QAD .NET User Interface Release Notes

March 2012

These release notes include information about QAD .NET UI 2.9.5 (2012 EE).

Review this document *before* proceeding with any phase of a QAD .NET UI implementation.

These release notes include the following sections:

Installation and Configuration Information

The following summarizes installation and configuration changes for this version of the QAD .NET UI.

This version of the QAD .NET UI is installed with QAD Enterprise Applications – Enterprise Edition 2012 as part of the QDT-based installation process.

Release Summary

QAD .NET UI Version: 2.9.5 (2012 EE)

Product Versions: QAD Enterprise Applications 2012 – Enterprise Edition

Microsoft .NET Framework Version: 3.0

Tomcat Versions: 5.5.*x* and 6.0.*x*. If you are upgrading and have already installed Tomcat 5.5.*x*, you do not need to upgrade to Tomcat 6. Tomcat 5.5.*x* is fully supported for use with QAD .NET UI 2012 EE.

Operating System: The QAD .NET UI client runs on Windows XP, Windows Vista, and Windows 7. It can run on 64-bit Windows, but only in 32-bit mode.

Note If upgrading from a previous version, please be sure to review the release notes for the versions between your current version and this version.

Release History

For information on the QAD .NET UI release history, see the *Platform and Product Availability Guide*, available from the General Reference section of the QAD support site (http://support.qad.com). The following table summarizes the recent QAD .NET UI 2.9.*x* releases.

Version	Date	QAD Product Version
2.9	March 2010	QAD Applications 2010 – Enterprise Edition
2.9.1	June 2010	QAD Applications 2010 – Standard Edition
2.9.2	September 2010	QAD Applications 2010.1– Enterprise Edition
2.9.3.65	March 2011	QAD Enterprise Applications 2011 – Enterprise Edition

Version	Date	QAD Product Version
2.9.4	June 2011	QAD Enterprise Applications 2011 – Standard Edition
2.9.3.79 (2011.1 EE)	September 2011	QAD Enterprise Applications 2011.1 – Enterprise Edition
2.9.5 (2012 EE)	March 2012	QAD Enterprise Applications 2012 – Enterprise Edition

Supported Languages

The user interface supports the following languages in this release:

Chinese (Simplified)	English (US)	Italian	Portuguese (Brazilian)
Chinese (Traditional)	French	Japanese	Spanish (Castilian)
Dutch	German	Polish	Spanish (Latin American)

The following languages have some support, but new terms added in this release may appear in English:

Bulgarian	Greek	Norwegian	Slovenia
Czech	Hungarian	Romanian	Swedish
Danish	Korean	Russian	Turkish
Finnish	Lithuanian	Slovak	Ukrainian

Process Maps Installation

Process Maps are now delivered separately from the QAD .NET UI. By default, the QAD .NET UI does not include process maps.

The process maps for QAD Enterprise Applications – Enterprise Edition have been redesigned to include the following:

- *New Color Scheme* — The new color scheme has designated colors for all the Supply Chain Processes.
- *Enhanced Navigation* — The navigation structure was flattened, and the process map footer makes navigation from map to map easier.
- *Updated Vertical Maps* — The new version features eight updated vertical maps with more granularity than ever before.
- *New End-to-End Maps* — The high-level end-to-end maps Quote to Cash, Procure-to-Pay, and Plan to Perform were added.
- *Improved Supply Chain Process View* — The supply chain process view was enhanced with the new categories Design, Enable, and Engage.
- *Linkage to the Document Library* — Each node allows you to search the QAD Document Library with a click of a button to find relevant user guides, installation guides, or training guides.

Process Maps and Internet Explorer

You may experience the following issues when maintaining and viewing process maps in Internet Explorer:

- Internet Explorer 8 and 9 do not include the Adobe SVG plug-in, which is required for using the QAD .NET UI's Process Editor. QAD includes the SVG plug-in with QAD .NET UI in the client directory on the home server (in *`HomeServerURL`*`/client/SVGView.exe`) so that you can install it on machines running Internet Explorer 8 or 9.
- By default, the Process Viewer uses Silverlight rather than SVG.

