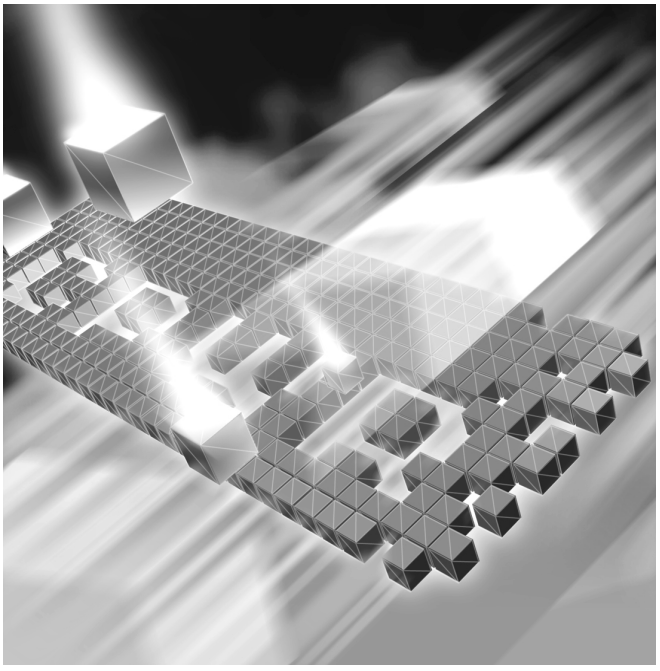


TrackRecord

Administrator's Guide

Release 06.03.00

COMPUWARE

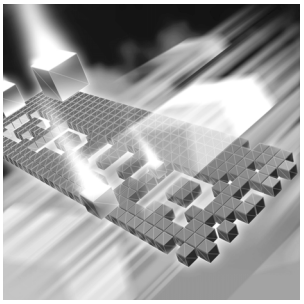
Technical support is available from our Technical Support Hotline or via our FrontLine Support Web site.

Technical Support Hotline:
1-800-538-7822

FrontLine Support Web Site:
http://frontline.compuware.com

Doc. CWQTAXX6F
March 23, 2007

Table of Contents



Chapter 3

Setting Up Groups, Team Members, and Users

Chapter 4

Administering TrackRecord

Chapter 5

Administering AutoAlert

Chapter 6

Administering WebServer

Appendix A

Using ActiveX

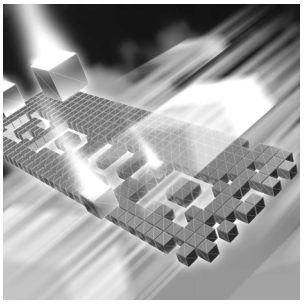
Appendix B

Using Tags

Glossary

Index

Preface



About This Guide

The *TrackRecord Administrator's Guide* is intended for those responsible for configuration and administration of the product. You can find additional information about TrackRecord in the online help facility and documentation set. This guide includes the following chapters:

- Chapter 1, "Getting Started"— Overview of TrackRecord and a checklist of administrative tasks that must be performed before TrackRecord can be used.
- Chapter 2, "Creating and Maintaining TrackRecord Databases"— Description of the actions required to create and maintain the TrackRecord database.
- Chapter 3, "Setting Up Groups, Team Members, and Users"— Guidelines and procedures for configuring groups, Team Members, and users of TrackRecord.
- Chapter 4, "Administering TrackRecord" — Guidelines and procedures for configuring projects, workflow, global preferences, types, and rules.

- Chapter 5, "Administering AutoAlert"— Description of various administrative tasks that must be performed to configure and maintain AutoAlert.
- Chapter 6, "Administering WebServer"— Description of various administrative tasks to maintain the TrackRecord WebServer.
- Appendix A, "Using ActiveX"— Reference material for programmers writing modules to access the TrackRecord databases.
- Appendix B, "Using Tags"— Reference material for administers creating types and fields.

Compuware assumes that you are familiar with basic Microsoft Windows navigation. If this is not the case, familiarize yourself with the documentation for Microsoft Windows before reading this guide.

Product Enhancements

For a detailed listing of product enhancements made in TrackRecord Release 06.03.00, refer to the "What's New Section" in the TrackRecord Release Notes. In addition to the most recent release, the Release Notes also list enhancements included in several previous releases of TrackRecord for your reference.

Reference Material for Programmers

For detailed reference material for programmers writing modules to access the TrackRecord database, refer to the "Object Reference" and "Using Tags" topics in the TrackRecord online help or "Using ActiveX" on page 125 and "Using Tags" on page 167.

Related Publications

In addition to this guide, the TrackRecord documentation set includes:

- *Installation Guide*— PDF file containing system requirements and instructions for installing and configuring Compuware products.
- *Licensing Guide*— PDF file containing licensing procedures and overview.
- The TrackRecord **online help**— Field descriptions, operating procedures, and reference information relating to TrackRecord. In the Web-

Server, click **Help** to see an outline of available topics. In the Windows Client, click **Help** or press **F1**.

- TrackRecord **Release Notes**— HTML file that contains technical information that may affect how you use the product and known issues related to the current release of TrackRecord. You can find the Release Notes from the TrackRecord program files.

You can access the online versions of the PDF files from the installation media browser or from Compuware's FrontLine technical support Web site at: http://frontline.compuware.com.

Viewing and Printing Online Books

TrackRecord's online books are provided in PDF format, so you need Adobe Acrobat Reader to view them. It is recommended that you install Acrobat Reader 4.0 or above to get the best results when viewing the online books. To install the Adobe Acrobat Reader, go to Adobe's Web site at www.adobe.com.

Because PDF is based on PostScript, a PostScript printer is the most reliable way to print the online books. In most cases, you can also print PDF files to PCL printers. If you cannot print the PDF files to your printer, refer to Adobe's Web site for trouble-shooting information.

World Wide Web Information

To access Compuware Corporation's site on the World Wide Web, point your browser at http://www.compuware.com. The Compuware site provides a variety of product and support information.

FrontLine Support Web Site

You can access online technical support for Compuware products via our FrontLine support Web site at http://frontline.compuware.com. FrontLine provides fast access to critical information about your TrackRecord product. You can read or download documentation, frequently asked questions, and products fixes, or email your questions or comments. The first time you access FrontLine, you are required to register and obtain a password.

Getting Help

At Compuware, we strive to make our products and documentation the best in the industry. Feedback from our customers helps us maintain our quality standards. If you need support services, please obtain the following information before calling Compuware's 24-hour product support hotline:

*Tip: To display the TrackRecord About window, choose **Help>About Compuware** TrackRecord. Click **OK** when you are finished.*

- The name and release (version) number of the TrackRecord product. This information is found on the covers of the product documentation or the About Compuware TrackRecord window.
- Installation information including:
  - Installed options
  - Whether the product uses local or network databases
  - Whether it is installed in the default directories
  - Whether it is a standalone or network installation
  - Whether it is a client or server installation
- Environment information, such as the operating system and release on which the product is installed, memory, hardware/network specifications, and the names and releases of other applications that were running.
- The location of the problem in the TrackRecord product software, and the actions taken before the problem occurred.
- The exact product error message, if any.
- The exact application, licensing, or operating system error messages, if any.
- Your Compuware client, office or site number if available.

TrackRecord Technical Support

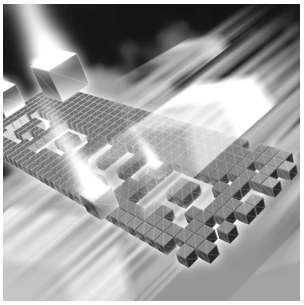
Compuware Corporation

One Campus Martius

Detroit, MI 48226-5099

1-800-538-7822

Chapter 1
Getting Started



This chapter provides a summary of Compuware TrackRecord. Because it provides an overview of TrackRecord functions and concepts, read this chapter carefully before you use TrackRecord for the first time.

Overview of TrackRecord

Compuware TrackRecord is a defect and project tracking tool designed to record and report information about products being developed or supported. TrackRecord records all the information relating to a project—project Team Members and testers, schedules and milestones, bug reports and feature requests in a database. You can then use TrackRecord's query and reporting features to retrieve and format the information you need to keep your project on track.

TrackRecord is customizable and can adapt to virtually any environment. With TrackRecord, you can track many types of information, implement full workflow to ensure a repeatable process, create multiple reports, and even generate graphs.

Product Integration

TrackRecord is part of the Compuware NuMega DevPartner Studio suite of software debugging tools and the Compuware *QA*Center family of automated testing tools.

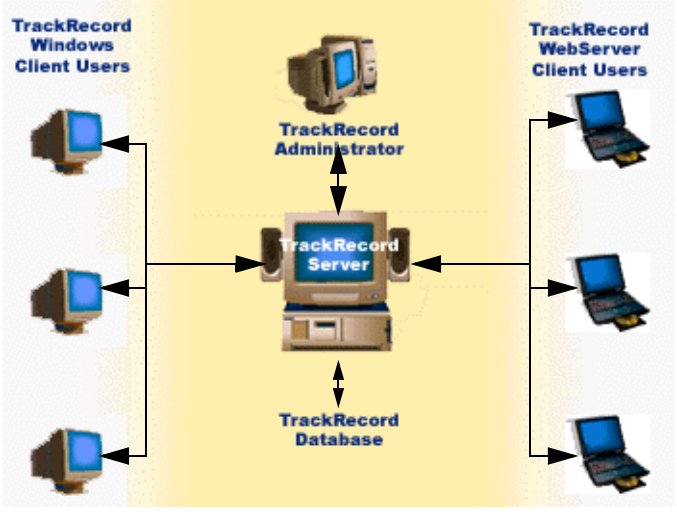
Types of Users

Figure 1-1 illustrates how TrackRecord users, administrators, and WebServer users interact with the TrackRecord server and database.

If you have a license to use TrackRecord on multiple PCs, you can install TrackRecord on a dedicated PC to act as your server. The server will house the database and will share it with the client workstations.

As information is entered in TrackRecord, the information is stored in a database. Interaction with this database occurs through queries and reports. For more information about the database, see "Database Overview" on page 21.

Figure 1-1. TrackRecord Users and Components



There are three types of TrackRecord users:

- TrackRecord **Administrator** — TrackRecord includes administrative functions that can only be performed by authorized users, such as setting up user groups, importing and exporting information to and from other applications, and editing the data types used by TrackRecord. When a TrackRecord administrator logs in, he or she sees an additional menu (the Administrator menu) on the TrackRecord menu bar. For additional information, see "Administering TrackRecord" on page 15.
- TrackRecord **Windows Client User** — A user running the TrackRecord client software accesses the TrackRecord database by logging in from client machines anywhere on your network. Whether a user can add or modify database information, including items and reports, depends on the groups to which the user belongs. The

administrator determines the access and editing privileges of each user.

- TrackRecord **WebServer User** — WebServer users access the database by logging in using standard Web browsers, such as Microsoft Internet Explorer. For a description of administrative tasks for the WebServer, see Chapter 6, "Administering WebServer".

Administering TrackRecord

A TrackRecord administrator is usually responsible for creating the key components of a working TrackRecord project. To guarantee the security of project data, TrackRecord restricts certain activities to users with administrative privileges.

Privileges

Administrator privilege includes the ability to:

- Access the Administrator menu in TrackRecord
- Globally share queries, Outline Reports, and templates
- Delete queries, Outline Reports, and templates owned by other users
- Change template ownership
- View all the types defined for a given database with the Item Browser
- Set the Home Page and default favorite reports and types for users
- Define workflow for defects, tasks, and other items

Getting Started Checklist

The following checklist is simply a suggested order to follow for configuring TrackRecord. If you are new to TrackRecord, perform the administrative tasks in the order presented in this chapter. After you've worked with

TrackRecord for a period of time, you may discover a sequence of steps that more closely matches your preferred style.

Table 1-1. Getting Started Checklist

✓	Item	Comment
	Document processes that will be managed with TrackRecord such as defect resolution, tracking of enhancement requests, or task management.	See "Planning a Workflow" on page 61.
	Determine if you require multiple databases, how many, and a naming convention.	See "Multiple Databases" on page 24 for a discussion of advantages and disadvantages of multiple databases.
	Define the groups to be set up in the system.	See "Planning Group Membership and Access Rights" on page 47.
	Define the Team Members.	See "Team Member Administration" on page 50.
	Create a naming convention for user identifications.	See "Naming Conventions for Users" on page 53.
	Determine which access privileges each group requires.	See "Types of Access Rights" on page 48.
	Plan your share groups.	See "Share Group Administration" on page 54.
	Plan the layout and composition of the forms you intend to use such as defect and project forms.	See "Planning the Composition and Layout of Forms" on page 80.
	Create an initial database.	See "Database Overview" on page 21.
	Edit data types such as defect, project. All editing should be complete before adding any data.	See "Creating and Modifying Types" on page 82.
	If you will be importing data, establish a duplicate abbreviation system.	See "Identifying Duplicate Imported Items" on page 78.
	Enter groups into TrackRecord.	See "Creating or Modifying a Group" on page 49.
	Establish Projects.	See "Creating a Project" on page 58.

