt_mstr.pt_group
+ ";"
+ (if available comd_det
then comd_comm_code
else "")
+ ";"
+ (if available pti_det
then pti_abc
else pt_mstr.pt_abc)
+ ";"
+ (if available ptp_det
  then ptp_pm_code
  else pt_mstr.pt_pm_code)
+ ";"
+ in_mstr.in_site
```

```
      + ";"
      + (if available ptp_det
      then ptp_buyer
      else pt_buyer)
      .

      {{&US_GP}gprunp.i "icobxr" "p" "lookup-inven-type"
        "(input l-fields,
           output l-inven-types,
           output l-matches)"}

      /* RETURNS 2 SEMICOLON-SEPARATED LISTS, */
      /*  SEQUENCED ON DESCENDING NUMBER OF FIELDS, */
      /* "icobxr" "p"  ASCENDING INVENTORY TYPE. */

      if num-entries(l-inven-types, ";") > 0
      /* ITEM/SITE BELONGS IN ONE OR MULTIPLE INVENTORY
TYPES */
      then do:

        /* CHECK THAT ONE OF THE TYPES IS WITHIN TYPE RANGE */
        l-match = no.
        inven-type-loop:
        do i1 = 1 to num-entries(l-inven-types, ";"):
          l-inven-type = entry(i1,l-inven-types, ";").
          if l-inven-type >= p-inven-type
            and l-inven-type <= p-inven-type1
```

```
            then do:
              l-match = yes.
              leave inven-type-loop.
            end.
          end.

          if l-match = no
          then do:
            hINQuery:get-next().
            /* ALL INVENTORY TYPES ARE OUTSIDE SELECTION RANGE.
*/
            /* BYPASS THIS ITEM. */
            next item-master.
          end.

        end. /* if num-entries ... */
        else do:

          l-inven-type = "".
          if l-inven-type >= p-inven-type
            and l-inven-type <= p-inven-type1
          then .
          else do:
            hINQuery:get-next().
            /* BYPASS THIS ITEM. */
            next item-master.
          end.
```

```
end. /* num-entries = 0 */

if p-conflicts-only = no
  or num-entries(l-inven-types, ";") > 1
then do:

  /* USE INVENTORY TYPE (FIRST ENTRY). */
  l-inven-type = entry(1,l-inven-types, ";").

  unique_id = unique_id + 1.
  create tt_report_results.
  assign
    tt_report_results.unique_id = unique_id
    tt_report_results.in_part        = pt_mstr.pt_part
    tt_report_results.pt_desc1       = pt_mstr.pt_desc1
    tt_report_results.in_site        = in_mstr.in_site
    tt_report_results.tt_inven_type  = l-inven-type
    tt_report_results.pt_prod_line   = pt_mstr.pt_prod_line
    tt_report_results.pt_part_type   = pt_mstr.pt_part_type
    tt_report_results.pt_group       = pt_mstr.pt_group
    tt_report_results.comd_comm_code =
              comd_det.comd_comm_code when(available comd_det)
    tt_report_results.pti_abc        = (if available pti_det
      then pti_det.pti_abc else pt_mstr.pt_abc)
    tt_report_results.pt_pm_code     = (if available ptp_det
```

```
    then ptp_det.ptp_pm_code else pt_mstr.pt_pm_code)
  tt_report_results.pt_buyer       = (if available ptp_det
    then ptp_det.ptp_buyer else pt_mstr.pt_buyer)
  tt_report_results.tt_conflict    =
    "*" when(num-entries(l-inven-types, ";") > 1).

if p-show-logic
then do:
  run display-logical-expression
    (input l-inven-type, input unique_id).
end.

if num-entries(l-inven-types, ";") > 1
then do:

  do i2 = 2 to num-entries(l-inven-types, ";"):
    l-inven-type = entry(i2,l-inven-types, ";").

    unique_id = unique_id + 1.
    create tt_report_results.
    assign
      tt_report_results.unique_id     = unique_id
      tt_report_results.tt_inven_type = l-inven-type
      tt_report_results.tt_conflict   = "*".
    if p-show-logic
    then do:
      run display-logical-expression
```

```
                (input l-inven-type, input unique_id).
              end.
            end. /* do i2 */
          end. /* if num-entries */
        end. /* if l-conflicts-only = no ... */
      end.
      hINQuery:get-next().
    end. /* Repeat query */

    hINQuery:query-close().
    end.
    delete object hINQuery.

    phReportResults = dataset dsReportResults:handle.
    delete object phReportResults no-error.

END PROCEDURE.

{{&US_GP}gpdelp.i "icobxr" "p"}
{{&US_GP}gpdelp.i "ppccxr1" "p"}

PROCEDURE display-logical-expression:

    define input parameter p-inven-type like iat_inven_type no-undo.
    define input parameter unique_id    as   integer        no-undo.
    /* VARIABLES FOR BUILD-LOGICAL-EXPRESSION */
    define variable l-current-line      as   integer        no-undo.
```

```
  define variable l-total-lines       as   integer        no-undo.


  /* BUILD WTLOGIC WITH A LOGICAL EXPRESSION FROM
INVENTORY TYPE DEFINITION */

  /* DO NOT LOOK FOR THE CURRENT LINE (USE 3 ZERO
PARAMETERS). */

  {{&US_GP}gprunp.i "icobxr" "p" "build-logical-expression"

    "(input p-inven-type,

      input 0,

      input 0,

      input 0,

      output l-current-line,

      output l-total-lines)"}


  /* DISPLAY LOGICAL EXPRESSION */

  for each wtlogic

  exclusive-lock:

    create tt_logical_expression.

    assign

      tt_logical_expression.ref_id        = unique_id

      tt_logical_expression.tt_logic_line = wt-logic-line.

    delete wtlogic.

  end.


END PROCEDURE.


&IF DEFINED(supportsAttachments1) &THEN

  {com/qad/shell/report/AddAttachedFiles.i  &FIELDNAME="in_site"}
```

&ENDIF

Dataset Reporting Utility

```
/* ReportHelper.p - Utility procedure to enable Dataset reporting           */
/* Copyright 1986-2008 QAD Inc., Carpinteria, CA, USA.                      */
/* All rights reserved worldwide.  This is an unpublished work.             */
/*                                                                          */
/* Revision: 9008      BY: Robert Jensen DATE: 03/18/2008  ECO: *....*      */
/* Revision: 12936     BY: Evan Bishop   DATE: 01/27/2009  ECO: *....*      */
/* Revision: 14607     BY: Robert Jensen DATE: 03/20/2009  ECO: *....*      */
/* $Revision: 14607$   BY: Evan Bishop   DATE: 01/27/2009  ECO: *....*      */
/*-Revision end----------------------------------------------------------*/

/* Id: ReportHelper.p 12936 2008-10-22 21:12:35Z czs  */

/
*=========================================================================*/

/* ***************************  TEMP TABLES
 ****************************** */

/
*=========================================================================*/

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

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

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

/
*=========================================================================*/

/* ***************************  PARAMETERS
 ****************************** */
```

```
/
*============================================================================*/


/
*============================================================================*/

/* ************************** GLOBAL VARIABLES
 ************************** */

/
*============================================================================*/

{mfsubdirs.i}

{{&US_BBI}mfdeclre.i}

{{&US_BBI}gplabel.i}

{{&US_BBI}pxmsglib.i}


/
*============================================================================*/

/* ***************************** Functions
 ***************************** */

/
*============================================================================*/


FUNCTION getLabel returns character (

  input pTerm as character):


  return getTermLabel(pTerm,1000).

