User Help for the RBDMS.NET Windows Administration Application (WinAdmin.NET)

Initial Release: November 2007
Revision 1: March 26, 2008
Revision 2: June 20, 2008
Revision 3: November 13, 2008
Revision 4: March 6, 2009
Revision 5: July 22, 2009
Revision 6: October 5, 2009
Revision 7: December 9, 2009
Revision 8: May 5, 2011

Table of Contents

Summary of Features

About the RBDMS WinAdmin.NET Application

The RBDMS.NET Windows Administrative (WinAdmin.NET) module was developed for developers and database administrators to use to control multiple Internet- and intranet-based applications, such as RBDMS, Data Mining, eReport, and ePermit. Users who are not members of the administrator or developer role should not be given access to the RBDMS WinAdmin.NET application because its misuse could result in application malfunction, data loss, or both. The administrative application is available as a desktop smart client application.

The application, which was developed with the .NET Framework 2.0, offers administrative control over the following functions at the level of granularity required:

Base Modules:

- Application configuration
- Security
- Workflow process design
- Menu management
- Personalization
- Filtering and reporting
- Error and exception handling

Add-on Modules:

- Rules engine
- Notification/communication
- Audit/conflict resolution

This help file is meant to document only the uses and inner workings of the RBDMS WinAdmin.NET application. Information about the availability, installation, configuration, use, and maintenance of Microsoft development tools and enterprise software, such as SQL Server, Reporting Services, and Visual Studio, is available from the Microsoft Web site.

The RBDMS WinAdmin.NET application was developed in part with Department of Energy grant funding administered through the Ground Water Protection Council. Programming was provided by Virtual Engineering Solutions, Inc. and Coordinate Solutions, Inc.

Introduction to Component Base Modules

Application Configuration

RBDMS WinAdmin.NET can be used to control multiple Internet- and intranet-based applications through the **app.config** file, selected items of which can be edited

through the **Configuration** form on the **Forms** menu. Program directory locations for report templates, Web applications, and database connection strings can be set on this form, as can the preferred security protocol for connecting to the specified database.

RBDMS WinAdmin.NET can be configured to manage the security settings and other behavior of multiple applications by editing the configuration files for connection strings and user settings. Two configuration files are installed by default into the **C:\Program Files\GWPC\RbdmsWinAdmin\configs** folder: **<appName>_connectionString.config** and the **<appName>_userSetting.config**, with the <appName> representing a two- or more-letter designation for the application name**.** Both of these files must be customized for the target environment through the **Configuration** form's **Database Connection** and **Settings** subpages. These two configuration files contain many of the settings that will be necessarily customized to target the installation environment (i.e., connection strings) and to define application behavior and user personalization rules.

Once the files have been customized and tested through the **Configuration** form, the database administrator can copy and re-name the two configuration files, then repeat the steps to re-customize the files for a different target application. These customized configuration files should be preserved for each application in the **\configs** folder and a copy of the customized configuration files maintained in a secure location.

The master application configuration file **RbdmsWin.exe.config** includes code that looks for the configuration files in the application **\configs** folder. WinAdmin.NET displays the letters before the underscore in the file name in the *Configuration Set* lookup list on the **Select Config** subpage of the **Configuration** form. Each time you select a new configuration set on the **Select Config** subpage, thus changing the managed application, RBDMSWinAdmin.NET will prompt you to close and then re-open automatically. This action will load the settings for the newly targeted managed application.

For more information, please see Configuring WinAdmin.NET to Target Managed Applications.

Security

ASP.NET 2.0 forms security and membership management have been extended in RBDMS WinAdmin.NET to derive login information from users' Windows login and to allow anonymous users to login via a Web form. In addition to tracking user access rights to forms, the module also has been extended to track *Create*, *Read*, *Update*, and *Delete* (CRUD) rights at a highly granular (individual control) level on the **Security** form. A context menu and a toolbar provide create, edit, and delete user commands, with drag-and-drop features for adding users to roles, roles to users, rights to roles, and roles to rights.

For more information, please see Managing Users, Roles, and Rights.

The December 2009 release of WinAdmin.NET offers encryption of the database connection strings to the managed locations by default for greater security.

Workflow Process Design

Integrated with the security module and the personalization module, the base process management module allows modeling of processes, tasks, and roles, and it leverages previous work with "sets" in RBDMS for Water through a drag-and-drop user interface. *Sets* in this case refer to user-defined lists of tasks, each of which is assigned to roles, grouped into a top-level process. For more information, please see Managing Menus and Processes.

Menu Management

The RBDMS menu management module inherits and extends the features of the Microsoft Site Map Provider. The integration with the security module ensures that the application displays only those forms, reports, and controls to which the logged-in user has been given access rights and processes in which the user is an active participant. For more information, please see Managing Menus and Processes.

Personalization

The base personalization module implements an extensible object that can be serialized. When written to the ASP.NET profile tables, the object retains user preferences between sessions, such as last selection criteria, form size, and skin design.

Filtering and Reporting

The ASP.NET version of the filtering module developed in 2005 for the GWPC has been re-factored, and the data-driven report menu and filtering form was converted to ASP.NET custom controls to facilitate their reuse throughout the various RBDMS implementations. Report templates are organized into folders in a program directory that can be defined in the application configuration file, and this organization is reflected in the menu tree control within the application. Double-clicking a report name in the tree control opens the report template in **Edit** mode. The .rdlc templates open in Visual Studio, while grid-style reports open in a report editor within RBDMS WinAdmin.NET.

The dynamic filtering parameters associated with each report are defined in the RBDMS.NET administrative application on the **Edit Filters** form, which includes the capability of defining the format and content of individual controls for use as selection criteria and then assigning a set (filter) name to the grouped controls. This filter name is used as the report parameter *CollName* in SQL Server Reporting Services .rdlc files. SQL Server Reporting Services expressions can be used in the filter form field definitions. A right-click context menu on the **Set Name** on the **Edit Filters** form gives the ability to preview the filters you create for correctness.

For additional information, please see the topics in the chapter **Managing Reports**.

Error and Exception Handling

The earlier work on RBDMS.NET in 2005 for the error and exception handling was integrated and re-factored, and standards were developed to describe how error and exception messages should be handled so that, for example, an application can be programmed to display detailed error messages if the user is a member of the developer security role, while displaying less detailed, user-friendly messages for other users. ASP.NET error display was implemented as a custom control and includes the ability to display data validation messages in addition to capturing and displaying errors in the Event Viewer *Application* log.

Rules Engine

The Rules Engine in RBDMS WinAdmin.NET has been abstracted from the Microsoft Windows Workflow Foundation. It is useful in adding business logic and data validation rules for RBDMS eCommerce applications such as ePermit and eReport. Users can create dynamic rule sets that link the rules to either a custom class or references to tables to govern application behavior. This result allows separate management of data objects and business rules.

The Rules Engine allows the agency to create business rules that can be applied to controls on the dynamic web forms and other applications. The Rules Engine consists of a Rule Set Browser and a Rule Set Editor. It creates and adapts business rules and data validation checks. Data fields in the online application will be governed by rule sets, and the Rule Set Editor allows database administrators to select, assign, and chain rules in a point-and-click environment. Once database administrators create new rules in the Editor, they can link the rules to either custom class or references to tables through the Rule Set Browser.

The application will support adding a new rule or having multiple rules against the same field, setting priorities, or using .NET commands, at the administrator's preference. The Rule Set editor also can be used with extension methods including regular expression methods, date type checks, range checks, stored procedure checks, and table checks (e.g., "check for valid key"). Forms may be bound to both internal and external rule sets, which may define different sets of form behaviors depending on whether the user is in a *Public* or *Agency* role. The database administrator will be able to change a rule without having to change (i.e., compile) the application.

For more information, please see Defining Rule Sets.

Notification/Communication

A notification (also referred to as "alerts") module built on work previously done for the RBDMS for Water application is being re-factored for general RBDMS.NET use. Alerts are automated messages that can be customized to respond to data QA/QC flags, program- or date-specific requirements or actions, or workflow process step notifications. Alerts are generated through .NET procedures that are stored in a business layer separate from the user interface, and alert sets can be shared or customized to specific users. Users can define the notification frequency of their subscribed alerts.

The Notification module is powered by the **ePermitWindowService**, a .NET 3.5 WCF service application that creates **eNotify** records and sends out e-mail notifications. The **ePermitWindowsService** is installed by default in the **C:\Program Files\GWPC\ePermitWindowsService** directory. Before starting the service, agency staff will need to edit the **ePermitWindowsService.exe.config** file to update the following:

- ConnectionStrings – update to connect to COGCC server.
- ApplicationSettings
  - SmtpHost
  - MailFromAddress
  - MailFromDisplayName
  - SmtpUserName
  - SmtpPassword

When the service is started, it writes an entry into the computer event log to indicate that it started. The service then creates an instance of the **ePermitBLL.Notify** class.

When the Notify *StartTimer* function is called, the *NotifyFrequency* setting is read from the .config file. The *NotifyFrequency* includes multiple date intervals (in decimal day units). The initial setting is for 0.0034722,0.041667,1 which represents 5 minutes, 1 hour, and 1 day, respectively. A list of *NotifyInterval* objects is created for each value.