Table 1-1. Getting Started Checklist

	Item	Comment
	Establish share groups.	See "Creating a Share Group" on page 55.
	Add Team Members and users.	For more information, see "Creating Team Members" on page 51 and "Creating a User" on page 53.
	Create a workflow. Create a naming convention for the workflow if multiple workflows will be used.	See "Creating a New State" on page 65.
	Enable a workflow for other data types if needed.	See "Applying Workflow to Types" on page 68.
	Establish a default status for new defects and other workflow enabled data types.	See "Default Status for New Items" on page 71.
	Test your workflow rules.	See "Testing Workflow Rules" on page 71.
	Create or edit queries.	See "Creating Global Queries and Outline Reports" on page 72. For an overview of queries, refer to the *TrackRecord User's Guide*. The basic query set should include, but is not limited to: Priority queries (Priority 1, Priority 2, etc.), Status queries, Assigned to current user, and Entered by current user.
	Create or edit Outline Reports.	See "Creating Global Queries and Outline Reports" on page 72. For an overview of Outline Reports, refer to the *TrackRecord User's Guide*.
	Change the default administrator user options for the Favorites tab.	See "Creating Global Favorite Types, Reports, and Home Pages" on page 73.
	Add additional rules to further define policies and procedures.	See "Adding a Rule" on page 96.
	Customize miscellaneous items.	See "Creating Global Templates" on page 74.

Table 1-1. Getting Started Checklist

Item	Comment
Customize the Item Browser.	See "Creating and Modifying Types" on page 82.
Configure AutoAlert.	For information on using AutoAlert, see Chapter 5, "Administering AutoAlert".

Starting TrackRecord as an Administrator

1. Click the taskbar's **Start** button and choose TrackRecord from the TrackRecord program files.
2. When the login dialog box appears:
   - ◇ Enter `admin` in both the **Name** and **Password** fields (lowercase only for Password).
   - ◇ Enter a database name. The default database name was specified during TrackRecord installation.

Note: If you are using multiple databases, you will perform Administrative tasks for each database independently.

3. Click **OK**.

Required: Change the administrator password immediately, as described in the *TrackRecord User's Guide*.

Restricting Access To a Single Database Server on the Client or WebServer

TrackRecord allows users access to multiple databases, by default. Administrators can elect to give users access to only one database server by changing a setting.

1. Click **Start>Run**. In the field type *Regedit*. The Registry Editor appears.
2. In the left pane expand HKEY_LOCAL_MACHINE.
3. Expand Software.
4. Expand Compuware.

5 Select TrackRecord.

6 Right-click the right pane and choose **New>DWORD Value**. A new key appears.

7 Rename the key "MultiServer".

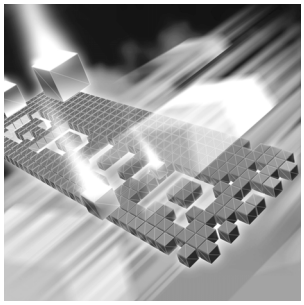
8 Double-click the MultiServer key. The **Edit DWORD Value** dialog box appears.

9 In the Value data field type “0”.

10 Click **OK**.

Chapter 2

Creating and Maintaining TrackRecord Databases



Database Overview

Understanding how the TrackRecord database records, links, and displays information is an important step toward using the product more effectively.

When you enter information using TrackRecord, the information is stored in a database. TrackRecord automatically opens its database when you start the program, and automatically closes your connection to the database when you exit. You interact with this database through item views, queries, and report views.

TrackRecord uses a customizable database so it can be easily reorganized, managed, updated, and accessed in a number of ways.

During installation, you are prompted for a database name. If the database does not exist, the TrackRecord installation program creates a database using the supplied name. This database contains sample types, queries, and views that can be used to familiarize users with TrackRecord.

Once TrackRecord is installed (as described in the *Installation Guide*) and configured (as described in Chapter 1, "Getting Started"), users can begin to generate and accumulate project data. Administrators will then need to protect that data by managing the database.

Types, Items, Links, and Fields

The TrackRecord database organizes information into *types*, such as Person, Company, Defect, or Project. For each type of information stored in the TrackRecord database — defects, tasks, Team Members, and so on — you will want to store different pieces of information. Each data type functions like a database table, establishing the properties (fields) TrackRecord expects for an item of that type. Visually, a data type is a form, with fields and controls to enter or display information. These types are extremely customizable and many different fields can be added or removed very easily. For example, you might want to store a name and phone number for a Team Member and a summary and description for a defect.

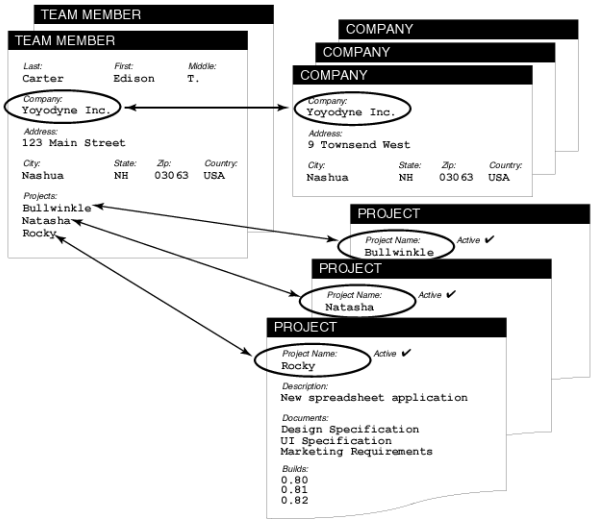
A data *item* is one record of a specific type that has its own data entry form, which is composed of fields appropriate to that item type. For example, Abraham Lincoln and Harry Truman would be two items of the type Person in a TrackRecord database.

Each data item consists of *fields* and *links*. Fields hold specific facts about an item, such as a person's address or the name of a project; links connect one item in the TrackRecord database to another item to capture the relationships between items. For example, a Team Member item (John Doe) might be connected to Project items (as a Team Member), Defect items (as the person reporting them or responsible for fixing them), and Company items (as an employee).

Figure 2-1 illustrates how a Team Member item links to a Company item and to several Project items. These links make it easy to retrieve informa-

tion, such as all Team Members assigned to a project or all projects with open defects.

Figure 2-1. Links Between Database Items



When an object is entered in a field, that field is automatically *linked* to the original item. When you change the item, no matter where it appears, all references to that item change *automatically*. For example, if you change the spelling of the person's name, that change is reflected in all other items that contain a link to that Person.

In addition to the obvious information types, such as Projects or People, the TrackRecord database can store special types of information, such as executable commands and database queries.

Change History

As database items are created and modified, TrackRecord captures the time and date of each change, the person adding or changing the item, and the information that was added or changed.

Client-Server Technology

The TrackRecord database resides on a central server on your network. The TrackRecord server software accepts simultaneous input from multiple users on client machines; the TrackRecord server periodically checks the database for new information during a session and forwards updated information to clients when necessary.

Multiple Databases

One of the first tasks of an administrator is to determine if multiple databases are required. When planning databases, careful consideration of the needs of each organization is required. For example, an organization may establish a database for production defects and a separate database for a large development effort if the workflow differs for each venture.

One of the major advantages of multiple databases is that each database can be customized differently for its user base and workflow. Some of the disadvantages of multiple databases are the loss of centralized reporting and the individual planning and customization that is required for each database.

Best Practices for Maintaining TrackRecord Data

This section was created to assist TrackRecord administrators in maintaining the TrackRecord databases used at their organizations.

Backing Up Information

As with any data that is important to your organization, it is essential to perform nightly backups. TrackRecord's backup utility is intended as an extra fail-safe, and currently keeps only the most current backup. For a primary backup, Compuware highly recommends that you use your own approved backup utility for your production databases. To do so, simply include the `Compuware\TrackRecord\Databases` subdirectory in the list of directories to be backed up. When using any third party utility to back up TrackRecord data, ensure that users are not accessing the TrackRecord database.

Maintaining Data Integrity

TrackRecord includes two features that help maintain data integrity:

- **Check Database—** A data integrity check done by TrackRecord, it checks all cross-links, data values, and other references. Then, it repairs anything suspect.
- **Rebuild—** A data integrity action that performs in the same manner as a relational database "rebuild indices" operation by rebuilding the indices for each of the database tables. As items are changed/deleted/added, "empty" space can build within a table. The compact option

provided during a rebuild removes any unused space within the tables resulting in a reduced physical size.

Both of these features can be automated through the Database Administration Utility.

Guidelines for Using Check Database and Rebuild

- For optimum data integrity, an organization should ensure that the Check Database function is run a **minimum** of once per week. Check Database can be scheduled to run without user interaction. This is intended to make it easier to run the utility on a regular basis at times when the interruption in database access will have minimal impact to the users. See "Checking Databases" on page 27.
- For optimal performance on large databases, it is generally recommended that you rebuild/compact the database at least once per week. See "Rebuilding Databases" on page 29.
- Compuware recommends following each Check Database immediately with a Rebuild.
- When using any third party utility to back up TrackRecord data, ensure that users are not accessing the TrackRecord database. Refer to "Scheduling Backups" on page 30 for more information.
- When making modifications to the types via the Type Editor, it is always good practice to:
  - Make a backup or image of the database before using the Type Editor.
  - Ensure all users have exited the database.
- When using the Workflow Editor, ensure that all users have exited the TrackRecord application, both client-server and Web.
- TrackRecord's backup utility is intended as an extra fail-safe, and currently only provides the ability to restore from the most current backup. If problems occur, restore your last backup. For a primary backup, Compuware highly recommends that you use your own approved backup utility for your production database. When using a backup utility, Compuware recommends that you shut down the TrackRecord server. If this is not possible, ensure that no live sessions of TrackRecord are occurring.
- Prior to performing a database backup, you must manually shut down AutoAlert and then restart it once the backup is complete.

Defining Data

The following guidelines should be followed when defining TrackRecord data:

- Do not add an auto-incrementing field after adding an executable button in the Type Editor as this will result in odd behavior and possible loss of database integrity. To avoid this situation, always ensure that the executable button is the last control added to the type during editing. When editing the type, the executable button should first be deleted from the type and added only after any changes have been made.
- If your organization uses the WebServer, note the WebServer does not recognize the "<", ">", or "&" characters or special characters, (higher than ASCII code 127) in field names or field names in types greater than 31 characters in length.
- Use caution when defining abbreviations. It is possible to create self-referencing abbreviations. For example, you could create type A abbreviation which references type B whose abbreviation references type A, etc. This and other combinations may result in an unstable database.
- Avoid creating child data types that are identical to the parent data type by ensuring that there are additional fields in the child data type.

Optimizing Performance

To optimize performance, ensure that:

- All TrackRecord machines meet the system requirements. For complete system requirement information, consult your *Installation Guide*.
- Applications running on the database server are minimized if possible.
- If connecting over WAN or Internet, use the WebServer not the Windows client.
- TrackRecord Webserver is not architected to benefit from a multiprocessor environment. Instead, scale by adding WebServers.

  If more than 50 users will be accessing TrackRecord through the Web simultaneously, set up multiple WebServers. Attempt to limit the number of concurrent connections per WebServer to 50. You can have as many WebServers as you want to access the same database(s).

- Database maintenance (Check Database, Rebuild) is performed regularly. It should be performed at least weekly as well as every time there is a change to the database schema.

Frequently Asked Questions

Refer to Compuware's FrontLine support web site, http://www.frontline.compuware.com, for answers to your frequently asked questions.

For more information, refer to TrackRecord's online help. To access the help files, open the Database Administration Utility and click the **Help** button.

Using the Database Administration Utility

TrackRecord's Database Administration utility provides the primary tool set for database administration.

Navigating the Database Administration Utility Interface

The Database Administration Utility provides for administration of multiple databases on multiple servers from any client machine. The interface allows you to easily rebuild or run a check on a database or schedule regular checks, rebuilds, and backups.

1. Click **Start>Program Files>TrackRecord>Database Administration Utility** to start the utility.
2. Click **File>New>Server** to add a server to the treeview on the left.
3. Select the database to schedule maintenance for and right-click to select **New>Task**. Alternatively, right-click and select **Check** or **Rebuild** to force immediate action.