END FUNCTION. /*getLabel*/


FUNCTION addFilterLevel returns character (

  input pCondition1 as character,

  input pCondition2 as character,
```

```
  input pOperator as character):

  /* Include an "AND" condition inside an "OR" condition */

  if (length(trim(pCondition1)) > 0) then do:
    if (length(trim(pCondition2)) > 0 ) then
      return pCondition1 + " " + pOperator + " ( " + pCondition2 + " )".
    else
      return pCondition1.
  end.
  else do:
    if (length(trim(pCondition2)) > 0 ) then
      return  "( " + pCondition2 + " )".
    else
      return "".
  end.

END FUNCTION. /* addFilterLevel */

FUNCTION escapeFilterValue returns character (
  input pDataType as character,
  input pValue as character,
  input pOperator as character):

  /**
  * Add single quotes to filter values of "character" or "date" data types if
```

```
* needed.
**/

/**
* Check for unknown value, which is legal in the case of IsNull and
* IsNotNull operators.  Just return a ? char without putting any single
* quotes around it.
**/
if pValue = ? then
  return "?".

/* Preserve leading spaces on character variables */
if ( pDataType = "character" ) then
  pValue = right-trim(pValue).
else
  pValue = trim(pValue).

if ( pDataType = "character" or pDataType = "date" ) then do:
  pValue = replace(pValue,"~~","~~~~").
  pValue = replace(pValue,"~.","~~~.").
  pValue = replace(pValue,"~'","~~~'").
  pValue = replace(pValue,"~'","~~~'").
  pValue = replace(pValue,"~{","~~~{").
  pValue = replace(pValue,"~\","~~~\").

  /* Matches requires an extra escape */
```

```
    if pOperator = "matches" then do:

      pValue = replace(pValue,"~~","~~~~").

      pValue = replace(pValue,"~.","~~~.").

      pValue = replace(pValue,"~"","~~~"").

      pValue = replace(pValue,"~'","~~~'").

      pValue = replace(pValue,"~{","~~~{").

      pValue = replace(pValue,"~\","~~~\").

    end.


    pValue = """ + pValue + """.

  end.

  else if ( pDataType = "decimal" ) then do:

    /* Do not let the value end with '.' or ','.  The query engine */

    /* does not handle it. */

    if substring(pValue, length(pValue), 1) = "." or substring(pValue,
length(pValue), 1) = "," then

      pValue = pValue + "0".

  end.


  return pValue.


END FUNCTION. /*escapeFilterValue */


FUNCTION appendFilterData returns character (input ipBufferName as
 character):

  /* Append filter data for the input table name */
```

```
define variable userFilter as character no-undo.
define variable userFilter1 as character no-undo.
define variable userFilter2 as character no-undo.
define variable iSeq1 as integer  no-undo.
define variable iSeq2 as integer  no-undo.
define variable fieldDataType as character no-undo.

assign userFilter = ""
     userFilter1 = ""
     userFilter2 = ""
          iSeq1 = -1
          iSeq2 = -1.

/** Filter conditions nesting precedence is
   1) "AND"
   2) "OR"
   3) "AND"

   For example:
     ((c1 or c2) and (c3) and ((c4 and c5) or c6))

   The combination (c1 or c2) is typically the same variable,
   pt_status = "OPEN" or pt_status = "HOLD"

   The combination (c4 and c5) is typically a range, ie
```

```
   pt_part >= "a" and pt_part <= "c"
**/

for each ttFilterCondition no-lock where
ttFilterCondition.bufferName = ipBufferName:

  if ttFilterCondition.filterSequence2 <> iSeq2 then do:
    userFilter1 = addFilterLevel(userFilter1,userFilter2,"or").

    assign
    iSeq2 = ttFilterCondition.filterSequence2
    userFilter2 = ""
    .
  end.

  if ttFilterCondition.filterSequence1 <> iSeq1 then do:
    if userFilter2 > "" then do:
      userFilter1 = addFilterLevel(userFilter1,userFilter2,"or").
    end.

    userFilter = addFilterLevel(userFilter,userFilter1,"and").

    assign
    iSeq1 = ttFilterCondition.filterSequence1
    iSeq2 = ttFilterCondition.filterSequence2
```

```
    userFilter1 = ""
    userFilter2 = ""
    .
  end.

  if userFilter2 > "" then do:
    userFilter2 = userFilter2 + " and ".
  end.

  fieldDataType = "character".

  for first ttReportFieldMeta no-lock where
  ttReportFieldMeta.bufferName = ipBufferName and
  ttReportFieldMeta.fieldName = ttFilterCondition.bufferField:
    fieldDataType = ttReportFieldMeta.dataType.
  end.

  userFilter2 = userFilter2
              + ttFilterCondition.bufferField
              + " " + ttFilterCondition.filterOperator + " "
              + escapeFilterValue(fieldDataType, ttFilterCondition.filterValue,
                                  ttFilterCondition.filterOperator).
end.

/* Get the last ones .. */
userFilter1 = addFilterLevel(userFilter1,userFilter2,"or").
```

```
  userFilter = addFilterLevel(userFilter,userFilter1,"and").

  if userFilter > "" then userFilter = " ( " + userFilter + " )".

  return userFilter.

END FUNCTION. /* appendFilterData */

FUNCTION appendFilterDataForVariable returns character (input ipBufferName
 as character, input ipFieldName as character):

  /* Append filter data for the input table name */

  define variable userFilter as character no-undo.

  define variable userFilter1 as character no-undo.

  define variable userFilter2 as character no-undo.

  define variable iSeq1 as integer  no-undo.

  define variable iSeq2 as integer  no-undo.

  define variable fieldDataType as character no-undo.

  define variable queryFieldName as character no-undo.

  queryFieldName = ipFieldName.

  if(num-entries(ipFieldName) = 2) then

    assign

      queryFieldName = entry(2,ipFieldName)

      ipFieldName = entry(1,ipFieldName).

  assign userFilter = ""
```

```
      userFilter1 = ""
      userFilter2 = ""
          iSeq1 = -1
          iSeq2 = -1.

/** Filter conditions nesting precedence is
    1) "AND"
    2) "OR"
    3) "AND"

    For example:
      ((c1 or c2) and (c3) and ((c4 and c5) or c6))

    The combination (c1 or c2) is typically the same variable,
    pt_status = "OPEN" or pt_status = "HOLD"

    The combination (c4 and c5) is typically a range, ie
    pt_part >= "a" and pt_part <= "c"
**/

for each ttFilterCondition no-lock
where ttFilterCondition.bufferName = ipBufferName
  and ttFilterCondition.bufferField = ipFieldName:

   if ttFilterCondition.filterSequence2 <> iSeq2 then do:
```

```
  userFilter1 = addFilterLevel(userFilter1,userFilter2,"or").

  assign
  iSeq2 = ttFilterCondition.filterSequence2
  userFilter2 = ""
  .
end.

if ttFilterCondition.filterSequence1 <> iSeq1 then do:
  if userFilter2 > "" then do:
    userFilter1 = addFilterLevel(userFilter1,userFilter2,"or").
  end.

  userFilter = addFilterLevel(userFilter,userFilter1,"and").

  assign
  iSeq1 = ttFilterCondition.filterSequence1
  iSeq2 = ttFilterCondition.filterSequence2
  userFilter1 = ""
  userFilter2 = ""
  .
end.

if userFilter2 > "" then do:
  userFilter2 = userFilter2 + " and ".
```

```
    end.

    fieldDataType = "character".

    for first ttReportFieldMeta no-lock where
    ttReportFieldMeta.bufferName = ipBufferName
    and ttReportFieldMeta.fieldName = ttFilterCondition.bufferField:
      fieldDataType = ttReportFieldMeta.dataType.
    end.

    userFilter2 = userFilter2
                + queryFieldName
                + " " + ttFilterCondition.filterOperator + " "
                + escapeFilterValue(fieldDataType, ttFilterCondition.filterValue,
                                    ttFilterCondition.filterOperator).
  end.