The application .config file also includes a *NotifyInterval* setting. The *NotifyInterval* specifies the interval in minutes for triggering the timer elapsed event. Each time the timer elapsed event occurs, the *NotifyElapsed* property of each *NotifyInterval* object is incremented with the elapsed time from the timer. If the *NotifyInterval* object's *NotifyElapsed* value is equal to or greater than the *NotifyFrequency,* then it selects **eNotify** records that have a *NotifyComplete* of False and have a *eNotify.NotifyFrequency* less than or equal to the current loops *NotifyFrequency*.

The **eNotify** objects timer is set to use an interval (in fractional days) for querying the **eNotify** table. The 1-day interval starts at 12:00AM, the 1-hour intervals start on the hour, and 5-minute intervals start at the 5-minute interval immediately less than the current time (i.e., if the current time is 1:37 a.m., then the interval will begin at 1:35 a.m.).

The query will order the results by *UserID, DueDate* and *FormKey* (e.g., *Doc_Num*). The program loops through the query results and creates a single email for each user with a table containing the **eNotify** records for the user ordered by *DueDate* and *FormKey*. Each loop is wrapped in a transaction, and once the transaction is complete, the *NotifyComplete* value for the records in each email is set to true. System administrators may choose to delete **eNotify** records with NotifyComplete = True after a month or so.

For additional information, please see the topic Setting Alerts.

Audit/Conflict Resolution

Beyond the existing capability of tracking the *ModifyUser* and *ModifyDate* fields, an audit/conflict module tracks the changes to specified database fields through a

library that will allow "roll backs" to previous data entries without having to restore the database from a backup once a user interface is built. The envisioned user interface will allow the user to review the audit trail or conflicts and select appropriate actions. An appropriate security right will control who has access to these functions. Configuration options in the module will control how large the audit trail is allowed to grow, which data fields are tracked, and how long the change histories are retained. For more information, please see Auditing and Resolving Database Conflicts.

Putting RBDMS WinAdmin.NET to Work

Installing RBDMS WinAdmin.NET

RBDMS WinAdmin.NET is delivered as a standard .msi file (**RbdmsWinSetup.msi**). Double-click the file name in Windows Explorer to launch the installation wizard and then follow the directions presented in the wizard. By default, the installation is loaded to the **C:\Program Files\GWPC\RbdmsWinAdmin** folder, and a shortcut to launch the application is written to the **Start | All Programs | GWPC** menu folder.

Once you have installed the application, you will need to configure RBDMS WinAdmin.NET to target the RBDMS.NET or eCommerce application you intend to manage with it. This may optionally involve attaching an RBDMS security database.

Configuring WinAdmin.NET to Target Managed Applications

You can configure RBDMS WinAdmin.NET to manage multiple applications through the settings available on the **Configuration** form, available from the **Forms** menu. The **Configuration** form includes three tabbed subforms: **Select Config**, **Database Connection**, and **Settings**. The **Configuration** form is the user interface for editing the application configuration files, which are located in the WinAdmin.NET installation folder, which (by default) is **C:\Program Files\GWPC\RbdmsWinAdmin\configs**.

Managing Multiple Applications

As discussed in Application Configuration, RBDMS WinAdmin.NET can be configured to manage the security settings and other behavior of multiple applications. A set of two configuration files must be defined in the **C:\Program Files\GWPC\RbdmsWinAdmin\configs** folder for each RBDMS.NET and RBDMS eCommerce application that you wish to manage through RBDMS WinAdmin.NET:

- **<StatePrefix>_connectionString.config,** which houses the database connection strings.
- **<StatePrefix>_userSetting.config,** which includes report templates, GIS paths, and other settings.

As a convention, each of these files is named with the application designation, often a state abbreviation, in front of the underscore in the file names. Once both of these files are customized for the target environment through the **Configuration** form's **Database Connection** and **Settings** subpages, they can be copied and preserved for that application through Windows Explorer and then another set can be re-customized and preserved for a different managed application through the same process. The WinAdmin.NET configuration file, **RbdmsWin.exe.config,** contains code that looks for these custom configuration files. The designations in front of the underscore in these file names are used for the lookup list that appears in the *Configuration Set* dropdown field on the **Select Config** page of the **Configuration** form in RBDMS WinAdmin.NET.

RBDMS WinAdmin.NET installs with a set of default configuration files. Upon installation, you may wish to copy and edit these files with Visual Studio to create set of files specific to your targeted RBDMS application.

When you change the selection of the configuration set files on the **Select Config** page, WinAdmin.NET will ask to close and then automatically re-open to refresh the settings and connection strings for the managed application that will become the focus of operations within RBDMS WinAdmin.NET.

Setting the Database Connection:

Note: The December 2009 release installs with the database connections strings encrypted by default. This encryption makes the connection strings invisible on the **Configuration** form. Before proceeding with the following steps, you will need to decrypt the connection strings in the **<StatePrefix>_userSettings.config** file.

This **Database Connection** tab of the Configuation form exposes several elements of the underlying **<StatePrefix>_connectionString.config** file for the managed application for editing. This .config file is located in the **C:\Program Files\GWPC\RbdmsWinAdmin\configs** folder. To set the ADO data source connection to the application that you will manage through RBDMS WinAdmin.NET, follow these steps:

1. Select **Forms | Configuration** to open the **Configuration** form.
2. On the **Database Connection** tab, select the connection you wish to configure. Three connection strings are shown by default for RBDMS applications: the connection string to the RBDMS database, the connection to the Security database, and the connection to the DMZ (Web server application).
3. In the *Connection Settings* portion of the **Configuration** form, enter the credentials appropriate to the SQL Server database you selected in step 2 (*User ID* and *Password):*
    a. In the *Server Name* field, enter the name of the SQL Server database server you want to connect to.
    b. In the *Database Name* field, enter the SQL Server database name.

OR

    a. Click the checkbox if you are using Integrated Security.

4. *C*lick the **Test Database Connection** button to verify that the database connection is valid. If an error is displayed, check the credentials, server and database, and re-test.
5. Once the database connection tests valid, click **Save Database Connection**.
6. Repeat steps 2-5 for each of the connection strings in use.
7. Close the **Configuration** form. A popup will offer you the opportunity to close RBDMS WinAdmin.NET and reopen with the new settings in effect. Click **Yes** to close.
8. The new settings will be in effect when you reopen the application. If you including a user name and password in the connection string, you should consider encrypting the connection string with the Microsoft-supplied

aspnet_setreg.exe utility program. The new connection string is stored as an encrypted value in the application configuration file.

When a user clicks the **Save Database Connection** button on this page, the application checks for the tables that house filters for reports and queries (*Coll*, *CollDevControls*, and *DevControls*) and, if the tables are not present, the application opens the query analyzer so that the user can run a .sql script to build these tables. Clicking the **Build RBDMS Filter Tables** button also will open the script.

Important Note: Make sure that each client installation of the managed .NET application is set to use the same connection string defined for that application within RBDMS WinAdmin.NET. Please see the documentation for the targeted application to determine how to set the database connection. Most .NET data applications use an application configuration file that allows the user to point the client software to a server application.

Defining Other Application Settings

Switch to the **Settings** tab of the **Configuration** form to continue defining the settings for the application being managed by RBDMS WinAdmin.NET. This tab exposes several elements of the underlying **<StatePrefix>_userSettings.config** file for the managed application for editing. This .config file is located in the **C:\Program Files\GWPC\RbdmsWinAdmin\configs** folder.

Setting the Path to Report Templates:

Since the ability to edit report templates should be reserved for administrative users only, the developers suggest that most user installations of the various applications managed by RBDMS WinAdmin.NET point to report templates stored in a centralized location on the server unless the users are inspectors who may not always have access to the central server while they are in the field. To set the path to the shared report template folder, click the **Open Folder**  button in the *Report Path* combo box and navigate to the correct network location. Again, as the database administrator, you may wish to save a developer's set of report templates in a secure location.

Setting the Web Directory:

If the application that you are managing is a Web application, the Settings page will display a *Web Directory* field. Set the path to the Web folder by clicking the **Open Folder**  button in the *Web Directory* combo box and navigate to the correct network location.

Setting the GIS Connections:

If the application that you are managing includes a GIS module, the **Settings** tab will show a *GIS URL* field. Set the URL to the location of the GIS application configuration page in the *GIS URL* field. Be sure to set the GIS mode (either *Network GIS* or *Detached GIS*). In addition, be sure to specify the name of the detached GIS configuration file, if applicable.