Checking Databases

Software developers sometimes experience computer crashes or network problems, such as the failure of a server acting as a router. These failures may occur while data is being submitted to TrackRecord, which may leave the database with incomplete entries. Check Database opens all of the items and corrects any errors it finds, including dates. It also re-links all items that it can.

Note: Compuware recommends that you automate the Check Database feature to run minimally on a weekly basis to ensure that Check Database recovers as much data as possible and places items in a workable state.

If you are running TrackRecord and are logged in to the database to be checked, click **Administrator>Check Database** to perform this task.

To check other databases manually:

1. Click **Start>Program Files>Compuware>TrackRecord>Database Administration Utility.**
2. Select the database to check from the treeview on the left and right-click to select **Check**.

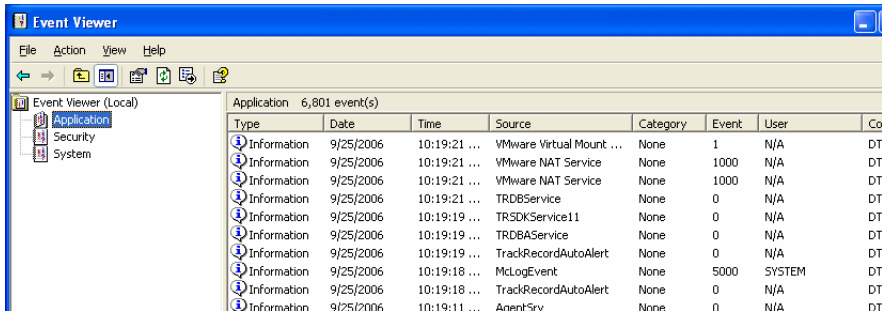
To schedule regular checks:

1. Click **Start>Program Files>Compuware>TrackRecord>Database Administration Utility.**
2. Select the database to check from the treeview on the left and right-click to select **New>Task** and bring up the **New Task** dialog.
3. Select "Check" from **Type**.
4. Enter a **Time** for the check to run in 24-hour notation, such as 06:30 or 18:30.
5. Enter a **Timeout** value in minutes for a threshold for the check.
6. Check which days the check should **Run on**, or select **All days**.
7. Click **OK** to save the scheduled check.

Logging Database Checks

Use Windows 2000, or Windows XP to automatically log the results of a database check. After running a database check, log the results by following these steps:

1. Double click the Event Viewer, accessed in of the following ways:
   - **Windows 2000 —** Click **Start>Settings>Control Panel>Administrative Tools**.
   - **Windows XP —** Click **Start >Control Panel>Administrative Tools**.



2 In the Event Viewer, click **Application log**. The log appears. The check generates three entries. Each entry has TrackRecord as its Source.

3 Double-click on the first log which verifies when the database check began.

4 Double-click the second log which verifies when the database check finished.

Use these values to determine the run length of the check.

5 Double-click the third log, which lists how many of the items are repaired, or verifies that no items are repaired.

Rebuilding Databases

Rebuild is a TrackRecord utility that performs in the same manner as a relational database "rebuild indices" operation by rebuilding the indices for each of the database tables. As items are changed/deleted/added, "empty" space can build within a table. The compact option provided during a rebuild removes any unused space within the tables resulting in a reduced physical size.

Caution: Compuware recommends that you rebuild the database on a bi-weekly basis to preserve database integrity. Regular rebuilds help prevent problems from spreading in the database and irreparable damage to certain database items. For example, TrackRecord creates pointers to attachments that are stored in the database. When damaged, these pointers are difficult to restore.

To rebuild your database index:

1 Click **Start>Program Files>Compuware>TrackRecord>Database Administration Utility**.

2 Select the database to rebuild from the treeview on the left and right-click to select **Rebuild**.

To schedule regular rebuilds:

1. Click **Start>Program Files>Compuware>TrackRecord>Database Administration Utility**.
2. Select the database to rebuild from the treeview on the left and right-click to select **New>Task** and bring up the **New Task** dialog.
3. Select "Rebuild" from **Type**.
4. Enter a **Time** for the check to run at in 24-hour notation, such as 06:30 or 18:30.
5. Enter a **Timeout** value in minutes for a threshold for the rebuild.
6. Check which days the rebuild should **Run on**, or select **All days**.
7. Click **OK** to save the scheduled check.

Scheduling Backups

Protection of data requires frequent and regular backups. Database Administration allows you to schedule daily backups at any time you specify, usually during the overnight period when development team members are not working with TrackRecord.

Caution: **TrackRecord's backup utility is intended as an extra fail-safe, and only one backup can be made for a database on a given day. Database backups are kept indefinitely and must be manually removed to preserve space on the server. The backups are stored in the Backups directory in the TrackRecord installation directory. For a primary backup, Compuware highly recommends using an approved backup utility for the production database. When using a backup utility, Compuware recommends shutting down the server. If this is not possible, ensure that no live sessions of TrackRecord are occurring.**

Scheduling Backups of Databases

1. Click **Start>Program Files>Compuware>TrackRecord>Database Administration Utility**.
2. Select the database to schedule the backup for from the treeview on the left and right-click to select **New>Task** and bring up the **New Task** dialog.
3. Select "Backup" from **Type**.
4. Enter a **Time** for the backup to run at in 24-hour notation, such as 06:30 or 18:30.

5 Enter a **Timeout** value in minutes for a threshold for the backup.

6 Check which days the backup should **Run on**, or select **All days**.

7 Click **OK** to save the scheduled backup.

Caution: A database can only be backed up once a day by the Database Administrator Utility. Subsequent attempts to backup the database will fail unless the previous backup for that day is deleted first.

Changing the Database Name

You can change a database name. To do so:

1 Halt the TrackRecord server. See the *Installation Guide* for instructions for manually stopping the server.

2 In Windows Explorer, navigate to the location of your TrackRecord database folder. By default, it is located in `x:\Program Files\Compuware\TrackRecord\Databases`.

3 Right-click on the database name you wish to modify and select **Rename** from the context menu that appears.

4 Type the name of your modified database name.

Integrating a Customized TrackRecord Database With *QA*Director

How Integration Works

When *QA*Director submits a defect into TrackRecord, it will attempt to import data into two types that are shipped with TrackRecord's default schema: *QA*Center Reported Defect and *QA*Director Test.

TrackRecord's *QA*Center Reported Defect item is specifically designed for logging defects reported by its testing tools. The *QA*Center Reported Defect item contains all the fields found in its parent type, Defect, plus additional *QA*Center-specific fields such as Test Tool, Script Name, and Failure Reason. If your testing reveals a defect in the test application, you can use the *QA*Center Reported Defect item to record the test, tool, and diagnostic information so that the problem can be re-created and fixed.

In *QA*Director's `Custom\win32` directory, there are two integration files for TrackRecord:

- **trqacenter_item.txt** contains the mapping of *QA*Director attributes to tags in the TrackRecord *QA*Center Reported Defect type.
- **trqadirector_item.txt** contains the mapping of *QA*Director attributes to tags in the TrackRecord *QA*Director Test type.

Note: Refer to the *QA*Director documentation for additional information on using this product.

Setting Up Integration

To set up *QA*Director's integration with a customized TrackRecord database, follow these steps:

1. Customize the TrackRecord database. Note that all editing should be complete before adding any data. See "Getting Started Checklist" on page 15.

Tip: For a general overview on tags, see Appendix B, "Using Tags".

2. Ensure that the **QC_Defect** tag is associated with the *QA*Center Reported Defect type or its equivalent. This will identify this type to *QA*Director as the defect type to which it should send its information. See "Creating and Modifying Types" on page 82.
3. Ensure that the **QC_Test** tag is associated with the *QA*Director Test type or its equivalent. This will identify this type to *QA*Director as the test information type contained in TrackRecord. See "Creating and Modifying Types" on page 82.
4. Open the `trqacenter_item.txt` and `trqadirector_item.txt` files.
5. In TrackRecord, apply any field tags listed in the files that you wish to use in the types.
6. If there are any unused tags that are listed in the *QA*Director text files, delete the attribute and tag lines. Each attribute must be mapped to a tag. If there are any tags that appear in these files that are not used, the integration will not work.
7. Save and close the `trqacenter_item.txt` and `trqadirector_item.txt` files.
8. In TrackRecord's *QA*Center Reported Defect type, ensure that the **QC_Defect_Test** field tag is applied to a single-item combination box and that field is pointed to the *QA*Director Test type. See "Using the Type Editor" on page 85.

   In the default schema shipped with TrackRecord, this field tag is associated with the **Test** single-item combination box. This will create a link between the *QA*Center Reported Defect and the *QA*Director Test types and allow TrackRecord to pass its internal database identifier to *QA*Director. When the integration is finished,

there will be a link to the corresponding *QA*Center Reported Defect in each test submitted from *QA*Director.

9 Test the integration by submitting a defect from *QA*Director to TrackRecord. Refer to the *QA*Director documentation for information on submitting a defect to TrackRecord.

Troubleshooting Integration

For a detailed listing of troubleshooting and information for a customized database, refer to "Troubleshooting" in the TrackRecord online help.

For the most current TrackRecord support information, please visit the Compuware Web site at: http://www.compuware.com. From this location, you will have access to our online Knowledgebase and the most current patches and updates for the product.

Importing and Exporting Data

TrackRecord allows you to transfer data between TrackRecord databases, import data from any delimited ASCII file, or export data to a delimited ASCII file or a Microsoft Access database. For example, you might import an address list to avoid retyping it, export a developer's list of notes or defect reports, or import an appointment book's contents.

Note: If you wish to export data to a Microsoft Access database, you must have MS Access 97 or MS Access 2000 installed.

TrackRecord provides two methods for importing and four methods for exporting data:

- The **Import-Export Wizard**. This wizard steps you through the process of importing, exporting, and creating a template. See "Using the Import-Export Wizard" on page 34.
- The **Import** and **Export** commands on the **Administrator** menu. Note that you must create a template file before using these commands. See "Import and Export Menu Commands" on page 39 for more information regarding the menu options. For more information on templates, see page 40.
- For exporting only, you can use the **Copy** option when available from the **Edit** menu to transfer data to the clipboard. The data can then be pasted into another program. "Transfer Data With Copy and Paste" on page 39 provides additional information.
- For exporting only, you can use the **command line interface** to automatically create an Access table on a periodic basis, for example,

using Microsoft's Task Scheduler. Refer to "Exporting Using the Command Line Interface" on page 39.

Using the Import-Export Wizard

The TrackRecord Import-Export Wizard guides you through the process of importing, exporting, and creating a template.

Importing With the Import-Export Wizard

To import data into a TrackRecord database, you must have previously created a delimited ASCII file. The Import-Export Wizard can create the ASCII file with the Export option on the Administrator's menu, or with any program capable of exporting to a delimited ASCII file, such as Microsoft Excel.

Caution: When importing data, it is advisable to make a copy of your database before attempting to import data.

If you are importing data from an ASCII file that was not created by the Import-Export Wizard, you should verify that the data is in the correct format by exporting data from your existing database and comparing the file created with the file you intend to import. If the formats do not match, you can alter the data to be imported before attempting the import. Importing data that was exported with the Import-Export Wizard is less error-prone.

Required: The format of date fields must correspond to the currently configured window's short date format.

To import using the import-export wizard, perform the following procedure:

1 Click the taskbar's **Start** button and from TrackRecord's program files, choose **Import-Export Wizard**. Be sure to log in to the database into which you want to import data.

2 Select the **Import** option from the Wizard dialog box (Figure 2-2). Click **Next**.

Figure 2-2. Import Export Wizard

3 Enter the name of the file containing your exported data or click the **Browse** button to select an existing name. Click **Next**.

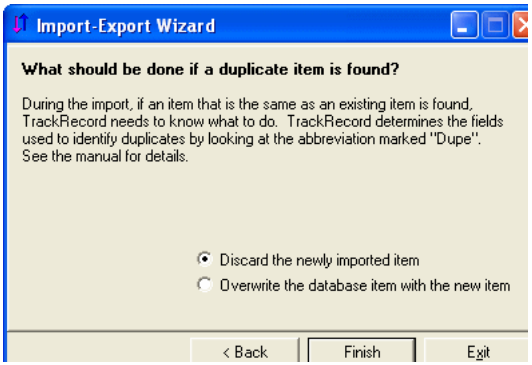
4 You will be prompted to indicate if you have an existing template file.

A template file specifies the type and fields to be imported.

- ◇ If you have already created a template file, click **Yes** and enter its filename or browse to the location. Click **Next** to continue.
- ◇ If you do not yet have a template file, click **No**. You will be prompted through the process of creating one. Select the type and fields to be imported and choose the delimiter. Click **Next** to continue through the Wizard.