  /* Get the last ones .. */
  userFilter1 = addFilterLevel(userFilter1,userFilter2,"or").
  userFilter = addFilterLevel(userFilter,userFilter1,"and").

  if userFilter > "" then userFilter = " ( " + userFilter + " )".

  return userFilter.

END FUNCTION. /* appendFilterDataForVariable */
```

```
function IsTrueValue returns logical (

  input ipBufferName as character, input ipFieldName as character, input
defaultValue as logical):



  /* True if any "= true" or "!= false" records sent for variable */



  define variable resultValue as logical no-undo.

  define variable falseValueSet as logical no-undo.



  resultValue = defaultValue.

  falseValueSet = false.



  /* One true value will return true */

  for each ttFilterCondition no-lock

  where ttFilterCondition.bufferName = ipBufferName

  and ttFilterCondition.bufferField = ipFieldName:

    if (ttFilterCondition.filterOperator = "=" and ttFilterCondition.filterValue =
"true")

    or (ttFilterCondition.filterOperator = "<>" and ttFilterCondition.filterValue =
"false")

    then return true.

    else

    if (ttFilterCondition.filterOperator = "=" and ttFilterCondition.filterValue =
"false")

    or (ttFilterCondition.filterOperator = "<>" and ttFilterCondition.filterValue =
"true")
```

```
      then falseValueSet = true.
    end.

    /* If not true value, return false if any false present */
    if falseValueSet then return false.

    /* Nothing is said, return default value */
    return defaultValue.

END FUNCTION. /* IsTrueValue */

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

    /* Get a single (really just the first) filter value
       entered by the user for a filter field */

    define variable resultValue as character no-undo.

    resultValue = defaultValue.

    for first ttFilterCondition no-lock where
    ttFilterCondition.bufferName = ipBufferName
    and ttFilterCondition.bufferField = ipFieldName:
```

```
    resultValue = ttFilterCondition.filterValue.
  end.

  return resultValue.

END FUNCTION. /* GetFilterValue */

/*
 * Calls the messaging infrastructure directly so that hard-coded text
 * can be used to display messages instead of a message number
 */
FUNCTION DisplayTextMessage returns character
  (input pText as character,
   input pSeverity as integer):

  run setMessage in dynamic-function('getGlobalHandle' in
global_gblmgr_handle, input {&MESSAGE-HANDLE-ID})
    (
      input "",
      input "",
      input 0,
      input pSeverity,
      input pText).
END FUNCTION. /*DisplayTextMessage*/

/*
 * Gets the value of a parameter from the input parameter table
```

```
 * ttParameterValues. If no parameter exists then it returns ?
 */
FUNCTION GetParameterValue returns character
  (input pParameterName as character):

  for first ttParameterValues no-lock where
    ttParameterValues.parameterName = pParameterName:

    return ttParameterValues.parameterValue.
  end.

  return ?.
END FUNCTION. /*GetParameterValue*/

/*
 * Gets the value of a the sys_report_code parameter from the
 * input parameter table ttParameterValues. If this parameter
 * does not exist then it returns ?
 */
FUNCTION GetReportCode returns character():
  return GetParameterValue("sys_report_code").
END FUNCTION. /*GetReportCode*/

/*
 * Gets the value of a the sys_report_definition parameter from the
 * input parameter table ttParameterValues. If this parameter
```

```
 * does not exist then it returns ?
 */
FUNCTION GetReportDefinition returns character():
  return GetParameterValue("sys_report_definition").
END FUNCTION. /*GetReportDefinition*/

PROCEDURE addFilterValuePrefix:
  define input parameter ipBufferName as character no-undo.
  define input parameter ipFieldName as character no-undo.
  define input parameter prefixValue as character no-undo.

  for each ttFilterCondition where
  ttFilterCondition.bufferName = ipBufferName
  and ttFilterCondition.bufferField = ipFieldName:
    ttFilterCondition.filterValue = prefixValue + ttFilterCondition.filterValue.
  end.

END PROCEDURE. /* addFilterValuePrefix */

/
*============================================================================*/

/* ************************  Main Routine
 ************************** */

/
*============================================================================*/

define input parameter dataset for dsReportRequest.
```

```
define variable phMetaResults as handle.
define variable reportName as character no-undo.
define variable fieldIndex as integer no-undo.

run ClearTables.

phMetaResults = dataset dsReportMetaResponse:handle.

for first ttReportRequest no-lock:
  reportName = ttReportRequest.reportProgramName.
  create ttDataObjectMeta.
  assign
  ttDataObjectMeta.dataObjectName = reportName
  ttDataObjectMeta.objectLabel = "Object Label"
  ttDataObjectMeta.defaultSort = ""
  .
end.

/* Helper procedures */
PROCEDURE CreateBufferHeader:
  define input parameter tableName as character.
  define input parameter tableLabel as character.

  create ttDataObjectBufferMeta.
  assign
  ttDataObjectBufferMeta.dataObjectName = reportName
  ttDataObjectBufferMeta.bufferName = tableName
```

```
  ttDataObjectBufferMeta.bufferLabel = bufferLabel
  .
END PROCEDURE. /* CreateBufferHeader */

PROCEDURE CreateField:
  define input parameter bufferName as character.
  define input parameter fieldName as character.
  define input parameter fieldLabel as character.
  define input parameter dataType as character.
  define input parameter fieldFormat as character.
  define input parameter lookupName as character.
  define input parameter isSearchField as logical.
  define input parameter isReadOnlySearch as logical.
  define input parameter isVisible as logical.
  define input parameter isSingleEntry as logical.
  define input parameter isOperatorChangeable as logical.
  define input parameter isRequiredCondition as logical.
  define input parameter isEditable as logical.
  define input parameter defaultValue as character.
  define input parameter defaultOperator as character.
  define input parameter defaultValueType as character.
  define input parameter defaultValue2 as character.
  define input parameter defaultValueType2 as character.

  define variable supportedDataTypes as character
```

```
    initial "CHARACTER,DATE,DECIMAL,INTEGER,LOGICAL,INT64" no-
undo.

  define variable tempDataType as character no-undo.


  supportedDataTypes = supportedDataTypes

    + ",BLOB,CLOB,COM-HANDLE,DATETIME,DATETIME-TZ,HANDLE"

    + ",LONGCHAR,MEMPTR,RAW,RECID,ROWID,WIDGET-HANDLE".

  if lookup(datatype,supportedDataTypes) = 0

  and datatype <> ? then do:

    tempDataType = keyword-all(datatype).

    if lookup (tempDataType,supportedDataTypes) <> 0

    and tempDataType <> ?

    then do:

      datatype = tempDataType.

    end.

    else do:

      /* Create entry in the log file */

      message "ERROR: Unknown data type. Field:" fieldName

        "(datatype:" datatype ") Report:" reportName.

      return.

    end.

  end.


  if fieldLabel = ? then fieldLabel = fieldName.


  if lookupName = "" then do:
```

```
  run findFlhMstr(input reportName, input fieldName,
                  output lookupName).
end.

create ttReportFieldMeta.
assign
ttReportFieldMeta.dataObjectName = ttReportRequest.reportProgramName
ttReportFieldMeta.bufferName = bufferName
ttReportFieldMeta.fieldName = fieldName
ttReportFieldMeta.fieldLabel =  fieldLabel
ttReportFieldMeta.dataType = dataType
ttReportFieldMeta.dataFormat =  fieldFormat
ttReportFieldMeta.valueList = ""
ttReportFieldMeta.searchOperatorList = ""
ttReportFieldMeta.summaryType = ""
ttReportFieldMeta.lookupName = lookupName
ttReportFieldMeta.fieldOrder = fieldIndex
ttReportFieldMeta.isSearchField = isSearchField
ttReportFieldMeta.isReadOnlySearch = isReadOnlySearch
ttReportFieldMeta.isIndexed = false
ttReportFieldMeta.isDefault = false
ttReportFieldMeta.isVisible = isVisible
ttReportFieldMeta.isOID = false
ttReportFieldMeta.isROWID = false
ttReportFieldMeta.isSingleEntry = isSingleEntry
```

```
    ttReportFieldMeta.isOperatorChangeable = isOperatorChangeable
    ttReportFieldMeta.isRequiredCondition = isRequiredCondition
    ttReportFieldMeta.isEditable = isEditable
    ttReportFieldMeta.defaultValue = defaultValue
    ttReportFieldMeta.defaultOperator = defaultOperator
    ttReportFieldMeta.defaultValueType = defaultValueType
    ttReportFieldMeta.secondDefaultValue = defaultValue2
    ttReportFieldMeta.secondDefaultValueType = defaultValueType2
    .

    fieldIndex = fieldIndex + 1.