- Note that with Internet Explorer 8 and 9, Print Preview is not supported for process maps and they do not print correctly.

Favorites Storage and Automatic Migration

Previously, when upgrading to a new release of the QAD .NET UI, saved Favorites were not migrated automatically. In this release, Favorites have now been enhanced so that:

- All Favorites are now stored on the home server (in a `UserMenu.xml` file) rather than locally.
- All Favorites, including menus, commands, browses, and operational metrics, are saved as QAD Shell URL (qadsh://) links.
- Configuration settings can be referenced in qadsh:// parameters.
- All favorites, starting with QAD .NET UI 2.7.1, are automatically migrated to the new format.

During migration, favorites previously stored locally (in the `UserMenu.dat` file) are converted to the `UserMenu.xml` file, which is stored on the home server. For backup, converted files are saved as .bak files.

Collections and Operational Metrics Installation

You no longer have to copy the browse collection, menu collection, and operational metrics files from the default directory for Enterprise Edition (*`TomcatInstallDir`*`/webapps/qadhome/configurations/default/eB3_storage`). Instead, the content installer in the QDT-based installation process will do that for you based on the system environments you specify while using QDT.

QAD Reporting Framework Service Installation

Starting with QAD 2012 EE, you should install and configure the QAD Reporting Framework Service on one or more Windows machines. This service continuously monitors a special report batch (QADSVC) for reports to be run immediately on the server. This service is necessary for several uses including report bursting and report integration with some maintenance programs. See the Service Mode section in chapter 6 (Administering Reports) of the *Reporting Framework User Guide* for more information.

Application Changes

The following summarizes application changes for this version of the QAD .NET UI.

Copying Links to Clipboard

You can now copy (and paste) QAD Shell URL (`qadsh://`) links to Desktop programs and browses by choosing Actions | Copy Link To Clipboard. You can then paste the URL into a browser to run the Desktop program or browse.

E-mail Action Options

The Action/E-mail feature in programs running in Desktop mode allows you to create an e-mail with a QAD Shell URI (`qadsh://`) to a Desktop program. There has been an inconsistency in this feature because different e-mail clients handle the QAD Shell URI differently, some recognizing it and allowing the e-mail recipient to launch the link and some not recognizing it and instead requiring the recipient to copy the link and paste it into a Web browser.

New options are now available to:

- Allow an administrator to set (using `client-session.xml`) whether they want the QAD Shell URI to be used directly (which is and has been the default) or if they want to wrap this QAD Shell URI in an HTTP URI (`http://`), which is more widely recognized by e-mail clients.
- Provide an option to turn on or off the inclusion of the full URI in the e-mail. (By default two links are put in the e-mail—the first is the program label that links to the URI and the second is the full URI. This option controls the second link.)
- Provide an option to create the link as text instead of HTML (for e-mail clients or settings that are text based).

These options are controlled by the following settings, which can be added to `client-session.xml`:

```
<EmailAction.UseHTTP>true</EmailAction.UseHTTP>
```

When set to `true`, the HTTP URI is used in the e-mail. When `false`, the direct QAD Shell URI is used.

```
<EmailAction.IncludeURI>true</EmailAction.IncludeURI>
```

When set to `true`, the full URI is added as a link in the e-mail. When set to `false`, it is not included.

```
<EmailAction.UseText>true</EmailAction.UseText>
```

When set to `true`, the link will be text. When set to `false`, the link will be HTML.

Parameters in Terminal Script in User Option Telnet Maintenance

You can now specify the following parameters in the Script Value field in User Option Telnet Maintenance:

${d} – Domain

${u} – User ID

${e} – Entity

${c} – Code Page

Any number of these can be included in Script Value, with each separated by a space. For example:

```
/dr01/scripts/telnet.ksh ${d} ${u} ${e} ${c}
```

These will then be available in the telnet script as standard parameters (${1} ${2} ${3} ${4}).