5 Indicate what action should be taken if duplicate data is found (Figure 2-3). Click **Finish**.

Figure 2-3. Using the Import-Export Wizard to Import



Duplicate imported data can be discarded or existing data can be overwritten. Duplicates are determined by comparing the field whose abbreviation is Dupe. (This abbreviation was established when the type was created, as described in "Identifying Duplicate Imported Items" on page 78.)

The data is considered a duplicate if it matches existing data of the type being added or of a parent type. Note that the data is not considered a duplicate if it matches a child of the type being added.

A message will display the number of records imported and the Wizard will be closed.

Exporting With the Import-Export Wizard

You can use the Import-Export Wizard to export data from a TrackRecord database into a delimited ASCII file or into a Microsoft Access database. This data can then be imported into another TrackRecord database or into another program, such as Microsoft Excel or Microsoft Word.

To export data from a TrackRecord database:

1 Optionally, log in to TrackRecord, being sure to specify the database from which you intend to export data, and create a query that would find the information you wish to export.

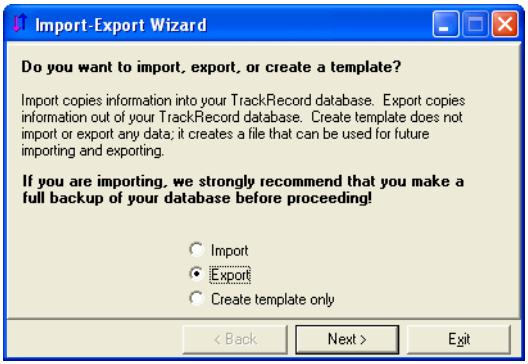
If you will be exporting all items of a particular type, you do not have to create a query and can proceed directly to step 2.

2 Click the taskbar's **Start** button and from TrackRecord's program files, choose **Import-Export Wizard**.

3 Log in to the database from which you want to export data.

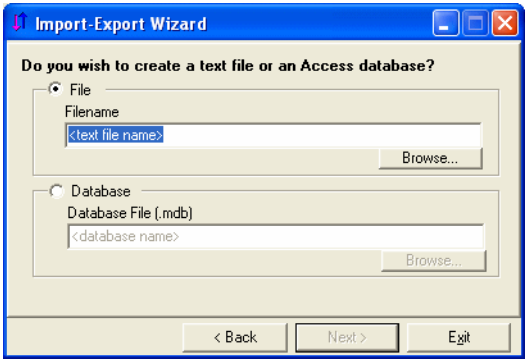
4 When the Import-Export Wizard opens, select the **Export** option (Figure 2-4) and click **Next**.

Figure 2-4. Using the Import-Export Wizard to Export



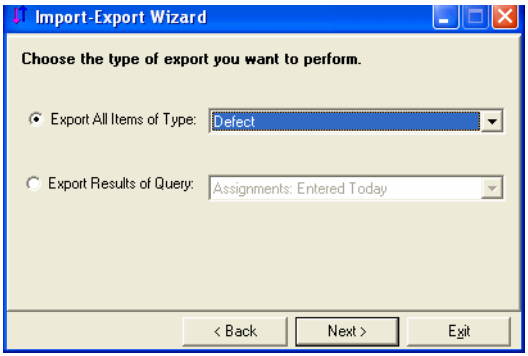
5 Select an option to indicate whether you wish to create either a file or an Access database (Figure 2-5). Enter a name for the file to be created. Click **Next**.

Figure 2-5. Creating a Export File



6 Select whether to export all items of a certain type or the results of a query (Figure 2-6). Use the drop-down lists to select the type or query. Click **Next**.

Figure 2-6. Selecting the Type or Query



7 If you have already created a template, enter its name.

A template file specifies the type and fields to be exported. If you do not yet have a template file:

- ◇ Click the **New Template** button. You will be prompted through the process of creating a template.
- ◇ Select the type and fields to be exported. When selecting fields, note that the **Add** button is disabled for Parent fields. You must add each child field individually.
- ◇ Choose a delimiter character, being careful to choose a character that is not contained in the data. The pipe (|) or tilde (~) characters are often good choices.
- ◇ If you want to save the template file, enter a file name.

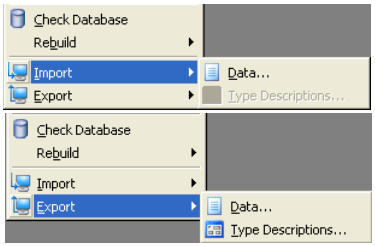
8 A message will notify you if the export was successful. Click **OK**.

You can choose to run the Wizard again or close the Wizard.

If you created a delimited ASCII file, it can be imported into another TrackRecord database or into any program capable of importing delimited ASCII data. If you created an Access database, it can be opened with Access.

Import and Export Menu Commands

The **Administrator's** menu contains **Import** and **Export** commands. These commands allow you to import and export data without using the Import-Export Wizard.



Required: A template file is required for both importing and exporting. You can use the Import-Export Wizard to create the template file by selecting the **Template Only** option. See "Using the Import-Export Wizard" on page 34.

When exporting, you can choose to export data (the entire database or the results of a query) or type descriptions (type descriptions are like the schema of a database).

When importing, you can import data or type descriptions.

Transfer Data With Copy and Paste

If the information displayed in an Outline view, the easiest way to transfer the information is to use Copy and Paste.

Configure the report to display the columns of information you wish to transfer, select the header you want to copy, and use **Edit>Copy** option to copy the header. The displayed data will be transferred to the clipboard and can be pasted into Microsoft Excel, Word, Adobe FrameMaker, or other applications.

Exporting Using the Command Line Interface

You can export TrackRecord data to an Access database through a command line interface rather than through the Import-Export Wizard.

Using the command line interface allows you to automatically create an Access table on a periodic basis, for example, using Microsoft's Task Scheduler.

To export using the command line interface, use the command **TRTOMDB.EXE**. The following parameters are supported:

-d	TrackRecord database to use
-u	User name
-p	Password
-t	Full path to the template file
-a	Full path of the Access table to be created
-q	Optional. Name of the query in this database that is to be used. If not specified, all items defined in the template will be exported
-o	Optional. When set to TRUE, it will suppress the MDB overwrite dialog box. The default is False

All parameters must be enclosed in double quotes. The following is an example of a properly formatted command line:

```
TRTOMDB.EXE -d"Default" -u"Admin" -p"Admin" -t"c:\template.tpl"
-a"c:\access.mdb" -q"Bugs" -o"True"
```

Manually Creating Template Files and Importing

To import from or export to an ASCII file, you must first create a template file, which describes the mappings of the fields in your data files to the equivalent TrackRecord fields. You can use the Import-Export Wizard to create a template, as described in "Using the Import-Export Wizard" on page 34, or you can manually create a template file.

Setting up a Template File and Import Information

Tip: Importing from ASCII files can be quite slow in many situations. The more items there are in your database, the slower importing may be because TrackRecord must check for duplicate data.

1. Determine which TrackRecord types correspond to the type of information you want to import.

   Each ASCII file can only contain items of a single type, but any subfield of a particular item can also be included in the import file.

   For example, you could import Tasks in a single ASCII file, and since Tasks can include an "Assigned to" field, you could also import People to fill that field in the Tasks. The People would be imported as an integral part of each Task in your ASCII file, but would be created as separate objects contained within the Tasks in TrackRecord.

2. Determine the delimiter character used to separate fields in your ASCII file.

Double quote characters can be used in import files at the beginning and end of fields, but another delimiter character (often a comma is used) must also be used to delimit the fields.

3 Edit a file called <*name*>.tpl, where <*name*> is any title you choose to assign to the template.

4 Insert the following text into the file:

```
[Template]
TypeName=<TrackRecord type name>
DelimiterChar=<delimiter char>
```

where <TrackRecord type name> is the name of the TrackRecord type for the information you are importing, and <delimiter char> is the delimiter character. For example, you might enter:

```
[Template]
TypeName=Defect
DelimiterChar=$
```

To use a newline character as the delimiter, specify `DelimiterChar=\n`.

5 List the names of the TrackRecord fields in the type specified that correspond to the types in your input file.

There are two types of field types:

- **Basic Types**— Simple fields for common types, like names, addresses, strings, numerics, etc. There may actually be more than one corresponding "field" in the import or export file
- **Compound Types**— All other types are Compound Types.

In the sample database, there are two fields that contain more than one field:

- **Names**— There will be three fields (Last, First, Middle)
- **Addresses**— There will be six fields (Line 1, Line 2, City, State, Zip, Country)

If a field type is a Compound Type, you must also specify how many subfields of the Compound Type are in the input file, by specifying the number after a comma on the **FieldName** line. For Basic Types consisting of multiple fields, each of these fields is considered one field, and the number of fields specified after the name of the Compound Field should reflect that, as shown in the Reported By field in the following example.

For example, to import a Defect containing a Description, a Project, and a Priority, you would use the following lines (without the comments):

```
FieldName=Description; The description of the defect
FieldName=Project,1; The Project, which has one subfield.
```

```
(thereisnoactualfieldintheinputfilecorrespondingtothisline;it
just affects the interpretation of the next line)
FieldName=Description;TheDescriptionfieldfromtheProject(theone
subfield)
FieldName=Priority;The Priority of the Bug Report
```

To ignore a field that exists in the input file, use a I lhœQdp h@specifier, but don't specify a field name.

To import Checkbox and Radio Button items, your ASCII (import) file should contain either 0 or a blank space for unchecked items, and 1 or an "X" for checked items.

So our sample would look like this:

```
[Template]
TypeName=Defect
DelimiterChar=$
FieldName=Description
FieldName=Project,1
FieldName=Description
FieldName=Priority
FieldName=Fixed
```

A more complex example, which would also export the people in the Reported by and Fixed by fields (and their names, companies, and addresses) would be:

```
[Template]
TypeName=Defect
DelimiterChar=$
FieldName=Description
FieldName=Priority
FieldName=Project,1
FieldName=Description
FieldName=Reported by,3; 3 fields are specified:
FieldName=Name; Name, Address, and Company
FieldName=Company,1; For the Company, only name is used.
FieldName=Name
FieldName=Address; This is the third field in Reported by
FieldName=Fixed by,3
FieldName=Name
FieldName=Company,1
FieldName=Name
FieldName=Address
```

6 Ensure your import file is formatted to correspond to the template file, with each field separated from the next field by the delimiter character (except for fields at the ends of records, which use the new-line character as a delimiter).

The following is a valid input file for the template described earlier:

```
Generic bug 1$TrackRecord$2
Generic bug 2$TrackRecord$1
```

Each record should be on a line by itself, with the newline character representing the end of the record. The only exception is that newline characters are allowed within double-quoted fields.

Fields corresponding to TrackRecord lists must be terminated with an extra delimiter character after the last item in the list.

Tip: *See "Importing File Specifications" on page 43 for information about specifying Name and Address fields.*

Note: The format of date fields must correspond to the currently configured window's short date format.

7 Back up the contents of the TrackRecord object database subdirectory, in case you want to "undo" the import if there is a mistake in the template or data files.

8 To verify that your data is in the correct format, export data from your existing database and compare the file created with the file to be imported.

If the formats do not match, alter the import file.

9 Start TrackRecord and from the **Administrator** menu, choose **Import**.

10 Select the Database, template, and input files. Click **OK**. If there are any errors, messages will appear.

Note: After a large import, clicking on a type in the Item Browser may result in a long pause as TrackRecord updates its indices. The more items you import, the longer this pause will be. This process can take several minutes to complete.

Importing File Specifications

The following Basic Types must have all subfields appear in the following order in an import file, with each subfield separated by the delimiter character:

```
NameLast Name, First Name, Middle Name
AddressAddress Line 1, Address Line 2, City, State, Zip, Country
```

For example, the following template could be used to import information about a person:

```
[Template]
TypeName=Person
DelimiterChar=,
FieldName=Name
FieldName=Address
FieldName=Phone Numbers,1
FieldName=Phone
```

Because Name and Address are Basic Type fields, you should only specify the name of the field (as above), not the name of all the subfields that make up the name or address. If in doubt about field names, look at an

item in the Item view for the particular type that you are importing; the name to use is displayed near the field itself.

For this template file, one valid input file might be:

```
Doe,John,J,123 Any Street,,Anytown,DC,12345,USA,2024567890,,
Doe,Jane,,456 Any Street,,Anytown,DC,12345,USA,2024567890,,
```

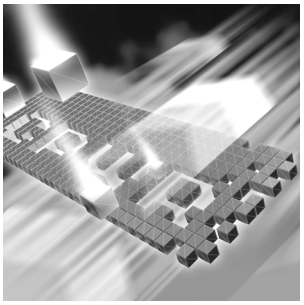
The extra comma after each of the phone numbers indicates the end of the list.