END PROCEDURE. /* CreateField */

PROCEDURE SetFieldMetaParameter :
/*-----------------------------------------------------------------------------
    Purpose: Sets the value of the specified field parameter in the meta data.
    Notes: Useful when the field's metadata record exists already, but some
           parameters need to be added or modified.
-----------------------------------------------------------------------------*/
    define input parameter bufferName    as character.
    define input parameter fieldName     as character.
    define input parameter parameterName as character.
    define input parameter fieldValue    as character.

    define variable h_buff as handle no-undo.
```

```
  h_buff = buffer ttReportFieldMeta:handle .


  h_buff:find-first("where ttReportFieldMeta.bufferName = '" + bufferName + "'
and ttReportFieldMeta.fieldName = '" + fieldName + "'").


  if h_buff:available then do:

    h_buff:buffer-field(parameterName):buffer-value = fieldValue.

  end.


END PROCEDURE.


PROCEDURE CreateFieldForDBField:


  define input parameter bufferName as character.

  define input parameter tableName as character.

  define input parameter fieldName as character.

  define input parameter isSearchField as logical.

  define input parameter isReadOnlySearch as logical.

  define input parameter isVisible as logical.

  define input parameter isSingleEntry as logical.

  define input parameter isOperatorChangeable as logical.

  define input parameter isRequiredCondition as logical.

  define input parameter isEditable as logical.

  define input parameter defaultValue as character.

  define input parameter defaultOperator as character.
```

```
define input parameter defaultValueType as character.
define input parameter defaultValue2 as character.
define input parameter defaultValueType2 as character.



define variable lookupName as character no-undo.
{{&US_BBI}gpfieldv.i}

field_found = false.

{{&US_BBI}gpfilfld.i
&field_name=fieldName
&field_table=tableName
&GPFILFLD-USE-VAR=yes
}

if field_found then do:
  run findFlhMstr(input reportName, input fieldName,
                  output lookupName).

  run CreateField(input bufferName,
                  input fieldName,
                  input field_label,
                  input field_data-type,
                  input field_format,
```

```
                input lookupName,
                input isSearchField,
                input isReadOnlySearch,
                input isVisible,
                input isSingleEntry,
  input isOperatorChangeable,
  input isRequiredCondition,
  input isEditable,
  input defaultValue,
  input defaultOperator,
  input defaultValueType,
  input defaultValue2,
  input defaultValueType2
                ).
end.
else do:
  message "DB Field not found " fieldName.
  run CreateField(input bufferName,
                input fieldName,
                input fieldName,
                input "character",
                input "x(8)",
                input "",
                input isSearchField,
```

```
                input isReadOnlySearch,
                input isVisible,
                input isSingleEntry,
    input isOperatorChangeable,
    input isRequiredCondition,
    input isEditable,
    input defaultValue,
    input defaultOperator,
    input defaultValueType,
    input defaultValue2,
    input defaultValueType2
                ).
  end.

END PROCEDURE. /* CreateFieldForDBField */

PROCEDURE CreateFieldLikeDBField:

  define input parameter bufferName as character.
  define input parameter fName as character.
  define input parameter tableName as character.
  define input parameter fieldName as character.
  define input parameter isSearchField as logical.
  define input parameter isReadOnlySearch as logical.
  define input parameter isVisible as logical.
  define input parameter isSingleEntry as logical.
```

```
define input parameter isOperatorChangeable as logical.
define input parameter isRequiredCondition as logical.
define input parameter isEditable as logical.
define input parameter defaultValue as character.
define input parameter defaultOperator as character.
define input parameter defaultValueType as character.
define input parameter defaultValue2 as character.
define input parameter defaultValueType2 as character.

define variable lookupName as character no-undo.

{{&US_BBI}gpfieldv.i}

field_found = false.

{{&US_BBI}gpfilfld.i
&field_name=fieldName
&field_table=tableName
&GPFILFLD-USE-VAR=yes
}

if field_found then do:

  run findFlhMstr(input reportName, input fieldName,
                  output lookupName).
```

```
  run CreateField(input bufferName,
        input fName,
        input field_label,
        input field_data-type,
        input field_format,
        input lookupName,
        input isSearchField,
        input isReadOnlySearch,
        input isVisible,
        input isSingleEntry,
  input isOperatorChangeable,
  input isRequiredCondition,
  input isEditable,
  input defaultValue,
  input defaultOperator,
  input defaultValueType,
  input defaultValue2,
  input defaultValueType2
        ).
end.
else do:
  message "DB Field not found " fieldName.
  run CreateField(input bufferName,
        input fName,
        input fName,
```

```
                input "character",
                input "x(8)",
                input "",
                input isSearchField,
                input isReadOnlySearch,
                input isVisible,
                input isSingleEntry,
            input isOperatorChangeable,
    input isRequiredCondition,
    input isEditable,
    input defaultValue,
    input defaultOperator,
    input defaultValueType,
    input defaultValue2,
    input defaultValueType2
                ).
  end.

END PROCEDURE. /* CreateFieldLikeDBField */

PROCEDURE ClearTables:
  assign
  reportName = ""
  fieldIndex = 0
  .
```

```
    empty temp-table ttDataObjectMeta no-error.
    empty temp-table ttDataObjectBufferMeta no-error.
    empty temp-table ttReportFieldMeta no-error.
    empty temp-table ttDataObjectDrillMeta no-error.
    empty temp-table ttDataObjectUrlMeta no-error.

END PROCEDURE. /* ClearTables */

PROCEDURE GetMetaDataResultSet:

    define output parameter dataset for dsReportMetaResponse.

    phMetaResults = dataset dsReportMetaResponse:handle.

END PROCEDURE. /* GetMetaDataResultSet */

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

    define variable queryFragment as character no-undo.

    queryFragment = appendFilterDataForVariable(bufferName, fieldName).

    if queryFragment > "" then
      queryString = queryString + " and " + queryFragment.
```

```
END PROCEDURE. /* FillQueryStringVariable */

PROCEDURE FillQueryString:
  define input parameter bufferName as character no-undo.
  define input-output parameter queryString as character no-undo.

  define variable queryFragment as character no-undo.

  queryFragment = appendFilterData(bufferName).

  if queryFragment > "" then
    queryString = queryString + " and " + queryFragment.
END PROCEDURE. /* FillQueryString */

PROCEDURE findFlhMstr:
 /* -----------------------------------------------------------
  Purpose      : Locate the best flh_mstr record for the current
                 field name and executable name.
  Parameters   : p-execname - the menu entry executable name
                 p-fld-name - the current field
                 p-flh-exec - flh_exec from found record
  Exceptions   :
  Notes        :
 -----------------------------------------------------------*/

  /* This procedure is Desktop lookups copied from mfww01a.p */
```

```
define input  parameter p-execname as character.
define input  parameter p-fld-name as character.
define output parameter p-flh-exec as character.
/* Define buffer to ensure the scope of the table is to this */
/* internal procedure.                                       */

define buffer b-flh_mstr for flh_mstr.
define variable desktopLookUp as logical no-undo.
define variable lookUpName as character no-undo.
define variable c-execname as character no-undo.
define variable tmp-program as character no-undo.

assign c-execname = substring(p-execname,1,r-index(p-execname,".") - 1 ).
assign c-execname = substring(c-execname,r-index(c-execname,"~/") + 1).