Lookups Not Limited to Starting Point of Field Value

By default, the initial item on a field's lookup browse is the current value in the field. For example, on the Sales Order Maintenance program's Credit Terms field, if the current value is 30D (for thirty days after invoice date), the lookup browse for the field will list the terms codes for the field starting with 30D, the current value. Although this is the list that many would like to see, in some cases you might want the lookup to list all the possible options for the field rather than just the ones starting at the current value. For example, for the Credit Terms field, you might want to have all the term codes listed, not just the ones starting at 30D.

You can now have the lookup browses for specified fields list all the options for the field rather than just the ones starting at the current value. The configuration steps must be done by a system administrator with access to the QAD system files.

Note The ability to change the lookup browse listing only applies to programs that run in Desktop mode.

Changing the Lookup Browse Listing

To change a field's lookup browse to list all the options for a field, do the following:

1. Identify the program name (for example, for Sales Order Maintenance, the program name is `sosomt.p`).
2. Identify the field name (for example, for the Credit Terms field, the field name is `so_cr_terms`).
3. In your QAD installation, locate the `com/mfgpro/setting.dat` file and `com/qad/mfgpro/setting.dat` file.
4. Add the following line to the end of both `setting.dat` files:

   `bl:`*`program_name`*`:b:`*`field_name`*`=true`

   where:

   `bl` indicates "blank lookup".

   `program_name` specifies the program, but with no dot (".") before the `p`. For instance, `sosomtp` rather than `sosomt.p`.

   `b` indicates the current frame.

   `field_name` specifies the field. For instance, `so_cr_terms` specifies Sales Order Maintenance's Credit Terms field.

   For example, for Sales Order Maintenance's Credit Terms field, you add the following line to the end of both `setting.dat` files and make sure to leave an extra blank line below it:

   ```
   bl:sosomtp:b:so_cr_terms=true
   ```

5. Restart the Connection Manager.

Output to Text Lines Now Configurable

You can now configure the maximum number of lines displayed to the user when running a legacy report with output to text. To do so, use the `<QView>` ... `<TextReportLines>` setting in the `client-session.xml` file. The default is 100,000 lines:

```
<TextReportLines>100000</TextReportines>
```

Browse Performance Checking

The system can now do a performance check on the index use of a new browse definition when you save it from Browse Maintenance. The performance check, Show Index Information, will help to avoid the creation of poorly performing browses with non-indexed fields in joins, filters, and sorts.

When a browse definition is saved from Browse Maintenance, the Progress Query Parser's INDEX-INFORMATION is examined. Checks are made to determine whether indexes can be used. Improperly defined query definitions are indicated as whole index scans and are displayed in red in the Index Information tab. For instance, the browse definition could result in a table scan that could cause performance issues and you might need to modify the definition so that no whole index scans occur. Tables with large numbers of records might negatively impact performance, so you might need to analyze the query string to identify possible causes. To do so, open the Query String tab to view the dynamically generated query string as determined by the Browse Engine.

The performance check is on by default, but can be changed from the Show Index Information setting Tools | Options or from the `config-session.xml` file, which now includes the following:

```
<DotNetBrowseMaintenanceShowIndexInformation>true</DotNetBrowseMaintenanceShow
IndexInformation>
```

The setting specifies whether the output of the Progress INDEX-INFORMATION attribute for a query is displayed when there is an issue.

Note The ability to examine what the Progress Query Parser determines as the indexes for a query is limited. The Browse Engine currently only exposes the dynamic query string prior to appending sorts, local variables, pre and post processor commands, and so on. This performance check will help eliminate most poorly performing browses that have been built from improperly constructed definitions. However, this check will not cover situations where users apply search conditions and sorts after the browse has been displayed in the user interface.