Editing the Navigation Form:

If the application that you are managing includes a **Navigation** form, such as that included with the RBDMS Inspection module for Indiana, you can add columns on the displayed data grid and filtering criteria for the search features of the **Navigation** form in various RBDMS.NET applications by editing the **Nav.xml file**, which is installed at the root of the managed program installation folder. The code in the **Nav.xml** file includes a <*Loc*> tag that contains the attributes and elements used to configure the **Navigation** form of the RBDMSWin.Net application. Tags within the <*Loc*> element are as follows:

- ***Col:*** The columns to include in the list view
- ***Field:*** The name attribute of the column.
- ***Width:*** The width of the column (percent of screen width)
- ***Filter:*** If a filter value is set to *Yes* within the <*Loc*> element, that attribute is included in the filter criteria.
- ***Dropdown:*** If a dropdown element is included, along with the SQL command for populating the dropdown, the criteria will be rendered as a combo box instead of the default textbox.
- ***TabKey:*** This element is used to turn certain tabbed pages on or off in the user interface.

Debugging Operations:

The **Configuration | Settings** page also includes two checkboxes labeled **Debug** and **Debug SQL**. These are useful for troubleshooting operations within WinAdmin.NET. For example, when you click the **Debug SQL** checkbox and run a report within WinAdmin.NET, the SQL underlying the report will be copied to the clipboard. From there it can be pasted into Query Builder or a similar tool for debugging.

The **Debug** checkbox is meant to write longer error and exception messages to the event log.

Attaching a SQL Server Security Database

An example of a SQL Server security database that can be downloaded, restored from the backup file, and edited to meet your application requirements is available at http://www.rbdmsonline.org/Downloads/MS/RBDMSSecurity.zip. Again, if you are seeking to manage the RBDMS.NET application, another option is to generate a test database in a server location of your choice and modify the connection strings in the **RbdmsWpf.exe.config** (for the compiled application) and in the app.config file and project settings (RbdmsWpf and RbdmsConfig) for the source code.

To use RBDMS WinAdmin.NET to manage an RBDMS application on a server where there are multiple SQL Server instances, you must establish an alias for the target SQL Server. Creating a server alias depends on what client tools are installed on the computer. If you have SQL Server 2000 client tools, use SQL Client Network Utility and select the *Alias* tab. If you have SQL Server 2005 tools, open SQL Server

Configuration Manager and expand the SQL Native Client Configuration node to find the node for *Aliases*. Most application servers do not have SQL Server tools installed; however, the client network utility is also included in the MDAC pack install. You can run it from DOS by executing **cliconfg** from the command prompt. You will need to provide the *Alias Name* for the server, the *Port No.*, *Protocol* (TCP/IP) and the *Server Name*. You will then enter the name of the alias you created as the Server Name in RbdmsAdmin.NET configuration form.

Within the SQL Server, you will need to create an RBDMS role and then establish user accounts with the ability to login, read, and write data assigned to the RBDMS role. You may wish to create several RBDMS roles with rights and privileges that align with your user groups' business needs. The developers recommend that you establish a separate RBDMS administrator role with rights to the security database.

Reminder: If the SQL Server is on a server within the same domain as your users, you can use trusted connections for your user accounts. However, if the SQL Server is in a DMZ or in a different domain, then you will need to use SQL Server authentication for your user accounts.

Please consult Microsoft's extensive SQL Server help system for more information about this procedure, which can vary greatly between computing environments.

Encrypting Database Connection Strings

RBDMS WinAdmin.NET is installed with default settings to encrypt the database connection strings to the managed applications for greater security. This encryption makes the connection strings invisible on the **Configuration** form.

To decrypt the connection strings, follow these steps:

1. In Windows Explorer, navigate to the installation folder for WinAdmin.NET. By default, this location is **C:\Program Files\GWPC\RbdmsWinAdmin**. Open the **\configs** folder.
2. Use Visual Studio to open the **<StatePrefix>_userSettings.config** file for the managed location.
3. Add the following setting:

```
<setting name="DoNotEncryptConnectionStrings" serializeAs="String">
<value>True</value>
</setting>
```

If this setting is missing, or set to *False*, then the connection strings are encrypted.

Managing Users, Roles, and Rights

RBDMS WinAdmin.NET has been designed to give you detailed control over which users are given access rights to the various objects, such as menus and forms, and permissions to perform various actions or to see particular data views. Where access is not allowed by security group definitions, the application has been developed to keep those objects from being available to those users. In addition, user group assignments determine who has read/write versus read-only rights to those forms

they are allowed to access. In other words, view, form, menu, and data access is filtered by role, user, and group assignments.

You have the flexibility to define the user roles and access privileges at a fine level of granularity and to assign new users to their appropriate groups. Attention to detail in refining the definitions of user classes and their needs is integral to designing application security. The rights and privileges conferred on the users of the applications you manage through RBDMS WinAdmin.NET are based on combinations of *Create, Read, Update,* and *Delete* rights for each defined user group.

Users, roles, and rights are defined on the **Security** tab of the **Menus and Security** form in RBDMS WinAdmin.NET. The form consists of four panes:

1. A *Users* list of application users. Clicking the "+" next to a user expands the subgrid to display roles assigned to the user. User records can be dragged and dropped on *Roles* records to assign the user to the role. The **Create, Edit,** and **Delete** buttons on the **Security** tab toolbar each call a popup governing those actions for user account management.
2. An *Operators for Users* list. This list is useful for eCommerce applications such as eReport and eForm (ePermit) in which a single user may operate as an agent representative for several industry operators. For the *User Name* with the active focus (designated with the arrow  at the beginning of the record) in the *Users* list, you can use the lookup list to associate company names with that user login. The lookup list is defined by a SELECT statement in the target application <StatePrefix> userSettings.config file.
3. A *Roles* list of application user groups. Clicking the "+" next to a role expands the subgrid to display users and rights assigned to the role. Role records can be clicked and dragged onto user records or rights records to assign the role to the user or add the right to the role. To create a new role, enter the appropriate information on the grid row marked with the **New Record** asterisk  .
4. A *Rights* list of application objects and defined privileges. Clicking the "+" next to a right expands the subgrid to display roles that have that right assigned. Right records can be clicked and dragged to the role grid to assign the right to the role. To create a new right enter the appropriate information on the grid row marked with the **New Record** asterisk  . You must create a *Rights* entry for each new object that you create in the target application, such as a report, a form, or (if appropriate to the granularity of control needed) a button or menu item.

The **User Information** tab of the **Menus and Security** form provides a grid that you can use to preview and edit the effective rights of each user. Select the user account name from the dropdown box at the top of the form to display the assigned rights. The contents of the grid can be sorted ascending or descending by clicking the column heading at interest.

Note: The **Menus and Security** form of RBDMS WinAdmin.NET is a good starting place for troubleshooting specific user login difficulties with menu or report or other types of access/object visibility problems. First, check to make sure that the right (application object) is defined. Next, make sure that appropriate roles are assigned

access to the right. Then, make sure that the user having difficulty accessing the object is assigned to a role with rights to the object.

The number of records managed on this form can be extensive. Therefore, the grids on this form can be sorted and filtered to minimize scrolling. You can filter on any of the column heads that display the filter icon  .

1. Click the column head filter icon to drop down a list of available values to filter your view of the data set. Active filters are denoted by a blue icon  .
2. To add operators and conditions to create your own filter, click the filter icon on the column head and then click *Custom* from the dropdown list box. If you select *Custom* from the dropdown list, you can refine your search with a full array of operators (*Like, Equals, Starts with, Contains,* etc.).
3. Complete the query form and click **OK**.
4. To remove a filter that you have placed, click the filter icon and select *All* from the dropdown list.

The records in the grids can be sorted in ascending and descending order by clicking the sort arrow next to the filter icon  in the grid columns.

For a discussion of the **Menus** subpage, please see Managing Menus and Processes.

Managing Menus and Processes

Some menus reflect defined workflow processes, while others are static application menus. Both types of menus can be developed and maintained on the **Menus** form in RBDMS WinAdmin.NET.

All menu items (also referred to as *tasks* when discussing processes) have an associated "action object" to which you assign rights (Create, Read, Update, and Delete). If a user has no read rights to an action object, any menu item associated with that action object will be invisible or disabled based on a configuration setting: *MenuHideMode* (Hide or Disable).

The **Menus** tab of the **Menus and Security** form is used to manage both application menus and workflow processes. The form consists of two panes:

- **Menu/Process Tree Control:** A list of existing menus and workflow processes in the managed application is available from the **Select Menu** dropdown box in the form toolbar. Selecting a menu in the managed application from the list box will display that menu in the left pane.
- **Rights List:** A list of all available objects in the managed application appears in the right pane of the **Menus** form. When you click an item in the tree control, the object (right) associated with the menu item becomes active in the *Rights* grid. Permissions to these objects are assigned by role on the **Security** tab of the **Menus and Security** form.