/* First, try to find an exact match with no extension*/
for first b-flh_mstr fields(flh_field flh_call_pgm flh_exec)
  where flh_field = p-fld-name and
  entry(1,flh_call_pgm,".") = c-execname
  no-lock:
end.

/* If not found, try to find a wild-card match on pgm name */
if not available b-flh_mstr then
  find last b-flh_mstr
  where flh_field = p-fld-name and
```

```
    c-execname matches entry(1,flh_call_pgm,".")
    no-lock no-error.

  /* If still not found, try to find a blank call pgm */
  if not available b-flh_mstr then
  for first b-flh_mstr fields(flh_field flh_call_pgm flh_exec)
    where flh_field = p-fld-name and
    flh_call_pgm = "" no-lock:
  end.

  assign p-flh-exec = "".

  if available b-flh_mstr and flh_exec <> "" and flh_exec <> "swnull.p":U
  then do:
    /* return the lookup only if it is web enabled */
    tmp-program = entry(1,flh_exec,".").

    for first pgmi_mstr fields (pgmi_exec pgmi_web_implemented
pgmi__qadc02)
    where pgmi_exec = tmp-program
    no-lock:
      desktopLookUp = no.
      if pgmi_web_implemented or pgmi__qadc02 <> "" then desktopLookUp =
yes.
      else do:
        /* Name is 'gplu002'.  Need 'gp002' for browse master lookup. */
        lookUpName = substring(pgmi_exec,1,2) + substring(pgmi_exec,5).
```

```
          for first brw_mstr where brw_name = lookUpName and (brw_filter > "")
 no-lock:

            desktopLookUp = yes.

          end.

        end.

        if desktopLookUp then do:

          assign p-flh-exec = if pgmi__qadc02 <> "" then pgmi__qadc02

                              else flh_exec.

        end.

      end.

    end.


END PROCEDURE. /* find-flh-mstr */



/
*============================================================================*/

/* ***************************  MetaDataHelper
 ***************************** */

/
*============================================================================*/


/*

 * Temp-tables and dataset to define meta-data information

 */

define temp-table ttField

   field BufferName       as character

   field FieldName        as character
```

```
field FieldSequence    as integer
field FieldLabel       as character initial ""
field IsTranslated as logical initial false
field DBTableName as character initial ""
field DBFieldName        as character initial ""
field DBFieldExtent  as integer initial 0
field IsSearchField    as logical initial false
field ValueList as character initial ""
field IsReadOnlySearch as logical initial false
field IsVisible        as logical initial true
field IsSingleEntry  as logical initial false
field IsOperatorChangeable as logical initial true
field IsRequiredCondition  as logical initial false
field IsEditable           as logical initial true
field DefaultValue1         as character initial ""
/* "constant", "method" or "session" */
field DefaultValue1Type     as character initial "constant"
/* "equals", "begins", "between", "greaterthan", "greaterthanequals",
 * "in", "lessthan", lessthanequals", "matches", "notallowed",
 * "notequals", "notin" */
field DefaultOperator      as character initial "equals"
field DefaultValue2        as character initial ""
field DefaultValue2Type    as character initial "constant"
index ByBufferByField is unique primary BufferName FieldName
```

```
  index ByBufferBySequence BufferName FieldSequence.

define dataset dsReportMetaData for ttField.
define variable translateTerm as logical no-undo initial true.

/*
 * Used to set whether display values are to be treated as
 * pre-translated values or as untranslated terms.  A value
 * of false will treat values as terms and attempt to convert
 * them, whereas a value of true will bypass this conversion
 * (since the value is already translated)
 */
procedure SetIsTranslated:
  define input parameter ilIsTranslated as logical no-undo.

  translateTerm = not ilIsTranslated.
end procedure.

/* Metadata definition block */
procedure FillMetaData:
  define input parameter ihDS as handle no-undo.
  define input parameter icMetaDataFile as character no-undo.

  run LoadMetaData(icMetaDataFile).
  run CreateMetaDataForDS(ihDS).
  run ProcessMetaData.
```

```
end procedure.

define variable transYes as character no-undo initial ?.

define variable transNo as character no-undo initial ?.

define variable logicalFormat as character no-undo.

function GetFormatForDataType returns character(input icDataType as
 character):

  define variable vcFormat as character no-undo initial "x(20)".


  if(transYes = ?) then do:

    transYes = getTermLabel("YES",10).

    transNo = getTermLabel("NO",10).

    logicalFormat = transYes + "/" + transNo.

  end.

  if(icDataType = "date") then vcFormat = "99/99/99".

  if(icDataType = "decimal") then vcFormat = "->,>>>,>>>,>>>,>>9.99".

  if(icDataType = "logical") then vcFormat = if(translateTerm) then transYes +
 "/" + transNo else "yes/no".

  if(icDataType = "integer") then vcFormat = "->,>>>,>>>,>>>,>>9".


  return vcFormat.

end function.

function ConvertDateToString returns character(input idDate as date):

  define variable vcConvertedDate as character no-undo.


  /* Note that appserver date format will always be ymd */
```

```
  vcConvertedDate = string(idDate,"9999-99-99") no-error.
  return vcConvertedDate.
end function.

function ConvertStringToDate returns date(input icDate as character):
  define variable vcConvertedDate as date no-undo.

  vcConvertedDate = ?.
  vcConvertedDate = date(icDate) no-error.
  return vcConvertedDate.
end function.

procedure SetMetaData:
  define input parameter icBufferName as character no-undo.
  define input parameter icField as character no-undo.
  define input parameter icMetaField as character no-undo.
  define input parameter icMetaValue as character no-undo.

  define variable vcTermValue as character no-undo.
  define variable vcTransValue as character no-undo.
  define variable i as integer no-undo.
  if(translateTerm) then do:
    /* Translate displayed value list */
    if(icMetaField = "ValueList") then do:
      do i=1 to num-entries(icMetaValue) by 2:
        vcTermValue = entry(i,icMetaValue).
```

```
        vcTransValue = getTermLabel(vcTermValue,50).
        entry(i,icMetaValue) = vcTransValue.
      end.
    end.
    /* Translate field label */
    if(icMetaField = "FieldLabel") then do:
      icMetaValue = getTermLabel(icMetaValue,50).
    end.
  end.
  run SetFieldMetaParameter (icBufferName,icField,icMetaField,icMetaValue).
end procedure.

define variable IsMetaDataLoaded as logical no-undo initial false.
procedure LoadMetaData:
  define input parameter vcMetaDataFile as character no-undo.

  define variable vhDS as handle no-undo.
  define variable viNextSequence as integer no-undo.

  vhDS = dataset dsReportMetaData:handle.
  vcMetaDataFile = search("com/qad/shell/report/reports/" + vcMetaDataFile).

  if(not IsMetaDataLoaded) then do:
    empty temp-table ttField.
    vhDS:read-xml("file",vcMetaDataFile,"empty",?,?).
    IsMetaDataLoaded = true.
```

```
    /* Reorder sequences */
    viNextSequence = 10000.
    for each ttField by FieldSequence:
      viNextSequence = viNextSequence + 1.
      ttField.FieldSequence = viNextSequence.
    end.
  end.
end procedure.

procedure ProcessMetaData:

  for each ttField where ttField.IsSearchField:
    translateTerm = not ttField.IsTranslated.
    if(ttField.FieldSequence <> 0) then
      run SetFieldMetaParameter (
 ttField.BufferName,ttField.FieldName,"FieldOrder",string(ttField.FieldSequence)).