Browse Export and Import Identifies Language Mismatches

Previously, when a browse was exported using Browse Maintenance, the labels used in the browse were included for all the supported languages. However, when imported, the only the labels for the user importing the browse were included with the imported browse. Now, an imported browse will include all the labels for languages supported by the system, not just the ones for the user's current language. If the browse includes labels for languages that are not defined in the system, a warning message will be included in the system log file ("WARNING: The Language *language* for the Label Term *label term* does not exist in the target language. Label not imported for this language").

Find Feature for Legacy Reports Output as Text

Any legacy report that is output to text now includes a Find box on the toolbar for searching the report. You can also activate the Find feature by entering Ctrl+F.

Active Directory Enhancements

The QAD .NET UI supports Microsoft's Active Directory authentication. With this release, Active Directory support has been enhanced so that the QAD .NET UI:

- Allows any domain to be specified at login.
- Allows the default domain to be specified on the home server.
- Enables a list of valid domains to be specified on the home server.
- Removes OS user ID restriction. Allows any domain and user ID combination to be entered, except local domain.
- Enables user ID mapping between Active Directory and QAD, which eliminates the eight-character user ID limitation.
- Eliminates blank QAD password vulnerability.
- Eliminates domain spoofing.

With these Active Directory enhancements, users will be required to do the following:

- Enter a QAD user ID and password the first time they use Active Directory authentication.
- Re-encrypt the credentials store whenever their Active Directory password changes. They will be prompted to enter the Active Directory password used during the original encryption.
- Re-enter their QAD credentials whenever they are unable to remember the Active Directory password used during the original encryption.

With Active Directory, QAD passwords are encrypted in a credentials file named *<domain>-<user>*-credentials.xml, located on the home server in a */<environment>*/storage/user-data/*<user>* directory. Passwords are encrypted using AES with a 256-bit key generated at runtime using the password base key deriviation algorithm (RFC 2898).

Active Directory Configuration Settings

The following settings in client-session.xml: configure Active Directory:

<QAD.Authentication.ActiveDirectory.Enabled>

This setting enables or disables Active Directory authentication, with true enabling Active Directory and false disabling Active Directory.

<QAD.Authentication.ActiveDirectory.Domain>

This setting specifies the default Active Directory domain. If a domain is not specified, the current PC domain is used. Domains are resolved in the following order:

1. Login form: users can specify a domain using the syntax {domain}\(username).
2. Configuration setting in <QAD.Authentication.ActiveDirectory.Domain>
3. Current active PC domain.

<QAD.Authentication.ActiveDirectory.ValidDomains>

This setting specifies a comma-delimited list of valid domains used during Active Directory authentication.

Note In client-session.xml, the <QAD.Authentication.ActiveDirectory> setting has now been replaced by the <QAD.Authentication.ActiveDirectory.Enabled> setting.

Enabling Active Directory Authentication

To enable Active Directory authentication:

1. Define users so that the user IDs match the Windows user IDs. Assign temporary QAD passwords to new user IDs.
2. Locate the client-session.xml file on the home server. (By default, the file is located in the *TomcatInstallDir*/webapps/qadhome/configurations/default directory.)
3. In the client-session.xml file, set <QAD.Authentication.ActiveDirectory.Enabled> to true.

When a user first logs in to the QAD .NET UI, the system prompts the user to enter the temporary QAD password that was assigned to the user ID. Entering the temporary password then completes the Active Directory setup. The next time that the user logs in to the QAD .NET UI, the user must use their Windows user ID and password.

Note In previous releases, when setting up Active Directory, you had to enable the Enforce OS User ID option in Security Control (mgurpmmt.p). Additionally, the passwords for user IDs had to be sent to blank. Starting with QAD .NET UI 2.9.5, these steps are no longer necessary.