To create a new menu, follow these steps:

1. Click **New Menu**, enter a name for the new menu in the popup and click **OK**. If you are developing a new workflow process, be sure to toggle the *Process* checkbox to *True*. A new Home node will appear as the start of the new menu tree control in the left pane.
2. To create a hierarchical, nested menu structure, you may need to create the upper-level nodes as rights in the *Rights* grid. You can do that in the *Rights* grid on the **Menus** tab or on the **Security** tab. However, if you create the right on the **Menus** tab, you must remember to grant the CRUD privileges to those objects on the *Rights* pane of the **Security** tab. On either the **Menus** or **Security** tab *Rights* grid, in the **New Record** row  ,

    a. Enter a RightName for the new node.
    b. Tab to the Active checkbox and fill it.
    c. Tab to the RightID, which is auto-assigned.
    d. Tab to the Description and enter the text that will appear on the menu for that item.
    e. Tab to the URL column. If applicable, for a Web application, specify the URL to retrieve the item.
    f. Tab to the WinForm column. If applicable, enter the file name to retrieve the object.
    g. Tab to the RightType. If applicable, enter an association for the right (Report, application description, etc.).
    h. Tab to the Image column. If applicable enter the path to any icon or small graphic that will appear on the menu next to the item.
    i. Tab to the Target column. If applicable, designate the target window for displaying the retrieved object.

    Having created the new node in the *Rights* list, you can then select it and drag and drop it to the correct position on the tree control in the left pane.

4. Add existing objects to the tree control by selecting and dragging objects from the *Rights* list and dropping them to the desired node position on the tree.
    a. Simply selecting and dragging an object from the right pane grid and dropping it onto a menu tree node will create a child node at that position on the tree.
    b. Holding the **Shift** key down to drag objects from the right pane onto the tree control will replace the targeted node in the tree control.
    c. Right-clicking on the tree control will display a context menu with such options as *Add New Child Node, Add New Node Above, Add New Node Below, Copy Node, Cut Node, Paste Node,* and *Delete Node*.
    d. Clicking on any of the tree nodes will activate the associated record for that action object in the right pane.
    e. Set the minimum permissions required for each menu item to be visible in the CRUD pane below the tree control.
5. When you have completed editing your menu, click **Save Updates** so that your changes will take effect in the target application.

You can preview the way that the menu will appear to individual users or groups of users (roles) and thereby check for the proper assignment of rights to the menu items and referenced objects. To do so, select either the user account or the role from the list boxes under *Display Tree* at the top of the menu tree pane.

To edit an existing menu, choose the menu from the **Select Menu** checkbox, make the desired revisions, and then click **Save Updates**. Likewise, you can begin with an existing menu, make any desired customizations and then use the **Save As** button to rename the menu.

The *Rights* grid on this form can be sorted and filtered to minimize scrolling. You can filter on any of the column heads that display the filter icon .

1. Click the column head filter icon to drop down a list of available values to filter your view of the data set. Active filters are denoted by a filled icon .
2. To add operators and conditions to create your own filter, click the filter icon on the column head and then click *Custom* from the dropdown list box. If you select *Custom* from the dropdown list, you can refine your search with a full array of operators (*Like, Equals, Starts with, Contains,* etc.).
3. Complete the query form and click **OK**.
4. To remove a filter that you have placed, click the filter icon and select *All* from the dropdown list.

You can change the look of the rights grid by invoking *Group by…* options. To change the default view, drag and drop a column header from the grid to the space labeled **Drag a column header here** to group by that column above. Then, to undo your *Group by…* selection and revert to the default view, drag the column header back to the data grid. You also can re-order the columns of data presented on the grids by clicking and dragging the column headers.

Setting Personalization Options

Personalization settings will be made available to allow users to set layout, view, and design skin preferences, among other customizable features.

This topic is still in development.

Troubleshooting with the Error and Exception Handling Module

Application exceptions that occur in RBDMS WinAdmin.NET are reported according to Microsoft Enterprise Library protocol. Examples of program exceptions could include the following:

1. Out of disk space
2. Logic errors in the programming
3. Database operation problems such as violating a data constraint
4. Network errors

When an exception occurs, the error is displayed to the user in a message box within the application, and a more detailed explanation of the error is written to the *Application* log in the workstation Event Viewer. From there, the description of the

error can be copied to the clipboard and pasted into a report for the database administrator or developer.

Defining Rule Sets

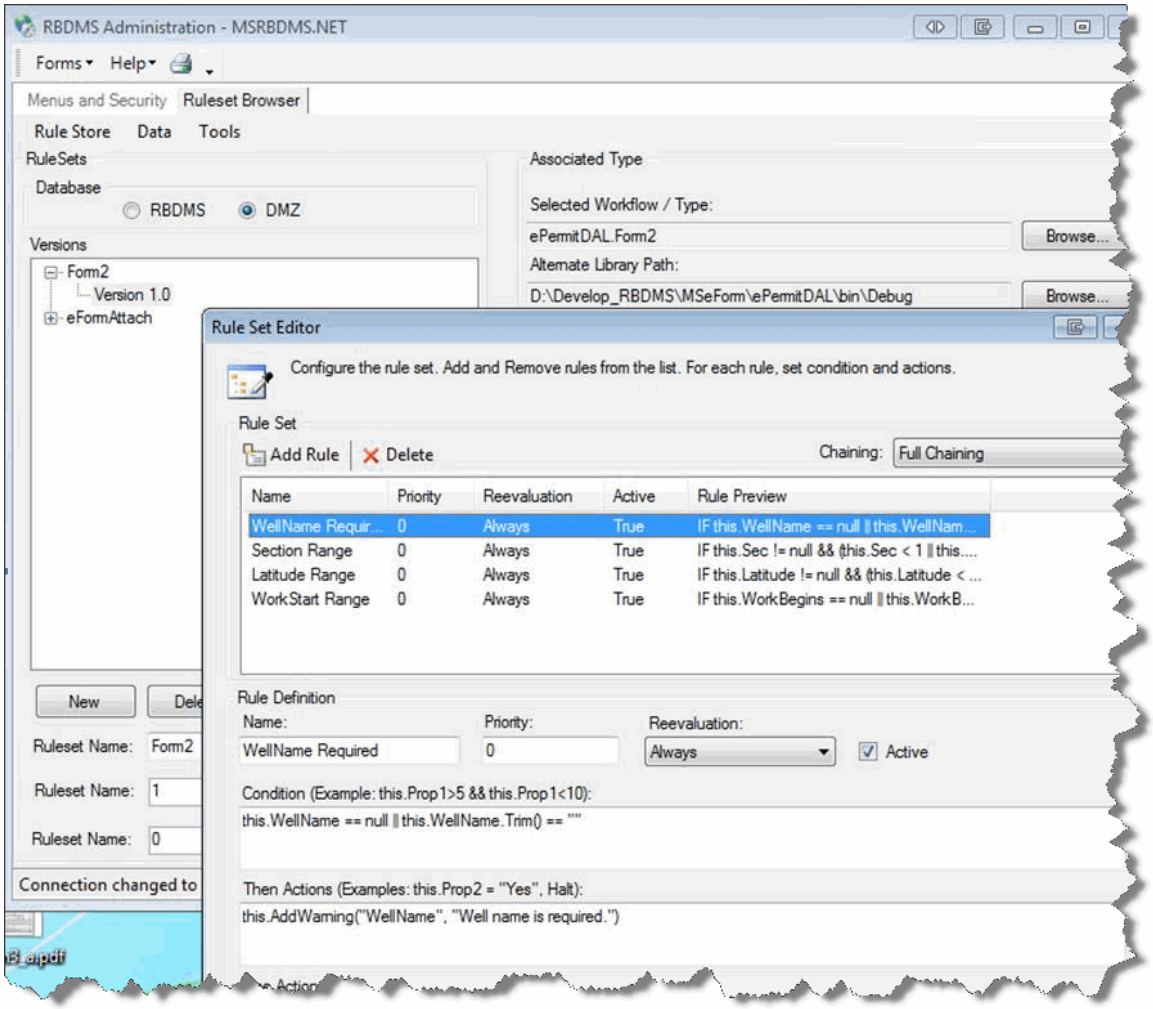
Using the rules engine is contingent on the presence of the **RuleSet** table in the database specified by the connection string to RbdmsDMZ or RBDMS. The RuleSet table tracks, by *RuleID*, the name of the rule, major and minor version numbers, the definition of the rule set, status of the rule, the path to the assembly referenced, the name of the library, and the modify date. The Rule Set Browser includes radio buttons to select the database location for the **RuleSet** table (RBDMS or DMZ). If the RBDMS database does not contain the **RuleSet** table, WinAdmin.NET automatically selects the DMZ database.

The rules engine consists of a Browser and an Editor. To open the rule set browser from the RBDMS WinAdmin.NET menu, click **Forms | Rule Set Browser**. The first time you open the Rule Set Browser, you will need to make sure that you set the path to the appropriate directory containing the .dlls for the data objects by clicking the respective **Browse** button and navigating to the appropriate directory location. This tells the rules engine to point to the object to which the rules you create will be bound. The file locations that you enter in the *Alternate Library Path* field are stored in the application configuration file for WinAdmin.NET, so each user can have a different path to the .dll location.

The tree control in the Versions pane on the left shows the rule set names and, nested below each, the available versions of each rule set. Clicking one of the version nodes displays in the Members pane the methods and attributes for the bound object that are available to the rule set.

The button bar below the Versions pane can be used to create new rules, delete existing rule sets from the list, copy rule sets, and edit the rules within a selected rule set. Clicking the **Edit Rules** button opens the Rule Set Editor as a dialog form.