Troubleshooting Input Files

If you are having problems with your input file and you are not sure if it matches your template, try using the same template to export some existing items to an ASCII file. After exporting the items, look at the file TrackRecord created. Your input file should follow much the same format.

Chapter 3

Setting Up Groups, Team Members, and Users



Required: The tasks described in this chapter require administrative privileges to the database. This chapter assumes familiarity with basic TrackRecord concepts. For an overview of TrackRecord and Administrative tasks, refer to Chapter 1, "Getting Started".

Group Administration

Group administration is used to control access to TrackRecord information. Groups are used to assign privileges, especially in the routing of defects. Before entering users into a TrackRecord database, you should define all of the groups present in the organization as well as their privileges. There are essentially three steps in Group administration:

Planning Group Membership and Access Rights

Because access rights (privileges) are determined through membership in a group, it is important to consider the access needs for each member. For example, some software developers might require read and write access to certain database entries, while others need read-only access to those entries. In this instance, you would create two separate groups.

The following groups are shipped in the default database: Project Admin, Development, Documentation, QA, Support, Release Engineering, and Guest.

Caution: If you do not assign a user to any group, they will be granted full read and write privileges across the database.

Considerations When Creating Groups

Things to note about groups:

- Groups are not the same as organizational departments, although the two might often coincide.

  Team Members belonging to several departments can belong to one group if their database access rights will always be equal. If different members of a group will have different privileges, e.g. Team leaders, then separate groups will be required in TrackRecord.

- Users can belong to up to fifteen groups.
- Database access privileges become those of their least restrictive group.
- Read-only access rights derived from membership in one group will be replaced by membership in another group that has read and write access.
- When you add a new group with read and write access to TrackRecord, everyone in that group will have full access to all information types unless you edit the types to reference the new group.
- If you do not assign a user to any group, they will be granted full read and write privileges across the database.

Types of Access Rights

After you have determined your group membership, you must select the group's access rights. Once a group is entered, set the privileges by clicking in the appropriate cell to invoke a drop-down list of choices. These rights include:

- **Read and write** access — Provides full access.
- **Read and add** access — Prevents editing of existing items, but allows the addition of new items. This is often used to allow a user to enter a new defect or person into the database.
- **Read-only** access — Allows viewing of items, but prevents users from making changes.

- **Hide** — Field-level privilege setting that allows making a field invisible on a group-by-group basis.

Implementing Group Access Rights

Now that you have determined the membership and access rights of the groups, you must decide how you are going to structure the access rights.

For example, suppose you wish to create a group with read and write privileges named Documentation Review. There are three levels in which you can assign access rights to the group you are going to create:

- **Global** — Retains default permissions for a group.

  In the example, users assigned to this group will be able to modify and add Defect reports as they review documentation.

- **Type** — Inherits the access control specifications that apply globally, and then lets you restrict access further within each group. Type permissions use the same names as global permissions: read, write, and add.

  If you edit the Defect type to change the access rights to read and add, the users in the Documentation Review group of our example will be allowed to add new Defect items, but are prevented from editing any existing Defect items. See "Type Administration" on page 75 for more information.

- **Field** — Inherits the specifications of global and type restrictions and, optionally, can refine permissions to the field level to add another restriction, Hide. As with type permissions, field permissions can be changed in the Type Editor. You can use the read and write restrictions at the field level, but not add.

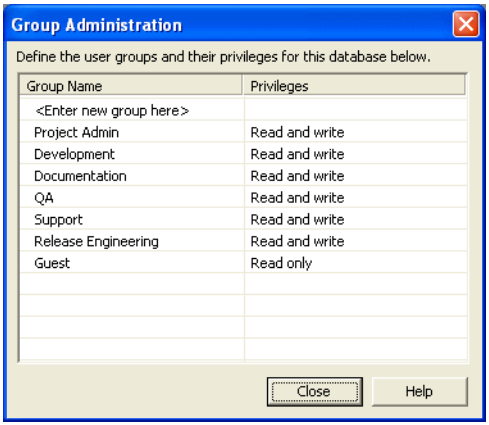
  You can further restrict the access rights of the Documentation Review group by field. For example, you could hide Defect details on the Defect form. For information on setting field permissions, refer to "Type Administration" on page 75.

Creating or Modifying a Group

To create or modify a group, perform the following procedure:

1 From the **Administrator** menu, choose **Group Administration**. The Group Administration dialog box appears.

Figure 3-1. Group Administration



2 Select **<Enter a new group here>** and enter a new group name, or select an existing group.

3 Select an access privilege from the drop-down arrow in the **Privileges** column.

Tip: For more information on access rights, see "Types of Access Rights" on page 48 and "Implementing Group Access Rights" on page 49.

4 Click the **Enter** button.

5 Repeat step 2–4 for each group.

6 Delete any unwanted groups by selecting the name and press the **Delete** key on the keyboard.

Caution: Deleting a group eliminates access to the database for anyone assigned only to that group.

7 Click **Close** to save your changes.

Team Member Administration

For each project listed in the TrackRecord database, a set of Team Members should be assigned to work on the project. A Team Member is an entity that exists in the TrackRecord database, but does not automatically have access to the TrackRecord database. Access is controlled by user names created by the TrackRecord administrator. The administrator can associate a user name (i.e., LoginID) with a Team Member (data item).

Required: In order for a Team Member to be able to log in, a user ID must be created for that Team Member.

Note: The sample database supplies a Team Member type which contains links to projects and groups that you can use to create your own. This association allows some fields to populate automatically with appropriate information for that Team Member. Before creating Team Member items, ensure that this type is suitable to your needs. If not, edit the Team Member type first, as described in "Type Administration" on page 75.

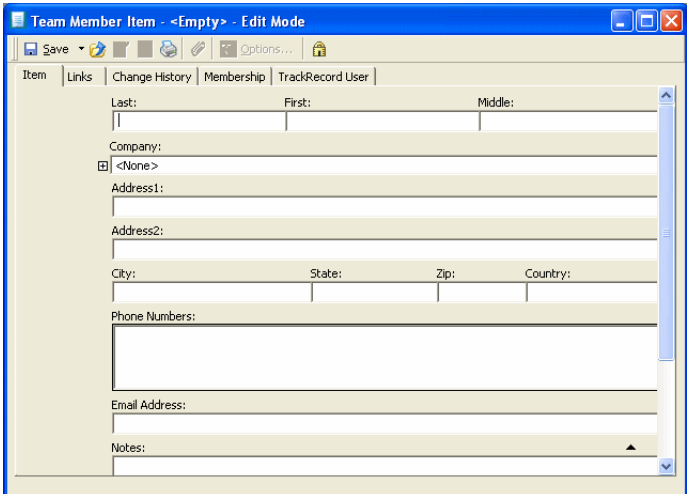
The administrator cannot be associated as a Team Member. The Admin user is a special user identification for TrackRecord and is treated uniquely within TrackRecord. However, you can create Team Members with administrative *authority*.

Non-administrator users can create Team Member items, but they cannot create user names. Therefore, they cannot authorize access to the TrackRecord database. If non-administrator users create Team Member items, an administrator must edit these items to add a login user name to authorize access to the TrackRecord database. This can be useful in certain circumstances: for example, you might want to associate an external vendor to a project by creating a Team Member item so you could then assign defects to them, but they would not have authority to access your database.

Creating Team Members

1. On the toolbar, click the **Item Browser** button.
2. When the Item Browser appears, select **Team Member.**
3. Click the **New** icon in the Item Browser's tool bar. A Team Member item appears, as shown in Figure 3-2.

Figure 3-2. Team Member Item



4 On the **Item** tab, enter information to identify an individual Team Member.

You can create a new company when creating a new Team Member or Person who belongs to the company.

Caution: The Links and Change History tabs are informational and should not be edited. The Links tab will show all items to which this Team Member is linked; the Change History tab will display changes that have been made to this item.

5 On the **Membership** tab, select the Projects and Share Groups with which this Team Member is associated.

If you have not created Projects or Share Groups, you can create them now with the **New** button, or you can add Team Members when you create Projects and Share Groups at a later time. Creating Projects is described in "Project Administration" on page 57.

Tip: Creating Share Groups is described in "Share Group Administration" on page 54.

Some users, such as managers who do not work on a specific project but who occasionally log in to TrackRecord to view information, may not require a project membership.

6 If you are an Administrator or have administrator privileges, you will see the TrackRecord **User** tab which contains a list of all available user IDs. Select the user ID or user IDs you wish to associate with this Team Member, or use the **New** button to create a new user to associate with this Team Member.

Table 3-1. TrackRecord User Types

User's Status	Description
Inactive (Deny Login)	Used only to retain user information in the database while locking out that user name and password for TrackRecord log-ins, also contains email address for receiving emails.
Normal	Active with no administrative authority
Administrator	Active with administrative authority

7 Complete the user information:

- Enter this user's **login name**.
- Select the user Type, as described in Table 3-1, TrackRecord User Types, on page 52
- Optionally, enter and confirm a password. To allow users to log in without a password or to change their password the first time they log in, you may leave the **Password** field blank.

- ◇ Enter an **email address** for this user. An email address is required if the user will be using the AutoAlert utility.
- ◇ Choose the Team Member you wish to associate to this user from the Associated Item dropdown list.

8 Click the **Privileges** tab and select the groups to which this Team Member belongs.

If the Team Member is not a TrackRecord user (step 6) they cannot belong to any group; the check boxes will be disabled.

Each group's database privileges are displayed for your information. Users can belong to up to fifteen groups and their database access privileges become those of their least restrictive group.

9 Click **OK** when finished.

User Administration

In order for a Team Member to be able to login, a user ID must be created for that Team Member. You may also want to determine if there will be additional users who do not have access to the system, but should be created for tracking purposes. This is often used for defects reported from the field.

Naming Conventions for Users

It is important that you use a standard naming convention for all users. You may wish to parallel user ids with their LAN, mainframe, system or sub-system identifications.

Caution: **The Admin user is a special user identification for TrackRecord and is treated uniquely within TrackRecord. Compuware recommends that this user id be used as the only admin-level login into the database and that you do not use this id for anything other than administrative actions, such as check database, edit types, etc. TrackRecord enforces certain restrictions on the Admin user. Admin cannot be associated with a Team Member and cannot be assigned to any group.**

Creating a User

Administrators can add a new user to the database from the User Administration dialog box, or from a Team Member item.

To create a new user or change an existing user's information:

1. From the **Administrator** menu, choose **User Administration**.
2. Click the **New** button or select a user from the displayed list and click **Edit**. This user name will be used for logging in, and to record information about changes made by this user.
3. Enter or change password, activity status, Administrative authority, and other pertinent information.
4. Click the **Privileges** tab to associate this user with a group.
5. Click **OK** when you are finished.

If you wish to only view the user, double-click the user name or select the user and click **View**.

Share Group Administration

Share groups gather people into logical units for sharing information. In share groups, members can share queries, Outline Reports, and templates. Users can belong to multiple share groups.

Use of share groups is optional, but useful. A project team may consist of a few members from different groups that share a common set of Outline Reports, queries, etc. In this instance, you would create a share group containing the project's Team Members, create the reports, and allow the share group access to these reports.

TrackRecord provides three types of sharing:

- **Not shared** (private)
- **Share to All** (administrators only)
- **Shared to** (sharing with all members of a specified share group)

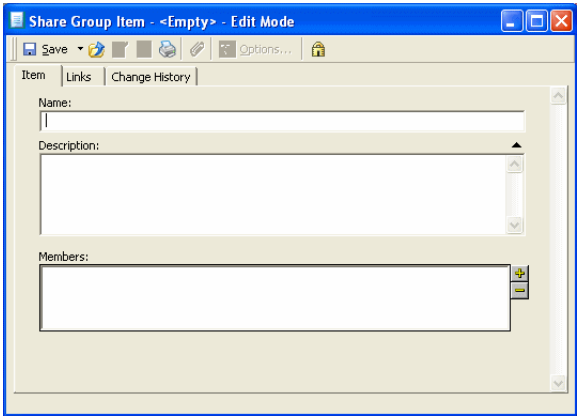
When TrackRecord users share a query, template, or view of database data, they can share it with members of any of the share groups to which they belong. An administrator can share those same items with all groups, change the owner of a report or query, and share anyone's report and queries with everyone.

TrackRecord's sample database contains a share group type. Administrators can use this type to create share groups items to mirror the software development process within their company or department.

Creating a Share Group

1. From the **Tools** menu, choose **Item Browser**.
2. Select **Share Group** and click the **New** button on the tool bar. The Share Group item form appears.