QAD Shell URL (qadsh://) Protocol Enhancements

Starting with QAD .NET UI 2.9.5, the qadsh:// protocol has been enhanced as follows:

- The qadsh:// protocol is now registered on a user basis, allowing different users to access different versions of the QAD .NET UI client on the same machine.
- The qadsh:// protocol is now registered during the launch of the QAD .NET UI, ensuring that the protocol is defined correctly.
- The qadsh:// protocol is only registered if the registry setting for the currently running QAD .NET UI client is different than the one already registered.
- The qadsh:// protocol always points to the the last QAD .NET UI version that was launched rather than the version most recently installed.
- The user registry settings do not require administrative permissions, enabling XCOPY installations.

QAD Document Library and QAD Assist

The QAD Document Library (http://www.qad.com/documentlibrary/) now offers a complete set of all QAD user guides, training guides, and other materials. With all the user guides now available in the QAD Document Library, QAD Assist now includes a link to the QAD Document Library and only includes the program and field help as content.

Legacy Report Output and HTML Syntax

You can now configure specific legacy reports to display HTML syntax in output to “page”. The current default displays the HTML code instead of rendering the HTML itself. This default exists to overcome the problem in which the report data had special HTML characters such as < or >. If the data has such characters and output to “page” is used, then an error results and the report does not render. However, there are situations in which HTML (such as a link or a button) in the report output is useful, so you can now specify which report programs will be allowed to display HTML.

Requirements

The report program specified to display HTML must not have the possibility of data in it that has special HTML characters that are not proper HTML syntax. For example, the item number `abc<1000>` has HTML brackets in it but is not proper HTML syntax and will cause the report not to render.

The report cannot be on the list of reports in the enhanced format (`beautifyReports.lst`).

Setting Report Programs to Display HTML

1. Edit `com/mfgpro/setting.dat` and add a comma-delimited list of report programs to the `htmlReports` setting.
2. Make the same change to `com/qad/mfgpro/setting.dat`.
3. Restart the Connection Manager.

Note QAD does not provide support for HTML customizations of reports.

Allowing Changing of Legacy Report Output

Legacy reports can now be configured to output their report files to a directory other than the default working directory of the common Connection Manager user.

This feature is turned off by default but can be enabled as follows:

1. Locate `com/mfgpro/setting.dat`.
2. Follow the instructions in that file to set the `reportPath` setting.
3. Make the same change to `com/qad/mfgpro/setting.dat`.
4. Restart the Connection Manager.
5. Restart the QAD .NET UI client.

This only applies to Desktop reports run from the QAD .NET UI. It does not apply to:

- Terminal or CHUI mode.
- Reports that have two output fields (such as Invoice Post and Print).
- Batch reports (in any mode).

File system permissions must be taken into account when using this mechanism. Keep in mind that even though the report file can be redirected to a different location, it is still being written by the common Connection Manager user. As a result, that user will need to have the access to write the report files to whatever location is specified here.

Reporting Framework Changes

The following summarizes Reporting Framework changes for this version of the QAD .NET UI.

Additional Bar Code Types (Including 2D)

Previously, the only bar code field supported for reports was limited to handling only nine types of 1D bar codes and no 2D bar codes. Furthermore, it had little support for parameters for these types. In this release, two new bar code fields have been added to the Report Resource Designer program:

- Linear Bar Code field
  - Supports 34 types of 1D bar codes
  - Exposes extensive sets of parameters to fully control the bar code settings
- 2D Barcode field
  - Supports 3 popular types of 2D bar codes (DataMatrix, PDF-417, and QR Code)
  - Exposes extensive sets of parameters to fully control the bar code settings

Note The original Bar Code field is still supported for back compatibility.

Dynamic Layout Selection by Data Source

You can now have a report dynamically determine which page layout to use. Optionally, the user can choose the layout.

If a report has more than one page layout design created for it (each giving different visualizations of the same data), you can have the data source program logic select a particular layout at run-time. For example, if a report needed to output one type of form for data where a customer is in Germany and another output form if the customer is in Brazil, then the data source logic could choose the proper form layout automatically.