To create a new rule, click the **Add Rule** button at the top of the Rule Set Editor. Then you can use VB.NET code snippets to define (1) the condition you wish to control with the rule (IF...), (2) the actions that will be taken in response (THEN...), and (3) other actions that must follow (ELSE). A method call is then added in the application program to associate the rule set with the form object. An example of a simple rule is shown below.



Setting Alerts

RBDMS WinAdmin.NET allows you to create and to subscribe to collections of automated messages referred to as *alert sets*. This functionality is now in development. The intent is to allow you to view messages from the database that pertain to any of a variety of events that can be configured. Some examples follow:

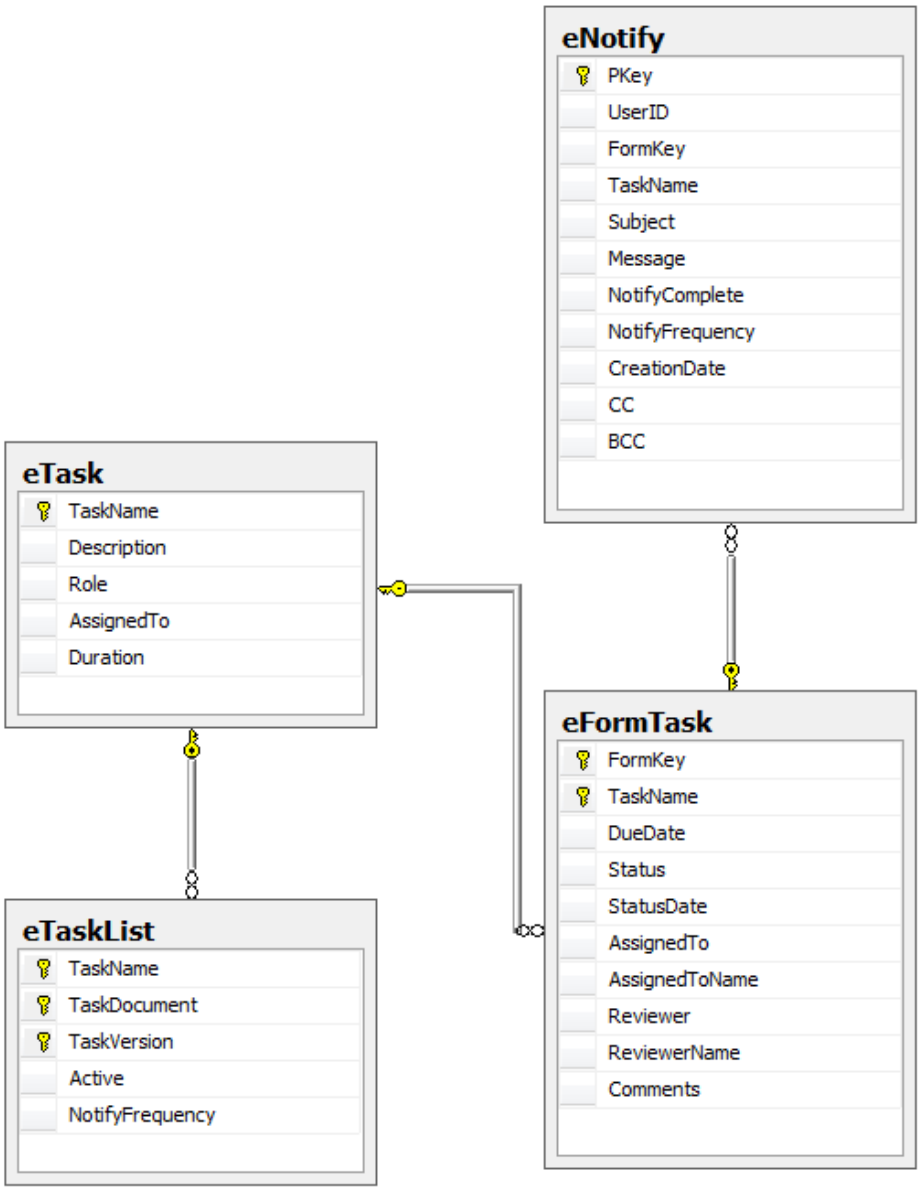
- Parameter exceedances for reported values across the sites, operations, or other types of records you are monitoring that have been associated with your user profile as favorites.
- Notification of events and due dates for the sites you are monitoring as associated with your user profile.
- Reminders associated with the requirements of the agency program to which you are assigned.

The table structure created for managing task data is shown below. The various ways in which it can be applied are diverse. The basic flow for the Colorado implementation of eForms is as follows:

1. Create **eNotify** objects. A variety of methods can be used to develop these eNotify objects: SQL Server stored procedures, scheduled jobs, or .NET procedures forged in the Rule Set Editor. Each notify record has a user key, task name, and message.
2. The application will read **eNotify** on scheduled request, looping through the records. The *Notify Complete?* Boolean will then be set.
3. Within the service, a library has been created for e-mailed notifications. The library can run as a service or as an application.
4. **eFormTask** is the task associated with a specific form, with a key of DocNum and task name. Users can create as many tasks as needed.

The basic flow for users in Mississippi is as follows:

1. Create **eNotify** objects. A variety of methods can be used to develop these eNotify objects: SQL Server stored procedures, scheduled jobs, or .NET procedures forged in the Rule Set Editor. Each notify record has a user key, task name, and message.
2. The application will read the eNotify table on a scheduled request cycle and select records with **NotifyComplete** = *False*. An email notification will be sent to the email address associated with each UserID. The email address is specified when the users' login security information is set up in the RBDMS WinAdmin.NET program. After an email is sent, the **NotifyComplete** field will be set to *True*. The notice will be emailed only once.
3. Notifications are not deleted so that users can run a report of notifications at any time.
4. For an MSOGB user to subscribe to the eForms submitted notifications, the user should be included in the eForm role. The simplest technique for subscribing to notifications is by associating notifications with specific roles. Other techniques may be developed in the future to allow users to opt in and out of specific notifications.



- **eTask:** a table that contains a list of all tasks (short name, description) and associated role.
- **eTaskList:** can be thought of as a "template table" that associates a list of tasks with each distinct form version. eTaskList.TaskDocument="Form 2."
- **eFormTask:** when a user creates a form (an object assigned a DocNum), the application looks through the table structure to derive a task list from the template table for the specific form and version number represented by the object with the DocNum. In other words, the eFormTask table contains one record for each task associated with that specific form.

One example of a way to implement this process as a workflow is the Colorado eForm application, a Web-based permitting application. eForm includes a grid on the **Review** page in the application. This grid is populated according to business rules by LINQ query that runs *onSubmit* of documents. The *onSubmit* query reads the template to see which tasks are associated with the incoming form application type. When the **Review** tab reads the task table, it populates the names of those staff members assigned for each task. Review assignments will be manual. As long as *Assigned to* is in the template record, the data can be populated.

Auditing and Resolving Database Conflicts

The Audit module tracks changes to specific tables in the database to allow for database rollbacks without having to restore the database from a backup. This capability is useful in the event of extended operator error or data concurrency conflicts.

The module will be based on the presence of judiciously selected **_shadow** tables within the database, with changes tracked by *AuditID* in the tables. A graphical user interface for the module is in development.

This topic is still in development.

Managing Reports

Adding Report Definitions

The organization of templates for reports of similar type into a tree control for menu access is controlled through the **Menus** page of the **Menus and Security** form of RBDMS WinAdmin.NET. Once you have created a new report, you can add it to the target application Reports menu by adding a new node, setting the link to the file, and optionally specifying an icon file to display as an image on the tree control, all from the **Menus and Security** form.

As a convention, subreport templates names should be preceded with a lowercase "zz" prefix. With a "zz" prefix, the subreports will display after the top level reports to avoid confusion when browsing for reports.

Reports are usually defined through three components:

1. A query (SELECT, WHERE, ORDER BY) run against a SQL Server database or by a data view within SQL Server. Multiple table adapters that describe a set of these queries or views can be defined as a dataset within a Reports project in VisualStudio. This dataset specifies the data source for purposes of report data binding.
2. A filter that contains elements compatible with the query or view and that is associated with the report.

3. A template with data fields bound to the fields exposed in the table adapter or view.

The **Reports** form in RBDMS WinAdmin.NET offers the ability to manage two types of templates: SQL Reporting Services report language definition client (.rdlc) and extensible markup language (XML) "grid-style" reports.

.rdlc Templates:

The .rdlc templates include formatting and layout options not available in the grid-style reports and can be developed either in Visual Studio or in the SQL Server report designer. The .rdlc format is preferred for intranets and small Web sites rather than the report language definition (.rdl) format typically used with SQL Server Reporting Services installations because many agencies prefer not to run SQL Server Reporting Services, either because they are still running SQL Server 2000 or because running SQL Server Reporting Services is problematic in some other way. However, the **Reports** module can accommodate either format.

The difference between the .rdl and .rdlc formats is that, when the .rdlc format is passed to the report viewer, the <query> element of the Report Definition Language schema is not processed. For Web sites with heavy usage, you should consider using the .rdl format.