Figure 3-3. Share Group Item Form



3. In the **Name** field, enter a name for the share group.
4. Enter a description of the share group in the **Description** field.
5. Click the + button at the right of the **Members** field to add Team Members to the group. If Team Members have not yet been created, you can add members to Share Groups when you create the member items, as described in the next section.
6. Double-click on the members to be added to this share group.
7. Click **OK** to dismiss the Item Browser dialog box and then click **Save** and then **Close**.

Sharing Information with Share Groups

After you have created Queries, Templates, and Outline Reports, you can determine which users can access them by assigning them a sharing property.

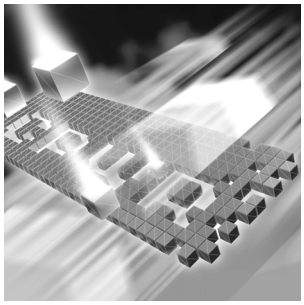
1. Open a query or outline Report.
2. Click **Edit** to lock the item for editing.
3. Click the **Options** button, and then click the **Sharing** button.
4. Select a sharing option.

Note: For information on AutoAlert behavior, refer to "Setting Up Mail Queries" on page 112.

5 Click **OK**.

Chapter 4

Administering TrackRecord



- Project Administration
- Workflow Administration
- Global Preferences Administration
- Type Administration
- Rules Administration

Required: The tasks described in this chapter require administrative privileges to the database. This chapter assumes familiarity with basic TrackRecord concepts. For an overview of TrackRecord and Administrative tasks, refer to Chapter 1, "Getting Started".

Project Administration

Before a team starts using TrackRecord, an administrator must define a project. A project provides an organizational framework for completing a software development deliverable. A project consists mainly of the tasks required to build a software application, the people who work on those tasks, and the task deadlines.

Project Administration Privileges

Project administration differs from other administrative duties in one important respect—project administration requires *no* administrative privileges, but *does* require (when you use the sample database as your starting point) that the designated project administrator belong to the Project Admin group.

Note: The Sample database restricts access to several information types to members of this group.

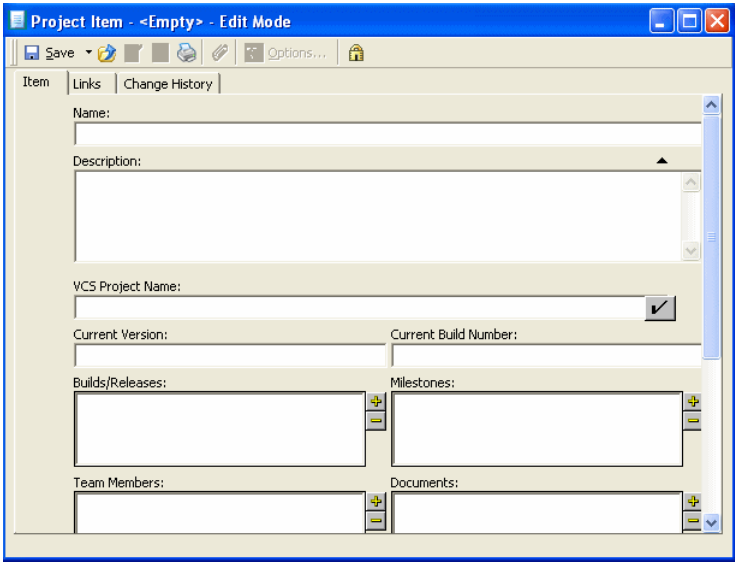
Creating a Project

TrackRecord supplies a Project type in its sample database. You can use this type as shipped, or make modifications to create your own Project type. If modifications are required, make changes before creating your projects. Refer to "Type Administration" on page 75 for information about modifying types.

TrackRecord also supplies a sample project called *YourProject*. You can edit this item to create Project items for your organization or start with a new project. Follow the procedure below to create a new project.

1. From the toolbar, click the **Item Browser** button.
2. When the Item Browser appears, select **Project**.
3. Click the **New** button on the tool bar. A Project Item form appears (Figure 4-1).

Figure 4-1. Project Item



4. Enter a name and description for the project.

Tip: The only required field is the project name.

5. Optionally, enter a **Current Version** and **Current Build Number**.

   This number is used as a starting point to construct build names that also contain an incrementing build number.

6. Optionally, enter Builds/Releases, Milestones, Team Members, Documents, Functional Areas, Keywords and Sub-projects.

   Remember that a form for one type can "contain" forms for other types. In the sample Project form, the fields listed in this step are separate types. Clicking the **Add (+)** button allows you to choose

from the existing items of this type and to create new items of this type.

Note: Note that you can modify the project later to add additional items. If Team Members have not yet been created, you can add them to Projects when the member items are created, as described in "Team Member Administration" on page 50.

7 Click **Save** and then **Close**.

Opening Projects

To open a project, click the **Item Browser** button from the toolbar. Select **Project** from the list and double-click the project you wish to open. You can also open a project from the Membership tab of the Team Member item.

Cloning Projects

When you create projects, TrackRecord lets you duplicate and reuse the milestones from an existing project. This feature allows development teams to create multiple projects that all use the same milestone events. Using identical milestones promotes consistency across projects.

To clone a Project, open the Project to be cloned, then select **Duplicate** from the **Close** drop-down menu in the toolbar.

Multiple Projects

Projects may share people, code, and other resources. By tracking multiple projects in a single database, a Team Member can see all the tasks that require completion on a given day, regardless of the project with which the task is associated. A single database makes it easier to track problems in shared code, as well.

Data from multiple projects can reveal potential problem areas. For example, if testers of Project A are finding many installation bugs, developers of Project B may want to emphasize the test and development effort on their installation module.

Historical Information

Retaining information after finishing a project provides helpful historical data. For example, when building a second version of a product, you could look back at the first version's defects and analyze which parts of the program gave you the most trouble during development. That knowledge could influence how you staff or schedule that part of the second version's development.

Workflow Administration

A defect, task, or other item can pass through a series of states between the time it is entered and the time it is closed. For example, when a defect is entered it is generally in an Unreviewed state. It might then be assigned to someone, resolved, and validated. This series of actions, and the states resulting from these actions, is called a workflow.

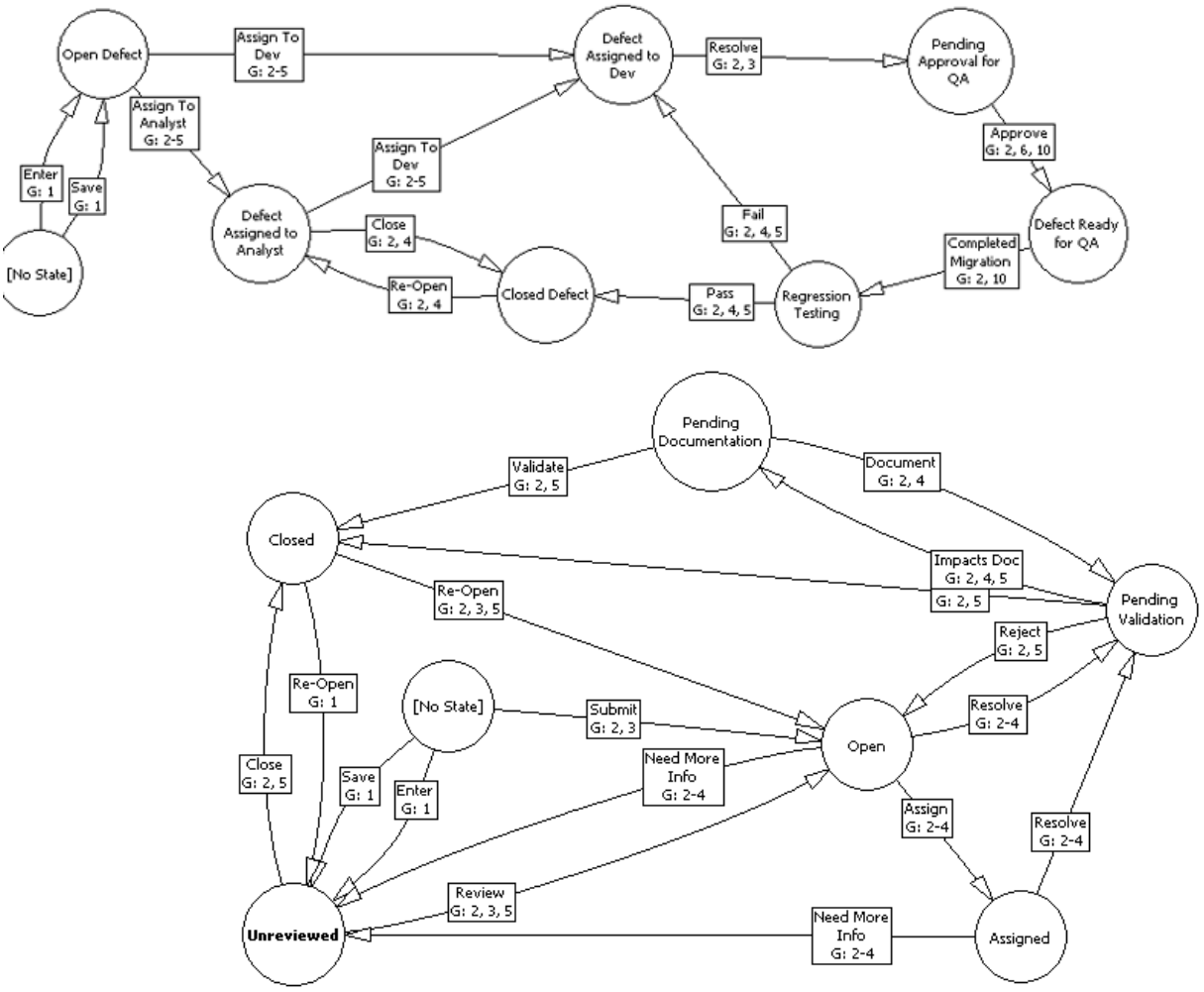
An automated, flexible workflow process helps route software deliverables and defects appropriately through development, testing, documentation, and other groups to maximize team effectiveness. Tracking the state of items allows for more accurate assessment of a project's progress. For example, project managers might determine how to best allocate resources by examining all defects in the Pending Validation state or those in the Unassigned state.

The Workflow Editor provides an intuitive diagram-based interface that defines the workflow to meet the specific team or project requirements. TrackRecord administrators also can define multiple workflow paths for different users and data types. Figure 4-2 illustrates two examples of workflow. The *circles* represent the various states for an item type and the *arrows* signify the actions that can be performed on an item in order to advance the item type from one state to another (depending on security).

Rectangles describe the action and groups of users who can perform the action.

Note: This function is not available in TrackRecord during a CARS engagement.

Figure 4-2. Two Examples of Workflow



Planning a Workflow

Before implementing a workflow, you must think about your defect or task resolution process— what states items can be in, who has authority to change each state, and so on. This step is extremely important and will save you hours of time when you start customizing TrackRecord.

Tips When Planning a Workflow

- Once you have determined your process, diagram the workflow similar to the examples shown in Figure 4-2.
- A good rule of thumb is that each state (phase) should be "owned" by a specific group. For example, Quality Assurance is the only group authorized to move defects to a *Fixed* state.
- A state is meant to communicate meaningful information. It is a notification to someone that they need to perform an action.
- You use TrackRecord groups to define who can move an item from one state to the next. You can also create different workflow paths and privileges for specific groups.
- Be sure to specify which groups can perform each action.
- Create a naming convention for each change in status. For example, you could use Reject, Open, Waiting for Information, and Closed.
- The simpler the process, the more effective it will be and the more likely that it will be followed.
- When an item's status is changed, a TrackRecord action has occurred. Therefore, the presence of a status field in a type causes an Action button and Action list to be displayed on the type form. Users will use the Action button to indicate the actions they perform on the item. The actions listed in the Action list, and the state to which the item is set after each action, is determined by the workflow transitions you define.
- You can use the TrackRecord sample database to implement workflow for your project data. The sample database building blocks for workflow include the types:
  - **Action** – Enter, Open, Validate, Close, and others
  - **Status** – Unreviewed, Open, Closed, and others

Implementing a Workflow

Once you have determined your task resolution process, implementing a workflow is a two-step process:

- Creating the workflow, as described in "Displaying the Workflow Editor" on page 63.
- Including a Status field in a type, as described in "Applying Workflow to Types" on page 68.

Optionally, you may also want to design a workflow to use separate rules for different types (refer to "Designing a Workflow Based on Type" on page 67 for more information) or customize your workflow by implementing rules that disable the standard workflow in certain situations (refer to "Rules Administration" on page 95 and to the online help for information on creating rules).

The following procedures describe setting up workflow using the building blocks from the sample database.

Displaying the Workflow Editor

A workflow is created and modified with the Workflow Editor utility.

Note: The sample TrackRecord database includes a Status field on the Defect type. To apply workflow rules to other types, such as Tasks, you must add a Status field to their type definitions.