In the normal case where the data source program does not attempt to control the layout selection, the list of available layouts (displayed when the user clicks the Layouts tool button in the report viewer) is still available for users to choose before running the report. However, when the data source program asserts control over the layout, the Layouts tool button is disabled to prevent users from overriding the layout selection. It is also possible for the data source logic to allow the user to choose any desired subset of the layouts, if a manual override is deemed allowable, by displaying those options as a search field with a list of allowed layouts.

Specifying Dynamically Selected Layouts

1. Create multiple layouts in the Report Resource Designer, all for the same report code (RRO code).
2. To have your data source program choose the layout, include a table in your business data set called DataSourceReportSettings, with a char field called sys_default_report_definition.
3. Put the choice of layout name into this field and the QAD .NET UI client will use this for rendering. The layout name is the value entered in the Report Definition Name field when saving the definition from the Report Resource Designer (stored in rptresd_det.rptresd_name after being saved).
4. Specify metadata for the DataSourceReportSettings.sys_default_report_definition field. This will cause the report viewer program to disable the Layouts tool button (which normally would be active and confusing due to the existence of multiple layouts).

   You have a choice of whether to make this field searchable. If you want to give the user no contol over the layout, make this field non-searchable. If you want to allow the user some control over which layout to use, make it searchable. You can provide a value list for this field in the metadata, which will control the list of layouts that the user will be able to choose from in the search panel. If you make this searchable, you should get the user-entered value from the filter conditions and use it when populating the DataSourceReportSettings.sys_default_report_definition field in the code described in step 2.

   Note When running this report from the menu, the above logic will apply. However, when running the report from Report Resource Designer | Preview, the layout used will always be whatever layout is currently in memory in the designer; the data source's choice will always be ignored. This is to keep true to the designer's underlying philosophy that the preview should always run what is in there, which might not even have been saved to the database yet. Therefore, to test the actual data source logic, it is necessary to run from the menu (perhaps requiring a temporary test menu item if the real menu item does not exist in the test system yet).

Dynamic Currency Formats

Currency Rounding functions are now available in the Report Resource Designer as part of the QAD_NumberUtil script object. These new VBScript functions provide rounding based on the data maintained by the Rounding Method menu items in the system. These functions can be invoked from within the Report Resource Designer program's VBScript hook points to provide standard rounding methods for any desired fields.

As described in the *Reporting Framework User Guide*, .NET script objects make it possible to expand the functionality available in the report designer to go beyond VBScript logic. In this release, the following script objects have been included to support dynamic currency formats.

QAD_NumberUtil.RoundNumber(currency, number)

Rounds a number based on the given currency's rounding method (rounding unit and threshold).

Table 1
QAD_NumberUtil.RoundNumber(currency, number) Parameters

Name	Input/Output	Data Type	Description
currency	Input	String	Currency upon which to base the rounding method.
number	Input	Object	Number to be rounded.
	Return Value	Object	Returns the rounded amount.

QAD_NumberUtil.RoundNumberBaseCurrency(number)

Rounds a number based on the base currency's rounding method.

Table 2
QAD_NumberUtil.RoundNumberBaseCurrency(number) Parameters

Name	Input/Output	Data Type	Description
number	Input	Object	Number to be rounded.
	Return Value	Object	Returns the rounded amount.

QAD_NumberUtil.ConvertFormat(currency, format, useZeros)

Converts the given number format string to one which properly matches the rounding method of the given currency. The useZeros variable controls whether additional decimal places are formatted with zeros (.000) or pound signs (.###).

Table 3
QAD_NumberUtil.ConvertFormat(currency, format, useZeros) Parameters

Name	Input/Output	Data Type	Description
currency	Input	String	Currency upon which to base the conversion.
format	Input	String	Existing numeric field format.
useZeros	Input	Boolean	Specifies whether additional places are formatted with zeros or pound signs.
	Return Value	Object	Returns a format that properly matches the rounding method of the currency.