Although instruction in the use of Microsoft products is outside the scope of this manual, the topic Hints for Developing .rdlc Reports may be helpful for creating new reports.

Grid-style Reports:

Structure

The grid-style reports are derived from XML files. These grid-style reports can be sorted and filtered with the column header controls and output as Adobe Acrobat, Microsoft Excel, or Microsoft XPS files. An example of the structure of a grid-style report is shown below. The filter set that defines the selection criteria is associated with the report through the **Grid Reports Edit** form, which opens when the report name label on the tree control on the **Report** form is double-clicked.

```
<?xml version="1.0" encoding="utf-8" ?>
- <RBDMSReport>
  <CollName>vwWeeklyReport</CollName>
  <Description>Weekly Report</Description>
  <sql>SELECT PermitDate, TypeDate, PermitStatus, Permit,PermitSubTypeI, Pool, PermitType, Elevation, ApplicationDepth, ApplicationNumber,
```

A tag denoting the type of file...

The collection name that defines the report filter...

A short description of the report...

The sql SELECT command that defines the content of the report...

EntityName, Address1, City, State, PostalCode, CommAddress, Lease, WellNumber, CountyName, Township, TownshipDir, RangeDir, Range, FootageNS, NS, FootageEW, EW, LandType, LandNumber, QuarterQuarterQuarter, QuarterQuarter, Quarter, OrientationType FROM vwWeeklyReport WHERE (TypeDate = 'Issued')</sql>

Close *file type tag.*

</RBDMSReport>

Display Options

You can change the look of the grid-style reports by invoking *Group by…* options. To change the default view, drag and drop a column header from the grid to the space labeled **Drag a column header here** to group by that column above. Then, to undo your *Group by…* selection and revert to the default view, drag the column header back to the data grid. You also can re-order the columns of data presented on the grids by clicking and dragging the column headers. For columns marked with a tiny down arrow  , you also can customize your view of the data grid by selecting new column heads to display by choosing items from the column header dropdown lists marked.

Some agencies have found embedding a URL into a column of grid-style reports helpful for linking information views. This action is not directly supported in the Administration application, but it can be done by editing the underlying XML. An example follows:

```
<RBDMSReport>
  <CollName>Companies</CollName>
  <Description>Active Well Operators</Description>
  <sql>
    <![CDATA[
    SELECT EntityPKey, Operator,
    '<a href="http://somewebsite.com?urlparameter1=' + CAST (operator
as varchar(50)) + '" target="_blank">Click this link</a>',
    StatusDescription, Address1, City, State, PostalCode,Country,Phone
    FROM rptOperators
    WHERE 1=1
    ORDER BY Operator
    ]]>
  </sql>
  <Columns>Company Filter<Column Name="EntityPKey" Width="80"
Caption="Company ID" /><Column Name="Operator" Width="150"
Caption="Company Name" /><Column Name="StatusDescription" Width="50"
Caption="Status" /><Column Name="Address1" Width="100" Caption="Street
Address" /><Column Name="City" Width="100" Caption="City" /><Column
Name="State" Width="50" Caption="State" /><Column Name="PostalCode"
Width="60" Caption="Zip" /><Column Name="Country" Width="80" /><Column
Name="Phone" Width="100" Caption="Phone" /></Columns>
</RBDMSReport>
```

Be sure to wrap the <sql> element in a CDATA tag and make sure that the <a href> information is correct.

Editing Report Templates

Double-clicking the report name label in the tree control on the **Reports** page of WinAdmin.NET will open the report template in **Edit** mode. The .rdlc templates will open in Visual Studio.NET or other compatible software. For the .rdlc report templates, the dynamic filtering parameters associated with each report are referenced as the hidden property *CollName* (accessed in the Visual Studio **Report | Report Parameters** menu call).

Double-clicking a grid-style report label in the tree control will open the template in an XML editor form called the **Grid Reports Edit** form within WinAdmin.NET.

The **Grid Reports Edit** form, which is used to define the grid reports, shows the fields for the path to the template, the title of the report as it will appear in the navigation tree control on the **Reports** form, the filter associated with the report for use as selection criteria, and the .sql query that drives the report results in the top pane. The table columns that are displayed in the report and their attributes are listed in the bottom pane. Column attributes include row, width, caption, VB format command (such as 'd' for short date), the type of control on the report (text or hyperlink), the target form to open, and the link label.

Filters for the grid-style reports are defined in the **Edit Filters** form (**Forms | Edit Filters** menu selection).

Note: The ability to edit report templates should be reserved for administrative users only, and the developers suggest that user installations of RBDMS.NET software point to report templates stored in a central location on the server. An exception to this rule might be inspectors who may not always have access to the central server while they are in the field and who may need to have a local copy of the report templates. Please also see the topic Adding Report Definitions and Creating and Associating Report Filters for more information.

Hints for Developing .rdlc Reports

Hints for Developing .rdlc Reports

This chapter of tips and hints for creating .rdlc-formatted reports for RBDMS applications is not meant to replace the voluminous Microsoft Visual Studio help system, nor should it be construed as anything other than an effort to provide new users some getting-started advice and to familiarize them with some conventions the developers have adopted for these applications.

The steps in creating RBDMS.NET reports are generally as follows:

1. Create the Reports project (or open an existing Reports project) and add a new report.
2. Set the report data sources.
3. Add header and footer information.

4. Develop the report body.
    a. Add groups for sorting, if needed.
    b. Add special formatting, such as background color and images.
5. Create report parameters.
6. Add the report to the application menu.
7. Create and associate filters.
8. Test and debug the report.

Add a Report to a New or Existing Reports Project

Visual Studio includes an option to create a ReportApplication with the MainForm and Report1.rdlc items created for you. When you create a ReportApplication, many of the steps to start a report template are accomplished via a wizard. Although some people find the wizard easier to use, some options you may otherwise exercise are taken away, and since the purpose is to further an understanding of the overall process, the topics in this help file assumes that the wizard is not used. These topics take you through all of the steps that would be accomplished with the wizard as well as the process of integrating the new report into the RBDMS application with the RBDMS WinAdmin.NET application.

RBDMS.NET uses the report language definition client (.rdlc) template format rather than the .rdl format typically used with SQL Server Reporting Services installations. Some agencies prefer not to run SQL Server Reporting Services, either because they are still running SQL Server 2000 or because running SQL Server Reporting Services is problematic in some other way. Therefore, the .rdlc format has proven to be easier to use with RBDMS applications.

Create the Project

1. In Visual Studio, click **File | New Project,** create a new Windows Form Application project, and name the project.
2. Click **File | Save All**. Set the path to the solution folder and click **Save**.
3. From the **Data** menu, select **Add New Data Source**.
4. When prompted to choose a data source type, click **Database** and **Next**.
5. Click **New Connection** and configure the data source for your RBDMS connection. For a local installation of the database, you can use Windows Authentication. Test the connection and click **OK**. Click **Next**.
6. Save the connection string to the app .config file. Click **Next**.
7. Choose the database objects that will provide the bindable dataset for the project.

Add a Report to the Project

1. In the Solution Explorer, right-click the project and select **Add | New Item**.
2. In the **Add New Item** dialog, choose **Report** and name the new .rdlc as appropriate.

Setting Report Data Sources

1. From the **Report** menu, select **Data Sources**.
2. From the **Project Data Sources** in the **Report Data Sources** dialog, select the object to which you would like to bind the report.
3. Click **Add to Report**.
4. Click **OK**.

Note: If the project does not contain any Server Connections or no datasets are defined, Visual Studio will launch a Dataset Configuration Wizard, which will go through the steps necessary to connect to the SQL Server data source. Other data sources besides SQL Server are available, but only SQL Server connections should be used in RBDMS.

Hint: Sometimes while working with report projects, you will find that you need to edit the data underlying a particular report to add fields or to streamline a query. If you must update a table adapter that you are using as the data source for binding data to a report (***<yourdatasetname>*.xsd** in the report project) or if you change an underlying view that the report is based on in SQL Server (e.g., if you wish to include additional fields), follow these steps:

1. Make the desired change in either the dataset in Visual Studio or the view in SQL Server, as appropriate, and save your changes.
2. In Visual Studio, open the report you want to update with the new view.
3. Refresh the data source for the report within Visual Studio.
    a. From the menu, select **Reports | Data Sources**. The **Reports Data Sources** dialog will open.
    b. Select the data source you want to refresh and click **Refresh All**.

Adding Header and Footer Information

Your steps may differ depending on the requirements of your report.

1. From the **Report** menu, select **Page Header** to view the page header space.
2. Drag a textbox from the Toolbox into the header space and enter a title for your report.
3. Set the formatting for your title block in the Properties window for the textbox.
4. Drag four more text boxes to set From: and To: dates for the report run (see Creating Report Parameters).
5. Add footer information: From the **Report** menu, select **Page Footer** to view the page footer space.
6. The footer space can be used to add information about the report. For example, if you would like to view the report run date, drag a textbox from the Toolbox into the footer space and enter *=Now().* If page numbers are desired, drag a textbox from the Toolbox into the footer space in the desired position and enter *="Page " & Globals.PageNumber & " of " & Globals.TotalPages* to format page numbers as "Page x of x."