    if(ttField.FieldLabel <> "") then
      run SetMetaData(
        ttField.BufferName,ttField.FieldName,"FieldLabel",ttField.FieldLabel).

    run SetFieldMetaParameter (
      ttField.BufferName,ttField.FieldName,"IsSearchField","true").
    if(ttField.ValueList <> "") then
```

```
    run SetMetaData(
      ttField.BufferName,ttField.FieldName,"ValueList",ttField.ValueList).
  if(ttField.IsReadOnlySearch) then
    run SetFieldMetaParameter (
      ttField.BufferName,ttField.FieldName,"IsReadOnlySearch","true").
  if(not ttField.IsVisible) then
    run SetFieldMetaParameter (
      ttField.BufferName,ttField.FieldName,"IsVisible","false").
  if(ttField.IsSingleEntry) then
    run SetFieldMetaParameter (
      ttField.BufferName,ttField.FieldName,"IsSingleEntry","true").
  if(not ttField.IsOperatorChangeable) then
    run SetFieldMetaParameter (
      ttField.BufferName,ttField.FieldName,"IsOperatorChangeable","false").
  if(ttField.IsRequiredCondition) then
    run SetFieldMetaParameter (
      ttField.BufferName,ttField.FieldName,"IsRequiredCondition","true").
  if(not ttField.IsEditable) then
    run SetFieldMetaParameter (
      ttField.BufferName,ttField.FieldName,"IsEditable","false").
  if(ttField.DefaultValue1 <> "") then
    run SetMetaData(
ttField.BufferName,ttField.FieldName,"DefaultValue",ttField.DefaultValue1).
```

```
    if(ttField.DefaultValue1Type <> "") then
      run SetFieldMetaParameter (
  ttField.BufferName,ttField.FieldName,"DefaultValueType",ttField.DefaultValue1Type).
    if(ttField.DefaultOperator <> "") then
      run SetFieldMetaParameter (
  ttField.BufferName,ttField.FieldName,"DefaultOperator",ttField.DefaultOperator).
    if(ttField.DefaultValue2 <> "") then
      run SetMetaData(
  ttField.BufferName,ttField.FieldName,"SecondDefaultValue",ttField.DefaultValue2).
    if(ttField.DefaultValue2Type <> "") then
      run SetFieldMetaParameter (
  ttField.BufferName,ttField.FieldName,"SecondDefaultValueType",ttField.DefaultValue2Type).
   end. /* for each ttField */
 end procedure.
 procedure CreateMetaDataForDS:
   define input parameter phDS as handle no-undo.
   define variable vhBuffer as handle no-undo.
   define variable vhField as handle no-undo.
   define variable i as integer no-undo.
   define variable j as integer no-undo.
```

```
define variable vcBufferName as character no-undo.
define variable vcFieldName as character no-undo.
define variable vcDBFieldName as character no-undo.
define variable vcFieldLabel as character no-undo.
define variable vcDataType as character no-undo.
define variable vcFieldFormat as character no-undo.

do i=1 to phDS:num-buffers:
 vhBuffer = phDS:get-buffer-handle(i).
 if(valid-handle(vhBuffer)) then do:
  vcBufferName = vhBuffer:name.
  run CreateBufferHeader
   (vcBufferName, vcBufferName).
  do j=1 to vhBuffer:num-fields:
   vhField = vhBuffer:buffer-field(j).
   if(valid-handle(vhField)) then do:
     assign
       vcFieldName = vhField:name
       vcDataType = vhField:data-type
       vcFieldFormat = GetFormatForDataType(vcDataType).
     find ttField where
       ttField.BufferName = vcBufferName and
       ttField.FieldName = vcFieldName no-error.

     /* Convert "TODAY" or "NOW" to today's date for date data type fields */
```

```
        if(available(ttField) and vcDataType = "date") then do:

          if(ttField.DefaultValue1 = "today" or ttField.DefaultValue1 = "now")
then

            ttField.DefaultValue1 = ConvertDateToString(today).

          if(ttField.DefaultValue2 = "today" or ttField.DefaultValue2 = "now")
then

            ttField.DefaultValue2 = ConvertDateToString(today).

        end. /* if(available(ttField) and vcDataType = "date") */


        if(available(ttField) and ttField.DBFieldName <> "") then do:


          /* Remove index component from array DB field name */

          vcDBFieldName = ttField.DBFieldName.

          if(index(vcDBFieldName,"[") > 0) then do:

            vcDBFieldName =
substring(vcDBFieldName,1,index(vcDBFieldName,"[") - 1).

          end.


          /* DB Field = Temp-Table Field: CreateFieldForDBField */

          if(vcDBFieldName = ttField.FieldName) then do:

            run CreateFieldForDBField

              (vcBufferName,

               ttField.DBTableName,

               vcFieldName,

               false,                   /* IsSearchable */

               false,                   /* IsReadOnlySearch */
```

```
    true,                   /* IsVisible */
    false,                  /* IsSingleEntry */
    true,                   /* IsOperatorChangeable */
    false,                  /* IsRequiredCondition */
    true,                   /* IsEditable */
    "",                     /* DefaultValue1 */
    {&ParameterOperator_Equals},    /* DefaultOperator */
    {&ParameterValueType_Constant}, /* DefaultValue1Type */
    "",                     /* DefaultValue2 */
    {&ParameterValueType_Constant}  /* DefaultValue2Type */).
end. /* if(vcDBFieldName = ttField.FieldName) */
/* DB Field <> Temp-Table Field: CreateFieldLikeDBField */
else do:
  run CreateFieldLikeDBField
    (vcBufferName,
     vcFieldName,
     ttField.DBTableName,
     vcDBFieldName,
     false,                 /* IsSearchable */
     false,                 /* IsReadOnlySearch */
     true,                  /* IsVisible */
     false,                 /* IsSingleEntry */
     true,                  /* IsOperatorChangeable */
     false,                 /* IsRequiredCondition */
     true,                  /* IsEditable */
```

```
      "",                       /* DefaultValue1 */
      {&ParameterOperator_Equals},     /* DefaultOperator */
      {&ParameterValueType_Constant},  /* DefaultValue1Type */
      "",                       /* DefaultValue2 */
      {&ParameterValueType_Constant}   /* DefaultValue2Type */).
  end. /* else do */
end. /* if(available(ttField) and ttField.DBFieldName <> "") */
/* No DB Field:  CreateField */
else do:
  run CreateField
    (vcBufferName,
     vcFieldName,
     upper(vcFieldName),
     vcDataType,
     vcFieldFormat,
     "",                      /* FieldLookup */
     false,                   /* IsSearchable */
     false,                   /* IsReadOnlySearch */
     true,                    /* IsVisible */
     false,                   /* IsSingleEntry */
     true,                    /* IsOperatorChangeable */
     false,                   /* IsRequiredCondition */
     true,                    /* IsEditable */
     "",                      /* DefaultValue1 */
```

```
            {&ParameterOperator_Equals},     /* DefaultOperator */
            {&ParameterValueType_Constant},  /* DefaultValue1Type */
            "",                              /* DefaultValue2 */
            {&ParameterValueType_Constant}   /* DefaultValue2Type */).
        end. /* else do */
      end. /* if(valid-handle(vhField)) */
    end. /* do j=1 to vhBuffer:num-fields */
  end. /* if(valid-handle(vhBuffer)) */
 end. /* do i=1 to phDS:num-buffers */
end procedure.

procedure AddAllConditions:
 define input parameter icDBTable as character no-undo.
 define input-output parameter bcQueryString as character no-undo.

 run AddSomeConditions(icDBTable,"",input-output bcQueryString).
end procedure.