QAD_NumberUtil.ConvertFormatBaseCurrency(format, useZeros)

Converts the given number format string to one that properly matches the rounding method of the base currency.

Table 4
QAD_NumberUtil.ConvertFormatBaseCurrency(currency, format, useZeros) Parameters

Name	Input/Output	Data Type	Description
format	Input	String	Existing numeric field format.
useZeros	Input	Boolean	Specifies whether additional places are formatted with zeros or pound signs.
	Return Value	Object	Returns a format that properly matches the rounding method of the currency.

QAD_NumberUtil.GetThreshold(currency)

Gets the rounding method’s threshold for the given currency.

Table 5
QAD_NumberUtil.GetThreshold(currency) Parameters

Name	Input/Output	Data Type	Description
currency	Input	String	Specifies the currency.
	Return Value	Object	Returns the rounding method’s threshold for the specified currency.

QAD_NumberUtil.GetThresholdBaseCurrency()

Gets the rounding method’s threshold for the base currency.

Table 6
QAD_NumberUtil.GetThresholdBaseCurrency() Parameters

Name	Input/Output	Data Type	Description
	Return Value	Object	Returns the rounding method’s threshold for the base currency.

QAD_NumberUtil.GetRoundingUnit(currency)

Gets the rounding method’s unit for the given currency.

Table 7
QAD_NumberUtil.GetRoundingUnit(currency) Parameters

Name	Input/Output	Data Type	Description
currency	Input	String	Specifies the currency.
	Return Value	Object	Returns the rounding method’s unit for the specified currency.

QAD_NumberUtil.GetRoundingUnitBaseCurrency()

Gets the rounding method’s unit for the base currency.

Table 8
QAD_NumberUtil.GetRoundingUnitBaseCurrency() Parameters

Name	Input/Output	Data Type	Description
	Return Value	Object	Returns the rounding method’s unit for the base currency.

New Immediate Service Report Server Mode

The original report server operated in scheduled batch mode: at scheduled times the server process would start and process the desired batch of reports (for example, every night at midnight, or at end of month). A new, additional report server mode is now available: a continuously running process that runs reports as soon as they are scheduled with a special batch ID (a virtual batch named QADSVC, representing the QAD service).

This new service is called the QAD Reporting Framework Service. The service is especially useful to support cases where the application that needs to run a report is not running in a QAD .NET UI process (for example, a Progress program running on a Linux box, or an application running on a mobile device). This new service is also essential to the infrastructure of the new report bursting capability (see “Report Bursting with Dynamic Output Routing” on page 14).

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

Note When running this mode from a Progress program, the report might not necessarily be viewable.

Note Any program created to schedule reports in the Immediate Mode needs to either be running the Service Interface Layer (Progress program) or be running from within the .NET UI. Otherwise, the program will not have security access.

Support for Progress Character Mode Programs to Launch Reports

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

The new immediate report server mode (see “New Immediate Service Report Server Mode” on page 12) provides the basis for a solution of the time delay: by scheduling the report to be run in the virtual immediate batch ID, the report will be run typically within a matter of seconds from the time it was scheduled.

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

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

Configurable Output File Naming and Location

Reports scheduled to be run on a report server previously had no flexibility regarding the file name and folder path of the output file stored on the server. This not only restricted the possibilities for organizing report output, but also reduced flexibility of implementing security authorization on the web repository in which these output files are stored.

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

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

Report Bursting with Dynamic Output Routing

This release contains new infrastructure to facilitate the mass-running and distribution of reports: report bursting. A report burst is a special way that a report can be run that involves the following aspects:

- A report burst can be configured to automatically split a big report into many smaller reports, relieving end users of the burden of manually running numerous reports. Any field in the top-level table of the report's data set can be chosen as the split field (for example, a report could be split to output one report per customer ID, or one report per item number).
- A report burst can be configured to dynamically route each of the split output reports to different e-mail, file, and printer destinations. The logic can be based on data values; for example, the customer ID in each of the output reports could be mapped to an e-mail address for that customer to send the report t..