Developing the Report Body

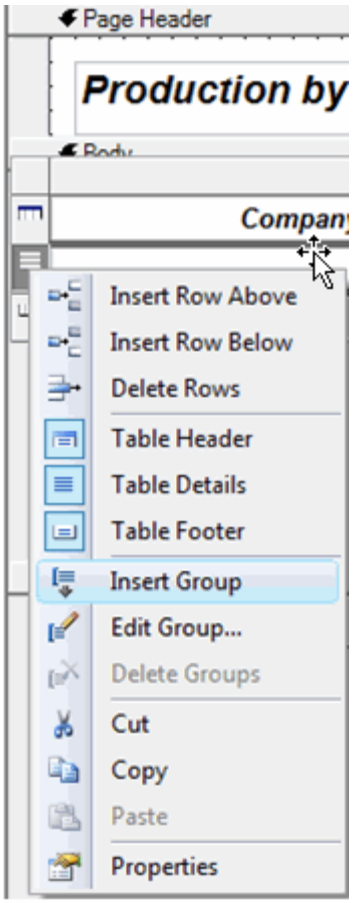
The first control you drag onto the Visual Studio work surface should ***always*** be a List box, a Table, or a Matrix control. Build the body of the report inside the list box or within the table. List boxes and tables allow the report to "grow," so if you fail to build the report with one of these controls, the report will stop printing after the first record. While it is not impossible to add a list box after the template is in a state of advanced development, it is awkward and, in short, no fun.

For example, assuming that you are working with tabular information:

1. Drag a table control onto the main field.
2. Format the grid to include the appropriate number of columns and enter the headings for the report.
3. Add desired formatting to these column headers and text alignment through the Properties pane.

Adding Groups for Sorting

Many reports include tabular totals that require specifying a grouping and sort order in spreadsheet fashion. For these reports, you must be sure to specify the sort/group on the **Grouping and Sorting Properties** dialog, which is available only as a right-click from the line in the table you are grouping on within the Visual Studio workspace.

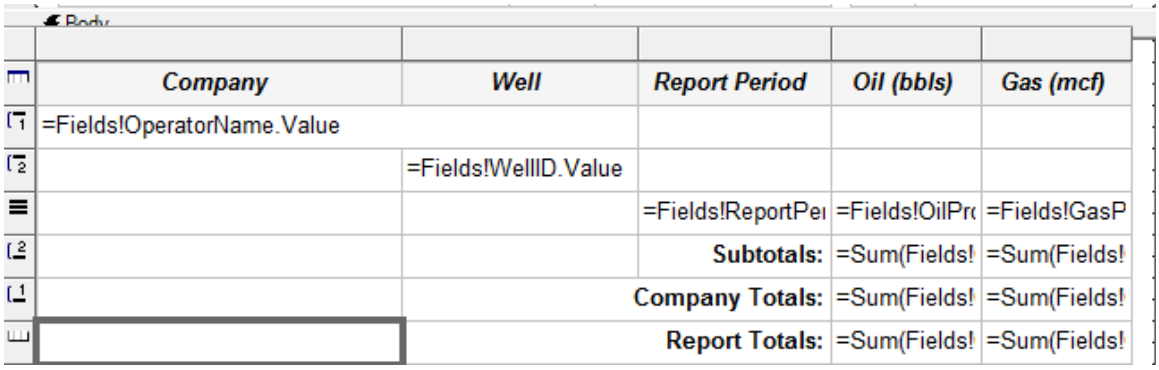


To insert a group:

1. Select the table row where you want to create a group, right click, and click **Insert Group**.
2. In the **Grouping and Sorting** dialog box, on the **General** tab, **Name** field, enter the name of the group you wish to create.
3. In the **Group on:** grid, select the field you want to use from the dropdown.
4. Switch to the **Sorting** tab. Under **Expression**, select the same field value you used in step 3. The **Direction** will default to Ascending. Change it if you wish.
5. Click **OK**.
6. Select the cell under the desired column header in the row marked with the group 1 icon and, in the textbox **Properties**, **Value** field, select from the dropdown list *=Fields!<fieldNameforGrouping>.Value*.
7. In the details row (delineated with the row header icon), under the appropriate column header, bind the field to the data by typing in the cell or selecting *=Fields!<dataFieldYouChoose>.Value* from the Value field in the **Properties** pane formatted for a short date (d) display. You can specify the format for these fields with an Expression in the Properties pane.
8. Add a totals row for the first group: On the row marked with the group totals row header, enter the words *Totals:* and then, for each column you wish

to total, enter code to summarize the data in the following format: =Sum(Fields!<fieldname>.Value) in the **Properties** pane. You can specify the format for these fields with an Expression, e.g., n0 for a number with no decimals) in the **Properties** pane.

You may add as many groups to your report as needed. The following image shows a report that contains two groups and a report summation:



Adding Background Color

For long grid reports, you may wish to introduce some eye-relief. To alternate background color on the report grid, use an "iif" expression for the BackgroundColor property for the row where you want the alternating color to begin, like so:

=iif(RowNumber(Nothing) Mod 2, "AliceBlue", "White")

Embedding Images

Adding an image to an .rdlc is a two-step process: embedding the asset and then calling it from the Properties pane. Adding an image via the Embedding option will copy the content of the selected image into the .rdlc report.

Although it is also possible to create a "Link" to an external image to be used by the report, additional programming is required and the *EnableExternalImages* property must be set to true on the RBDMS ReportViewer. Therefore, this method is not directly supported.

1. From the **Report** menu, choose **Embedded Images** to open the **Embedded Images** dialog box.
2. From the new record row, click **New Image** and navigate to a logo or other small image on your computer and select it. You can add more images at this point if you need to.
3. Then click **OK** to close the **Embedded Images** dialog.
4. Drag the Image control from the toolbox to the desired position on the workspace.
5. In the **Properties** pane, *Source* field, choose **Embedded** from the dropdown.
6. In the **Properties** pane, the *Value* field will be populated with a list of the images you embedded. Select the one you want.

Controlling the Layout

If you find that you are having problems with the layout stretching horizontally off your targeted paper size for printing, be aware of two factors in your debugging:

1. Make sure you turn OFF the "allow to grow" option on the individual control properties on your report unless that is desirable behavior.
2. The reality is that an .rdlc template is basically a text file. Controlling the placement of elements is often a matter of patient trial and error.

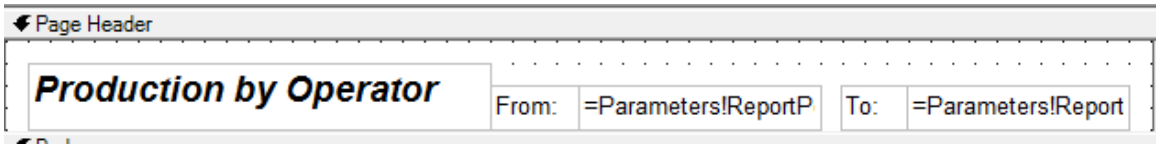
Creating Report Parameters

As a convention, the RBDMS.NET developers use the report parameter *CollName* to specify the name of the filter set created on the **Edit Filters** page in WinAdmin.NET. The report parameter *CollName* should be added to the dialog box accessed by the Visual Studio **Report | Report Parameters** menu call. The filter set name should be similar enough to the report name that it can be easily identified as being associated with the report when you are working with it from within WinAdmin.NET.

Any other report parameter that you define to be passed to the .rdlc (e.g., start and end dates, etc.) also must be defined as a filter control in WinAdmin.NET and assigned to the CollName (NameYouChose) filter set on the **Edit Filters** page of WinAdmin.NET.

So, assuming that you wish to add report start and end dates to a report you have created, you could follow these general steps:

1. From the **Report** menu, select **Report Parameters** to open the **Report Parameters** dialog box.
2. Click **Add**.
3. In the **Properties Name** field enter *ReportPeriodStart* and change the data type to DateTime.
4. Click **Add**.
5. In the **Properties Name** field enter *ReportPeriodEnd* and change the data type to DateTime.
6. Click **Add**.
7. In the **Properties Name** field, enter *CollName* as data type=String. Unclick the **Null** checkbox and enter a name for the collection of filter controls that you will develop in WinAdmin in the function field. This is how you associate the filter set with the report.
8. Return to the report header. Create a textbox with the word *From:* in it. Next to that, add a textbox and enter *=Parameters!ReportPeriodStart.Value*. Next to that, add another texbox *To:.* Next to that, a fourth textbox will hold the entry *=Parameters! ReportPeriodEnd.Value*. Specify a short date format (d) for these fields in the *Format* field in the Properties panel. Your page header should look similar to this:



Be sure to save your work.

Adding a Report to an Application Menu

You must make the .rdlc template for your report available to the application you are managing.