procedure AddSomeConditions:
 define input parameter icDBTable as character no-undo.
 define input parameter icExclusionList as character no-undo.
 define input-output parameter bcQueryString as character no-undo.

 define variable vcFieldName as character no-undo.
```

```
 for each ttField where ttField.IsSearchField and
(ttField.DBTableName=icDBTable or ttField.BufferName=icDBTable):

  if(not can-do(icExclusionList,ttField.FieldName)) then

    if(ttField.DBFieldExtent > 1) then

      run AddExtentConditions(

        ttField.BufferName,

        ttField.FieldName,

        ttField.DBFieldName,

        ttField.DBFieldExtent,

        input-output bcQueryString).

    else do:

      /* Map field name to DB field name, if DB field provided and not eqult to
field name */

      vcFieldName = ttField.FieldName.

      if(ttField.DBFieldName <> "" and ttField.DBFieldName <> vcFieldName)
then

        vcFieldName = vcFieldName + "," + ttField.DBFieldName.

      run AddCondition(ttField.BufferName,vcFieldName,input-output
bcQueryString).

    end.

 end.

end procedure.


procedure AddSpecificConditions:

 define input parameter icDBTable as character no-undo.

 define input parameter icFieldList as character no-undo.

 define input-output parameter bcQueryString as character no-undo.
```

```
define variable i as integer no-undo.
define variable vcData as character no-undo.
define variable vcFieldData as character no-undo.
define variable entryName as character  no-undo.

do i=1 to num-entries(icFieldList):
 vcData = entry(i,icFieldList).
 entryName = entry(1,vcData,":").
 vcFieldData = entryName.
   for each ttField where ttField.IsSearchField and
                        ttField.FieldName = entryName and
                        (ttField.DBTableName=icDBTable or
ttField.BufferName=icDBTable) no-lock:
      if(num-entries(vcData,":") > 1) then do:
        vcFieldData = ttField.FieldName + "," + entry(2,vcData,":").
      end.
      if(ttField.DBFieldExtent > 1) then
        run AddExtentConditions(
          ttField.BufferName,
          vcFieldData,
          ttField.DBFieldName,
          ttField.DBFieldExtent,
          input-output bcQueryString).
      else
        run AddCondition(ttField.BufferName,vcFieldData,input-output
bcQueryString).
```

```
      end.
    end.
  end procedure.

  procedure AddExtentConditions:
    define input parameter icBufferName as character no-undo.
    define input parameter icFieldName as character no-undo.
    define input parameter icDBFieldName as character no-undo.
    define input parameter iiExtent as integer no-undo.
    define input-output parameter bcQueryString as character no-undo.

    define variable tempQS as character no-undo initial "".
    define variable endParen as character no-undo initial "".
    define variable i as integer no-undo.

    tempQS = "".
    endParen = "".

    run FillQueryStringVariable
      (input icBufferName, input icFieldName + "," + icDBFieldName + "[1]",
     input-output tempQS).

    if tempQS <> "" then
      assign
        endParen = " )".
```

```
    tempQS = replace(tempQS, "and  ( ( (", "and  ( ( ( (").

  bcQueryString = bcQueryString + tempQS.

  do i=2 to iiExtent:

    tempQS = "".

    run FillQueryStringVariable

     (input icBufferName, input icFieldName + "," + icDBFieldName + "[" +
 string(i) + "]", input-output tempQS).

    tempQS = replace(tempQS, "and  ( ( (", "or  ( ( (").

    bcQueryString = bcQueryString + tempQS.

  end.

  bcQueryString = bcQueryString + endParen.

end procedure.

procedure AddCondition:

  define input parameter icBufferName as character no-undo.

  define input parameter icBufferFieldName as character no-undo.

  define input-output parameter bcQuery as character no-undo.

  define variable vcCondition as character no-undo.

  vcCondition = trim(appendFilterDataForVariable
 (icBufferName,icBufferFieldName)).

  if(vcCondition <> ? and vcCondition <> "") then do:
```

```
    bcQuery = bcQuery + " and " + vcCondition.
  end.
end.