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

Report bursts internally schedule each of the split output reports to be asynchronously (but immediately) run by a report server using the QAD Reporting Framework Service (see "New Immediate Service Report Server Mode" on page 12). This leverages the many benefits of the report server architecture such as robustness, scalability, and failover.

Although the new bursting capability is powerful, it is also so new that the QAD 2012 EE release does not contain any new QAD application programs that take advantage of it (though the possibility is now open for such applications in future QAD releases). You can, however, set up your own desired scenarios for bursting using the following mechanisms:

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

The logic that controls the dynamic routing settings is completely configurable. However, this release contains no default routing logic. It is therefore necessary to first codify the desired rules using the Progress programming language in a special dynamic-routing program that will be invoked during report bursts. Future QAD releases may contain out-of-the-box sets of configurable rules for dynamic settings.

If assistance is desired with creating report burst programs or setting up the desired routing rules, please consult QAD Services.

Report API Enhancements

Application Programming Interfaces (APIs) allow one program to invoke functionality in another. Prior to this release, the only API for the QAD Reporting Framework was the Scheduled Report API, which allows programs to schedule reports to be run asynchronously on a report server. This API is implemented as a Progress program that gets called on an application server and is callable from Progress and .NET programs.

This release enhances the reporting framework APIs in several ways:

- A native .NET version of the Scheduled Report API has been added. This allows .NET applications to invoke the Scheduled Report API in a more simple fashion: a simple direct .NET object can be constructed and called to schedule a report. This encapsulates the inner details of the remote proxy networking, hiding it from the application programmer.
- A .NET Run Report API has been created. This allows any .NET program running under QAD .NET UI to be able to invoke a report to be run synchronously in the same process (instead of asynchronously on a server).
- A new capability has been added to the Scheduled Report API (both .NET and original Progress versions): the ability to pass the report data set into the request, instead of having the report run a data query at run-time. This is useful in scenarios where the data has already been queried and just needs to be passed to the report for rendering. For example, an application can use this if it needs to rerun a report with the exact same data as the previous run.

  Note This data input option is also available on the synchronous .NET Run Report API.

These API enhancements greatly increase the options and possibilities for leveraging the QAD Reporting Framework in custom programs and system automation.

Scheduled Report Default Printer

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

The format to specify the default printer in `client-session.xml` is as follows:

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

Programs in Terminal Mode Only

Some programs are only available in Terminal mode, which emulates the Character UI within the QAD .NET UI. You navigate the program in the same way as in the Character UI. The following programs are only available in Terminal mode:

- Accounts Not To Convert Maint
- AP Integrity Report
- Archive File Reload
- Call Queue Manager
- Change Deferred/Accrued Accounts
- CIM Data Load Process Monitor
- Combined Integrity Checks
- Compile Programs
- Convert Ship Qty in Ship UM

- Count Program
- Create Records for Printer Output
- Database Connect
- Database Disconnect
- Database Table Size Inquiry
- Debug CIM Document
- Dump Export/Import Doc for Edit
- End User Time Zone Change Util
- Escalation Monitor
- Exit to Operating System
- Export/Import Document Query
- Field Eligibility Maintenance
- Fixed Asset Maintenance
- Fixed Assets Integrity Report
- GL Integrity Report
- GLRW Mismatch A/C Code
- Initial Euro Exchange Rate Copy
- Inventory Integrity Report
- License Registration
- Multiple Time Zones Startup Util
- PO Integrity Report
- Process Import Documents
- Program Level
- Program/Text File Display
- Receive Import Documents
- Reload Edited Export/Import Doc
- Required Ship Schedule Update
- Send Export Documents
- Sequence Maintenance
- Server Time Zone Change Util
- Set Multiple BOL Print Utility
- Ship-From to AR
- Trading Partner Library Load
- Trading Partner Library Unload
- WIP Integrity Report