1. From the **Start** menu, select **All Programs | GWPC | RBDMS WinAdmin**.
2. In WinAdmin.NET, make sure that your database connection is pointed to your installation of MSRBDMS (or another application that you are managing with WinAdmin.NET) on the **Forms | Configuration Options** page, **Database Connection** tab. If it is not already pointing to the local installation, then reset the connections
3. With Windows Explorer, copy the .rdlc file you created into the folder shown for the report path, as shown in **Configuration | Settings** page.
4. In RBDMS WinAdmin.NET, switch to the **Security** tab of the **Menus and Security** page.
5. In the **Rights** pane, scroll to the **New Record** row and create the right for the new report.
6. Expand the Rights record and assign the *EveryOne* role to the right. Be sure to click off the new record to commit the write action.
7. Switch to the **Menus** tab of the **Menus and Security** page.
8. In the **Select Menu** dropdown, select the name of the menu to which you will add your report.
9. Add the report .rdlc to the menu by selecting the record in the right pane and dragging it to the appropriate node of the menu tree control in the left pane.
10. Click **Save Updates**.

Creating and Associating Report Filters

The dynamic filtering parameters associated with each report are defined in RBDMS WinAdmin.NET on the **Edit Filters** form. These filter controls give you the ability to limit the results returned by the query underlying the report to the dataset of interest.

1. In RBDMS WinAdmin.NET, select **Forms | Edit Filters** to open the two-paned page that is used to manage report filter controls.
2. In the left pane on the **New Record** row, enter the *CollName* you specified as a report parameter in Visual Studio for the report. Enter a description for the filter collection in the **Description** column.
3. Build the filter collection to include any other report parameters you may have specified (e.g., *ReportPeriodStart* and *ReportPeriodEnd*) along with any

other filter control that your users will need to retrieve information. Build the collection by selecting filter controls on the right and then dragging and dropping them on the **Set Name (CollName)** record for OperatorProduction. Right-click the **Set Name** record to preview the filter.

Note: You may delete items from your set on the left pane without deleting the control from the database. However, deleting filter controls from the right pane is a permanent action.

Creating New Filter Controls

If you must create a new filter control on the right pane of the **Edit Filters** page in RBDMS WinAdmin.NET, the following data dictionary snippet may provide useful hints in how to complete each new control record. The snippet shows the columns in the **DevControls** table, which governs the filtering in RBDMS.NET:

Column Name	Data Type	Length	Nulls	Default	Description
ControlName	varchar	50	NO		Unique name for the control (i.e., "Operator")
xml	text	2147483647	YES		Contains the XML serialization of the control object
ColumnName	varchar	50	YES		Column name for this control. Can include table name if needed.
Prompt	varchar	50	YES		The label to create next to the left of the comparison and criteria controls.
DataType	varchar	255	YES		The data type for the criteria ("String","Integer","DateTime","Single","Double")
ControlType	varchar	255	YES		The control type to use ("TextBox","ComboBox","CheckBox","DateCombo","ListBox")
DefaultValue	varchar	255	YES		A default value to use for the criteria.
Tooltip	varchar	255	YES		A tooltip to show when the cursor is over the control. Useful when you need a longer description than will fit in the label area.
CompareItems	varchar	255	YES		A semicolon delimited list of SQL comparison operators to make available (<;<=;=;>=;>;<>;Like)
DefaultCompare	varchar	10	YES		The default comparison criteria (i.e., "=").
HideCompare	bit	1	YES	((0))	If true then the comparison control will be hidden. This is useful if the default value for the comparison control and you don't want the user to change it.
InputMask	varchar	50	YES		A data input mask to use for a textbox control.
MinValue	varchar	50	YES		The minimum allowable value.
MaxValue	varchar	50	YES		The maximum allowable value.
SQL	varchar	1024	YES		A SQL statement to retrieve data to display in a combo or listbox.
DisplayField	smallint	2	YES		The position of the description field (0 based).
ValueField	smallint	2	YES		The position of the value field (0 based)

Column Name	Data Type	Length	Nulls	Default	Description
ListItems	text	2147483647	YES		A semicolon delimited list of items to use in a combo or listbox.
Hidden	bit	1	YES		True if the control is hidden and always used with a defaultValue and DefaultCompare
Format	varchar	50	YES		Format specification
Sort	bit	1	YES	((0))	If true then this field will be available for sorting.
Required	bit	1	YES	((0))	
ParameterOnly	bit	1	YES	((0))	If true this criteria will be passed as a parameter only and not part of the WHERE clause.

For more information about using filter sets with reports, please see the topics Adding Report Definitions and Editing Report Templates.

Testing and Debugging the Report

Either in the application you are managing (preferably a development copy rather than the live application) or in RBDMS WinAdmin.NET, test the report for correctness.

1. From the menu, select **Reports**.
2. Expand the tree control to see whether your new report is visible on the correct node.
3. Test the filter and either run or debug the report.

About Subreports

Subreports must be developed as separate .rdlcs from the main report and should not have a header or footer. A main report can contain many subreports. Subreports also may contain their own subreports if desired, but these must be defined within those subreports (not the main report). As a convention of the RBDMS developers, subreports are always named with the prefix "zz."

You can embed a subreport into the main report body by selecting the Subreport control from the Toolbox pane in Visual Studio. When you insert a subreport control on the main report body, you must specify the data link to the subreport as a named parameter. For example, in MSRBDMS.NET, the subreport zzConstructCasing is linked to the Well Completions report by the **ConstructKey** which is specified as the parameter value *=Fields!ConstructKey.Value* in the **Subreport Properties** dialog. Also in the Subreport Properties dialog, your entry in the **Prompt** field of the linked subreport definition must match the parameter value name (in this example, *ConstructKey* must be entered in the Prompt field). If you forget to link the subreport properly to the main report, the subreport will not work.

Understanding Program References and Application Structure

Configuring Source Code

RbdmsDAL is the data access layer implementation for the Risk Based Data Management System (RBDMS). One of the prime directives of the RBDMS program is to maintain a "core" set of functionality while allowing implementation-specific customization. Ideally, "core" functionality would reside in compiled assemblies to prevent deviation. In practice that is impossible at this time. To prevent deviation from the RBDMS "core" the following provisions and rules have been made for implementation-specific customization:

1. DO NOT EDIT ANY .DBML FILE!!!

The classes modeled by the DBML represent the "core" data access and MAY BE REPLACED AT ANY TIME! Following are instructions for the foreseen use-cases of adding database fields and implementing business logic.

2. Adding Database Fields

Each **[Name]_DataClasses.DBML** file has a corresponding **[Name]_Custom.vb** file allowing addition of database fields. An example is the **Well** entity. The source code includes a **Well_DataClasses.dbml** and a **Well_Custom.vb**. The **Well_Custom.vb** file is pre-filled with partial classes for the datacontext and each entity defined in the **Well_DataClasses.dbml** file. To add a database field, you would add code to the appropriate entity class in **Well_Custom.vb**. As follows:

```
Partial Class Well
      Partial Private Sub OnTitleChanging(ByVal value As String)
    End Sub

    Partial Private Sub OnTitleChanged()
    End Sub

    Private _Title As String

    <Column(Storage:="_Title", DbType:="VarChar(50)")> _
    Public Property Title() As String
        Get
            Return Me._Title
        End Get
        Set(ByVal value As String)
            If (String.Equals(Me._Title, value) = False) Then
                Me.OnTitleChanging(value)
                Me.SendPropertyChanging()
                Me._Title = value
                Me.SendPropertyChanged("Title")
                Me.OnTitleChanged()
            End If
      End Set
    End Property
```

End Class

3. Adding Business Logic

Each **[Name]_DataClasses.dbml** file has a corresponding **[Name]_BLL.vb fil**e allowing addition of business logic. Again, for example, for the Well entity, the source code includes a **Well_DataClasses.dbml** and a **Well_BLL.vb**. The **Well_BLL.vb** file is pre-filled with partial classes for the datacontext and each entity defined in the **Well_DataClasses.dbml** file. Each partial class is already coded to inherit from **RbdmsBasBusiness.BaseBusiness** to facilitate further customization and future enhancements. To add business logic you would add code to the appropriate entity class in **Well_BLL.vb**. As follows:

```
Partial Class WellAKA
    Inherits RbdmsBaseBusiness.BaseBusiness

    Private Sub WellAKA_PropertyChanged(ByVal sender As Object, ByVal e As
System.ComponentModel.PropertyChangedEventArgs) Handles Me.PropertyChanged
        Me.IsDirty = True
        ExecuteValidationChecks()
    End Sub
    Private Sub ExecuteValidationChecks()
        Me.CheckAlias(Me.Alias)
        Me.CheckAliasType(Me.AliasType)
    End Sub
    Private Sub CheckAliasType(ByVal value As String)
        If String.IsNullOrEmpty(value) Then
            Me.AddError("AliasType", "Alias type is required.")
        Else
            Me.RemoveError("AliasType")
        End If
    End Sub
    Private Sub CheckAlias(ByVal value As String)
        If String.IsNullOrEmpty(value) Then
            Me.AddError("Alias", "Alias is required.")
        Else
            Me.RemoveError("Alias")
        End If
    End Sub
End Class
```

Index