delete object phMetaResults no-error.
```

Approval Delegation

Using Approval Delegation, you can delegate approvals to other users. You can define whom you want to delegate approvals to, and under what conditions and when the delegation is effective.

Click the User Options icon on the menu bar and choose Approval Delegation. Set approval delegations in the Approval Delegation Settings window.

Approval Delegation Fields

Main

Delegation Active

Indicate whether to enable the Approval Delegation functionality for you.

Start Date

Specify the start date when the delegation starts to be effective.

End Date

Specify the end date when the delegation stops being effective.

Delegate To Approvers

Approval Type

Specify the approval type for which the delegation is set.

Route Code

Specify the route for which the delegation is set.

Approver

Specify the approver whom you want to delegate the approvals to.

Future Approvals

Using Future Approvals, you can view approval processes that are in progress and require your approval in the future.

Click the User Options icon on the menu bar and choose Future Approvals. The system displays the Future Approvals browse. The Future Approvals browse lists all approvals that are pending for your approval or require your approval in the future, even though they are at an approval step for someone else.

Double-click a record to open the Details screen, and you can view all information of the approval process.

If the Early Approval option is selected in Approval Configuration for this entity, action buttons are displayed at the bottom of the Details screen and you can choose an action. After the approval request is approved or denied, the system closes the pending tasks and sets the status of the approvers who are before you in the sequence to Skipped. The system also sends notifications to the skipped approvers, informing them of the action.

Submitted Background Processing

There are many configuration options for background processing, and the configuration is based on the implementation of each background process type.

Background Processes Submission

There are many types of background processes in the application, such as Confirm Supplier Payment Selections or Run Consistency Check. Different types of background processes have different ways of submission. For example, for the Unconfirm Supplier Payment Selections process, there is a checkbox to indicate this needs to be run as a background process and the process is submitted after the user clicks the Unconfirm button. Note that other background process types may behave differently.

Processing Options

Process in Background

Indicates whether this process is run in the background. You need to define this setting for each action using background processing.

The background processing results will be sent to your Inbox. If the volume of records exceeds the predefined threshold for the action, the checkbox is selected and disabled; the system will enforce background processing.

Note The Processing Options panel is always the last panel on the Actions form.

Background Processing Description

Enter a meaningful description of the background process to distinguish important background processes from the other submissions. The background processing description is used in the Background Processing Notifications.

Background Processing Views

There are three different browses to view background processing.

- Submitted Background Processing — This view is available to all users from the user options drop-down. The Submitted Background Processing view is filtered to show only the background processes submitted by the logged in user. It is used by users to monitor and manage processes that they submitted.
- Background Processing — This view is available to administrators from the Activity folder. The Background Processing view shows all background processing records created in the system. It is used by administrators to monitor and manage background processes.
- Background Processing Details — This view is available to all users, and it is used by users to monitor and manage processes that they

have access to. You can access this view from the drill-down link on the business component or from the link in notifications. The Background Processing Details view is filtered to show only the background processes the user submitted or is granted access to based on the Manage Background Processing security permission to the business component.

Submitted Background Processing

Once you submit the process, you can go to the Submitted Background Processing view and check the status of your process.

Note that the Background Processing Details view looks similar to the Submitted Background Processing view, so the descriptions below apply to both views.

The Main and Summary panels display the information for the most recent submission. If there are multiple submissions for the background processing record, review the Submissions panel to see the previous submission information. Note that the only field you can modify here is Description.

Main

Business Component

The business component related to the background processing record.

Type

The type of background process. This references the action the background processing record is related to (for example, Confirm).

Description

Every background processing has a description, usually specified by system or user when the background processing is submitted. You can modify and enter a description, which is more meaningful.

Link

Link to the source record, from which the background processing is started.

Open

This opens the link to the source record.

Status

The status includes the following options:

-  Queued
-  Processing
-  Successful
-  Completed with Warnings
-  Failed - Application Error
-  Failed - System Error
-  Paused
-  Canceled

Completed

The processing percent complete.

Submitted By

The user ID and user name of a person who made the most recent background processing submission.

Submit Date

The date and time when the most recent background processing submission entered the queue.

Processing Start Date

The date and time when the most recent background processing submission started processing.

Processing End Date

The date and time when the most recent background processing submission finished processing.

Submissions

Submission

The submission data for the background processing record. One background processing record can have multiple submissions. If there are multiple submissions for a single background processing record, you can view them by using a drop-down menu.

Details

This opens the Submission Details browse for viewing further details on the selected record.

Failed

The count of failed records.

Successful

The count of successful records. Warnings are also counted as Successful.

Warnings

The count of records with warning messages.

Processed

The count of total records at the end of processing. Processed = Failed + Successful.

Records with Messages

Only records with messages are listed in the "Records with messages" grid. One record can have multiple messages, and the system only shows the highest severity message in this grid. If you want to see all messages, click the Details button.

The "Records with messages" grid only shows 999 records. If the grid exceeds 999, click the Details button to open the Submission Details browse to access and review the full set of records with messages.

Record

The record processed in background processing submission.

Open Record

This opens the related record in a pop-up window for error correction and reprocessing.

Record Type

The type of record the record ID represents, for example, Supplier Invoice or Sales Order.

Result

The processing result for the record includes the following options:

- Successful — The record is processed successfully.
- Failed — The record processing failed.

Message Count

The number of messages returned after processing.

Message Type

The most critical message type returned for the record during processing. The message type priority logic is:

1. Error
2. Warning
3. Information

Message

The first message returned of the highest priority message type for the record.

Field

The field label of business component field the message is related to.

Field Value

The invalid record value that triggered the message.

Step

The background processing step the record is processed in.

Description

The description of the API called for the step.

Submission Details

Click the Details button on the Submissions panel to open the Submission Details browse, where you can review the full set of records with messages. One record can have multiple messages, and the system only shows the highest severity message in the Submission Details browse. If you want to see all messages, double-click the record.

Some background processing, like Supplier Payment Selections, may not want to return records that are successfully processed; the application only returns records with messages. If only records with messages are available to review in the Submission Details browse, you will see a message about this.

Note You cannot update or delete records in the Submission Details form.

If the record processing failed due to an error, you can open a detailed record by clicking the Open button and view the errors. This way, you can quickly and efficiently correct processing errors and resubmit the action. If an error is still valid, it is displayed in the Errors grid. If an error may not be valid, you will see a message about this. You can also view the messages by clicking the messages icon on the right-hand side of the view.

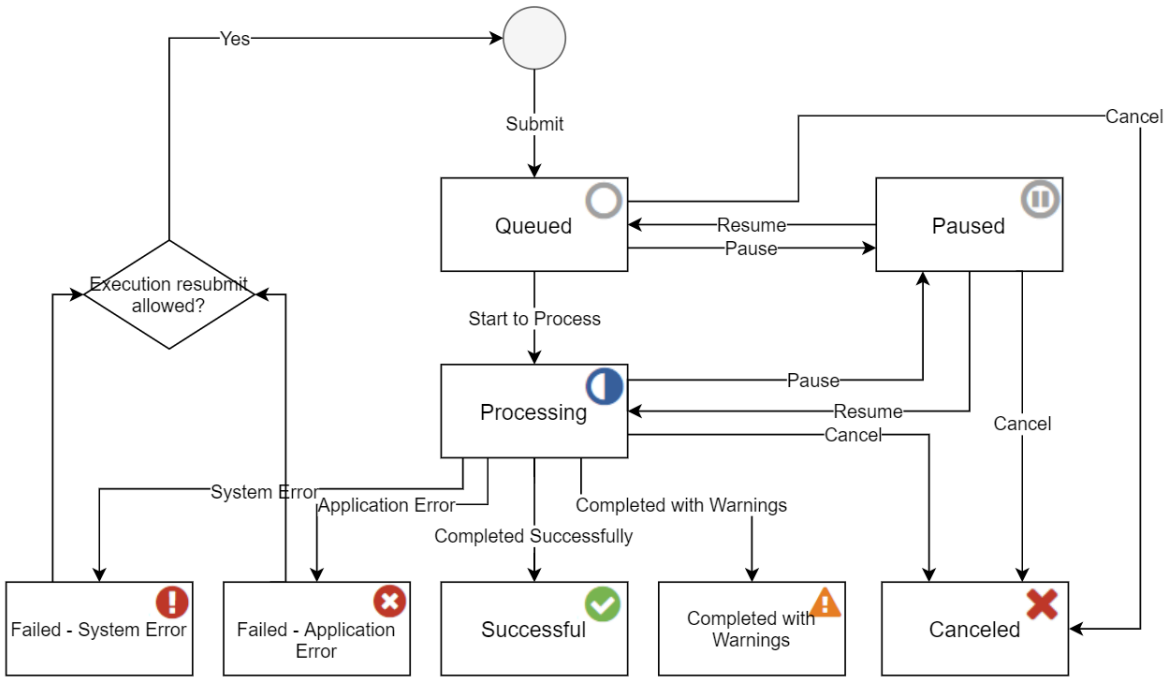
You can export the submission details data to Excel by using the More menu. However, if more than one message is generated for the record, only the first error of the highest priority error type is included in the Excel export. To view all record messages, you can work in the Submission Details form.

Managing Background Processing

As long as you monitor background processes, you can take the following actions from the Actions menu: pause, resume, or cancel the background process. You can only cancel or pause background processes as long as

the background process is not executing the last step. If you try to cancel or pause a background process when the last step is executing, you will see a message about this.

The background processing status diagram below shows under what conditions you can take what actions. If an action is not available, it means you cannot take the action under current conditions.



Actions

Cancel

Terminates the background process and updates the status to Canceled. This action is available when the status is Queued, Paused, or Processing (not executing the last step). When you cancel background processing, it will stop after the current step is completed.

If you cancel the background process, you cannot resume it. You will need to resubmit the action.

Pause

Stops the background process and updates the status to Paused. This action is available when the status is Queued or Processing (not executing the last step). When you pause background processing, it will pause after the current step is completed.

Resume

Updates the status to Queued and processing will begin at the point the background processing is paused. This action is available when

the status is Paused. Use this action when you want to restart the background processing.

Background Processing Notifications

Background processing results are sent to the user who submitted the background process when the background processing completes. According to the user's notification preferences, the system automatically sends notifications to the user's QAD Inbox or Email address.

The notifications include:

- Notification icon for the background processing result
- Subject: "Background Processing:" [Description]
- Submission summary:
  - Failed - [Records Failed]
  - Successful - [Records Successful]
  - Warnings - [Records with Warnings]
  - Processed - [Records Processed]

You can configure background processing notifications in your user profile. There is Background Processing under Category Settings and under Event Settings that you can configure to enable the QAD Inbox or Email notifications for background processing.

Background processing notifications are always sent at the outcome of each process, allowing you easy access to the background processing record and providing the processing summary information in the extended Inbox and Email notifications. You may want to configure your notification preferences differently depending on the background processing outcome. It is likely to be more important to be notified about failed or canceled background processes than it is for the completed ones. To help you better manage the notifications you receive, two background processing notification event settings are available:

- Background Processing Completed/Paused
- Background Processing Failed/Canceled

Notification events for background processing can only be configured at the event level, not at the background processing type level. The following options are available for QAD Inbox and Email:

- All — The notification option for every process type = Yes.
- Configured — It is equal to choosing None. Avoid using Configured because for the background processing events you cannot configure at a lower level.
- None — The notification option for every process type = No.

Most background processing notifications are based on a default notification template, which you can see under Notification Templates on page 7004. Some types of background processing may use their own templates.