The Workflow Editor assumes the existence of certain types and tags in your database. If your database does not contain these types and tags (for example, if you are using an older database or if you have modified the types in the sample database), the warnings described in step 2 below are displayed to allow the Workflow Editor to make minor adjustments to your database.

Caution: Before using the Workflow Editor, ensure that no one is accessing the database.

To open the Workflow Editor, follow these steps:

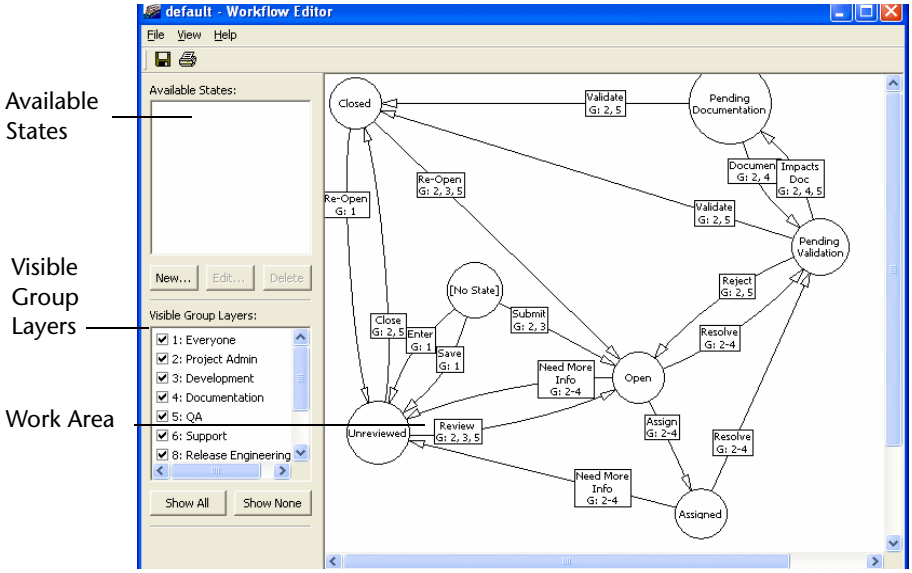
1. From the **Programs** menu on the **Start** button, choose **Compuware>TrackRecord>Workflow Editor**.
2. Two warnings may be displayed advising you to back up the database and to request users to log off.

   If these warnings are displayed, close TrackRecord. The warning dialogs and, subsequently, the Workflow Editor will remain active.

   Refer to "Active Users in the Database" on page 79 for information about the Active User dialog box. When all users have logged off, the dialog box is dismissed, and the Workflow Editor makes database modifications and unlocks the database. The Workflow Editor appears and displays the current workflow.

Note: If a user tries to execute a manual task or it is time for an automated task to run, the task will come back "Failed" until the administrator is finished making changes and closes the Workflow Editor.

Figure 4-3. Workflow Editor and the default workflow



In addition to the standard menu bar and toolbar, the Workflow Editor consists of three frames:

- **Available States:** Lists states that have been defined but are not yet used in the workflow. Using the buttons, you can create a new state, edit a state definition, or delete a state, as described in following sections.
- **Visible Group Layers:** Lists all groups that have been defined. Checkmarks indicate the groups whose workflow is visible. The buttons allow you to quickly display the workflow for all or no groups. By default, the workflow for all groups is displayed. To view the workflow of a subset of groups, uncheck the groups that are not to be displayed or use the **Show None** button to remove all groups, then check the group(s) to be displayed. The workflow for Everyone is always visible (as a watermark if not selected) to prevent the addition of unnecessary transitions.
- **Work Area:** The work area is where you create the workflow. **Circles represent states; rectangles represent actions; arrows indicate the state transition caused by that action**. For example, if a defect is in the Unreviewed state and the action Review is performed, the defect state will change to Open.

The action text boxes also indicate the groups that are permitted to perform that action. The numbers correspond to the numbers in the Visible Group Layers frame.

Moving the cursor over a state or action will display a tool tip describing the state or action.

Working with States

To change a state, select it by clicking on the state circle in the work area. You can then do one of the following:

- **Move the state**— Moving a state may increase the readability of your workflow. All actions remain attached. To move a state circle, click in the circle to select it, drag the circle to the desired location and release.
- **Remove/delete the state**— Right click on the state circle and select a menu option to either remove the state from the workflow or to delete the state.

  If the state is removed, it will be listed in the Available States frame and can be reinserted into the workflow. If it is deleted, the state will no longer be available. (You can recreate the state if necessary using the **New** button in the Available States frame.) You will be prompted to confirm the delete operation. All transitions leading into or out of this state will also be deleted.
- **Create a new transition**— Allows an item to leave this state, as described in "Creating a New Transition" on page 67.

Creating a New State

Tip: Use the diagram you designed when planning your workflow. For information on planning your workflow, see "Planning a Workflow" on page 61.

1. Access the Workflow Editor as described on page 63.
2. Click on the **New** button in the Available States frame. The State Properties dialog box appears (Figure 4-4).

Figure 4-4. State Properties Dialog Box

3. Enter the name of the new state. All states in the database must have a unique name.
4. Enter a description for this state.

5 If all Defect items should be placed in this state by default, select the **Initial Defect State** option.

Note that the default state for other types, such as Tasks, is set through the use of templates, as described in "Default Status for New Items" on page 71.

6 Click **Save and New** to continue creating new states or click **Save** if you have completed creating new states. The state is listed in the Available States frame.

7 To use the new state, click on the name of the state in the Available States frame and drag it onto the work area. Once the new state appears, it is no longer listed in the Available States frame to avoid duplicate states.

8 Create transitions that cause the item to leave this state, as described in "Creating a New Transition" on page 67. Click in other states and create transitions that cause items to enter this state.

Working with Transitions

Each state can have multiple transitions. For example, you might want to have two possible actions for an item in the Open state: Assign and Reject. Further, you might want the Reject action to result in a different state depending on who entered that action. You would create three transitions:

- **Assign** transition leading to the **Assigned** state.
- **Reject** transition leading to the **Closed** state if a member of the Project Admin group enters it.
- **Reject** transition leading to the **Unreviewed** state if a member of any other group, which would allow the defect to be reviewed again, enters it.

To change a transition, click on any part of the transition line. Handles (small black squares) are displayed on both ends of the transition line. Click on a handle and drag it to a new state. You can change the state in which the transition is generated or the state in which the transition results.

To change the properties of a transition, double-click on any part of the transition line. A Transition Properties dialog box appears. You can change the name of the action or the groups that can perform the action.

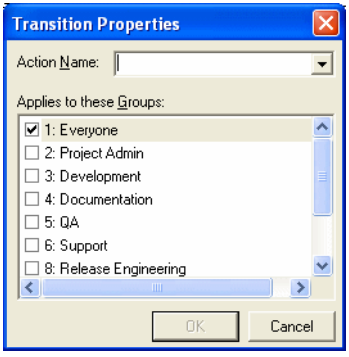
To delete a transition, right-click on the action and select **Delete** from the menu. The transition is deleted from the work area. The action associated with the transition, however, is not actually deleted. Once an action

name has been created, it is never removed from the drop-down list in the Transition Properties dialog box.

Creating a New Transition

1. Click in a state circle to select it. When you move the cursor, an arrow is displayed.
2. Click on the handle (the small black square) at the end of the arrow, drag it to another state, and release. You must release in another state circle or the transition will not be created. The Transition Properties dialog box appears, as shown in Figure 4-5.

Figure 4-5. Transition Properties Dialog Box



3. Select an action from the drop-down list or enter the name of a new action.

Note: Do not create two identical actions from the same state for a given group that result in different states.

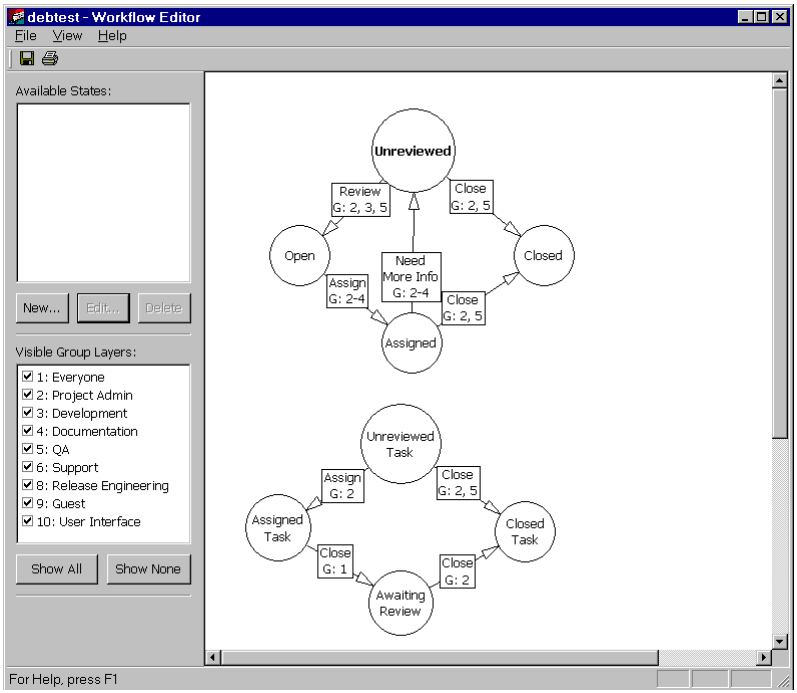
4. Select the groups that will be allowed to perform this action from this state. Users belonging to more than one group will be able to perform actions permitted for all groups to which they belong.
5. Click **OK**. A transition line is created in the work area. The path of the transition line, the location of the text box, and the point at which the line intersects the state circle are determined automatically and cannot be altered.

Designing a Workflow Based on Type

Each TrackRecord database can have only one workflow, but you can design that workflow to allow different behavior based on the item type. To accomplish this, define a separate set of states and transitions for each

type. For example, Figure 4-6 shows a separate set of states and transitions for defects and tasks.

Figure 4-6. Separate Workflow Based on Type



Tip: Each set of states should use unique state names.

Each type's initial state determines which set of workflow rules (states and transitions) the type will follow. The initial state can be set from within the Workflow Editor only for the Defect type. For all other types, you can set an initial state by creating a template for the type and automatically filling the Status field, as described in "Default Status for New Items" on page 71.

Applying Workflow to Types

The TrackRecord sample database includes a status field on the Defect type. To apply workflow rules to other types, such as tasks, you must add a status field to their type definitions.

Tip: Refer to "Creating and Modifying Types" on page 82 for a general discussion of adding fields to types.

The following steps are specific to adding status fields.

1. Log on to TrackRecord as an administrator.
2. From the **Administrator** menu, choose **Edit Types** to open the Type Editor.

   If users are logged on using this database, a dialog box will be displayed listing current users. Send a message requesting users to log off. (Refer to "Active Users in the Database" on page 79 if you would

like a description of this dialog box.) When no users are logged on using this database, the Choose a Type dialog box appears.

3 Open the Defect type.

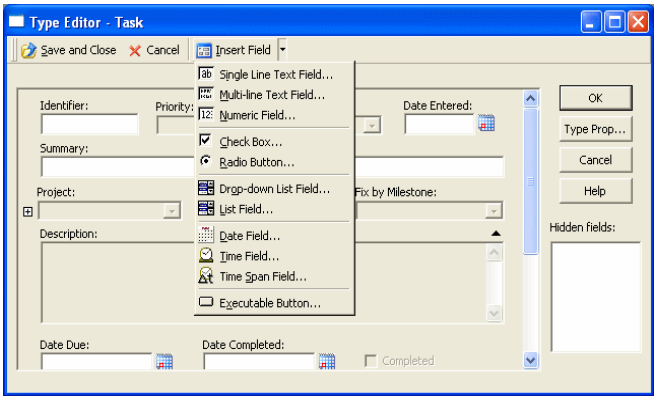
4 Locate the Status field and remove the TR_Item_State tag from the General tab.

5 Save and Close.

6 Double-click the new type to which you want to add a Status field.

7 Click the **Insert Field** button (Figure 4-7) and choose **Drop-down List Field**.

Figure 4-7. Adding a Status Field

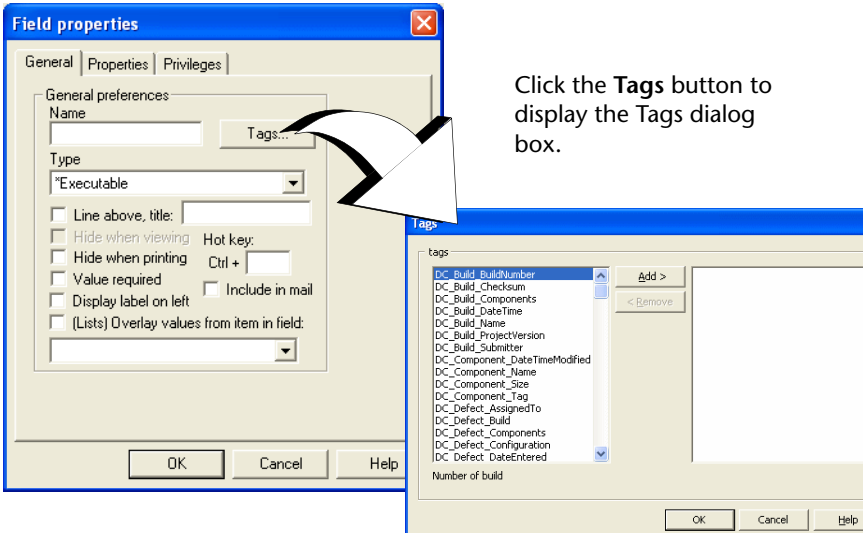


8 Move the pointer around the form. The pointer indicates where the field will be added. Click the mouse to drop the new field onto the Type Editor layout region.

If the type inherits from a parent type and you place the new field in the inherited section, the new field will be added to all types inheriting from that parent.

On the Field Properties dialog box (Figure 4-8), enter a name for the field and click the **Tags** button. The name will be the label for the field. A logical choice might be Status or State.

Figure 4-8. Field Properties Dialog Box



9 Select **TR_Item_State**, and click the **Add** button and **OK**.

10 From the **Type** drop-down list, select **Status** and check any additional options for this field. Click **OK**.

The field will be displayed on the type form. Other fields will be resized or repositioned to accommodate the new field.

To change the field's properties, right-click on the new field and select **Open**. To change the location of this or any other field, click on the field and drag it to another location.

11 Click the **Type Prop...** button from the Type Editor and add the TR_Defect tag.

12 Click **OK** or **Save and Close** to close the Type Editor.

13 Click **Close** to close the Choose a Type dialog box.

14 Create a template to set a default status, as described in "Default Status for New Items" on page 71.

A type should have only one Status field. If a type contains more than one status field, the first one encountered in the form will be active. The second will be ignored.

Default Status for New Items

For all types that include a status field, you should specify a default state to which new items will be set. If you do not set a default state, users will be able to select any available status when they first create an item of this type, and the Action button will not be displayed until a status is set and the item reopened.

For Defect types, you can set the default state through the Workflow Editor, as described in "Creating a New State" on page 65.

For other types, you must set the default state through a template, as shown in the following steps:

1. From the **File** menu, choose **Manage** and then Templates to open the Manage Templates dialog.
2. Open the Defect template and click **Options**.
3. Uncheck the "Apply this template to all new items" box.
4. Save and close.
5. Create a new template for the new type.
6. Set the initial state by choosing from the new Status field.
7. Click **Options** and check the "Apply this template to all new items" box.
8. Save and close.

Testing Workflow Rules

In order for the automated workflow process to route software deliverables and defects appropriately, the workflow should be thoroughly tested to ensure that it is working as expected. It would be good practice to create a series of defects for testing and a list of the various iterations to be tested. These test cases should address the user's group, status of the defect, action desired, and the resulting state. For instance, verify that users do not have the ability to record inappropriate actions.

Global Preferences Administration

The tasks of global preferences administration include:

- Creating queries and Outline Reports for all TrackRecord users
- Identifying favorite types, reports, and a Home Page for all users
- Creating global templates

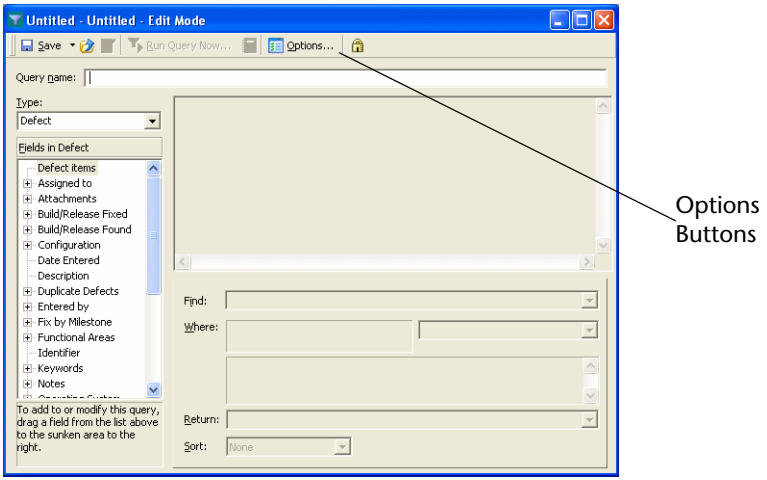
To make the members of a project team productive as quickly as possible, administrators should provide a representative set of queries and Outline Reports to TrackRecord users. These data-extraction tools will allow users to accomplish tasks without spending time learning the intricacies of TrackRecord. Only administrators can create queries and Outline Reports that are globally shared with every TrackRecord user.

The sample database shipped with TrackRecord provides a limited set of global queries and Outline Reports that administrators can study to learn how these constructs behave.

Creating a New Global Query

1 From the **File** menu, choose **New>Query**. Alternatively, if the query already exists, choose **Manage**. The New Query dialog box (Figure 4-9) appears. For existing queries, the Manage Query dialog box appears.

Figure 4-9. New Query Dialog Box



2 Create or modify the query.

3 Click the **Options** toolbar button.

4 On the Options dialog box, click the **Sharing** button.

5 Select the **Share to All** option. Alternatively, you could select share groups to share this query, as described in "Share Group Administration" on page 54.

6 Click **OK** to close the dialog boxes.

7 Click **Save** and then **Close** on the toolbar to close the query.

Creating a New Global Outline Report

1 From the **File** menu, choose **New>Outline Report**. Alternatively, if the Outline Report already exists, use the **Manage** option to select the Outline Report. An Outline Report dialog box appears.

2 Create or modify the Outline Report as described in "Creating Global Queries and Outline Reports" on page 72.

3 Click the **Options** button to open the Outline Options dialog box.

4 Click the **Sharing** button.

5 Select **Share to All** or select share groups.

6 Click **OK** to close the Sharing dialog box. Click **OK** again to close the Options dialog box.

7 Click the **Save** icon when finished.

Creating Global Favorite Types, Reports, and Home Pages

TrackRecord supports many types of items and many reports, but most users will generally use only a few of these types and reports. It is helpful to make it easy for the user to find the types and reports in which they are most interested rather than forcing the user to select from a long list of all available types.

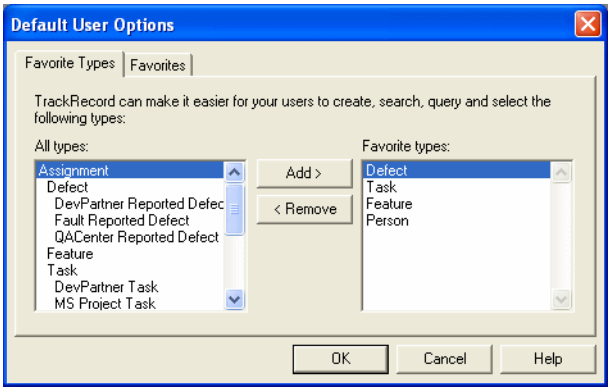
To assist the user, you can establish global favorite types and reports or views. These types and reports will be displayed for all TrackRecord users.

Of the favorite reports and views, you can select one to be the default Home Page for all users. The Home Page is the report or view that will be displayed by default when the user logs in to TrackRecord. The user can always return to this report or view easily by clicking the **Home Page** icon on the tool bar.

Individual users can configure their own favorite types, favorite reports and Home Page, which will override the global favorites. To create global favorite types, reports, and a Home Page, perform the following steps:

1 From the **Administrator** menu, choose **Default User Options** to open the Default User Options dialog box.

Figure 4-10. Default User Options Dialog Box



2 On the **Favorite Types** tab, select a type from the **All Types** list and click the **Add** button to place it in the **Favorite Types** list. This action adds the type to the Favorite Types list for every user of the current database.

3 Use the **Remove** button to delete a global favorite type.

4 Click the **Favorites** tab.

5 Click the **Add** button to add a new global favorite report. Choose the type of report— Outline, Milestone or Graph. The existing reports of that type are listed.

6 Click a report to be listed on the **Favorites** menu for all users and click the **Open** button. The report will be added to the list of favorites.

7 Select one of the favorite reports to be used as a default Home Page for all users from the **Set as Default Home** drop-down list.

8 Click **OK** when finished.

Creating Global Templates

TrackRecord uses templates to supply default field information for new items. Templates are appropriate for any types that have new items created on a regular basis. Global templates supply default values for every user so that data can be entered more quickly and without errors. Users can create their own templates and make them private or share them with members of the groups to which they belong.

Tip: See the TrackRecord User's Guide for more information on templates.

To create a global template, perform the following procedure:

1. From the **File** menu, choose **Manage>Templates.** The Manage Templates dialog box appears.
2. If a template already exists, select it. If not, click the **New** button and select the type.
3. In the **Summary** field, type a descriptive name for this template. This name will identify this template in the Manage Templates dialog box. It will not be displayed when the template is used to enter items.
4. Enter the information to be displayed to all users.
5. Click the **Options** button to select the following:
   - ◇ Select the **Apply This Template to All New Items** check box.
   - ◇ Click the **Sharing** button and select the **Share to All** option.
   - ◇ Click **OK**.
6. When all template items have been set, click **Save and Close**.

Type Administration

TrackRecord allows you to use the default types provided with TrackRecord, modify these default types, and create new types. Before modifying or creating new types, be sure to assess the impact these types will have on your development process. Once types have been used by TrackRecord users, the changes you can make to them will be restricted, as described in "Restrictions to Modifying Types" on page 94.

Note: This function is not available in TrackRecord during a CARS engagement.

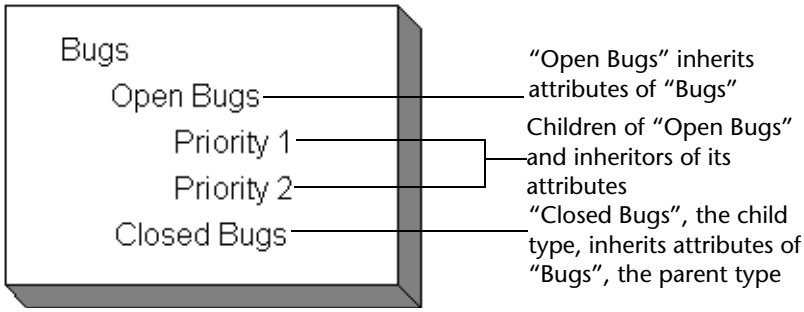
Understanding Data Type Inheritance

TrackRecord information types, such as the Person type, Defect type, and so on, provide the framework on which the TrackRecord database is built and is organized into a data *hierarchy*.

When one type inherits from another, the inheriting type (the *child*) receives all the attributes of the original type (the *parent*). In the example (Figure 4-11), Bugs is the parent of Open Bugs, and Open Bugs is the

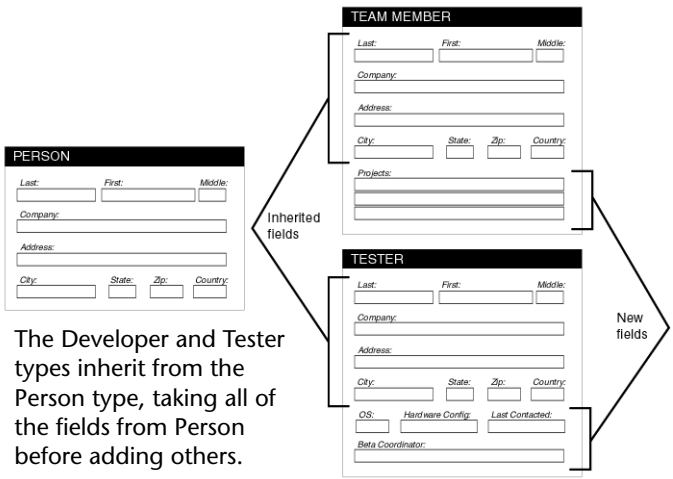
parent for Priority 1 and Priority 2; Open Bugs and Closed Bugs are the child of Bugs, and Priority 1 and Priority 2 are the child of Open Bugs.

Figure 4-11. Data Type Inheritance



After the child inherits fields from the parent, it can add fields of its own. Figure 4-12 illustrates how the Person data type is the parent to the Developer and Beta Tester data types; each child type starts with the fields it inherits from the Person type, and adds its own fields below the inherited fields.

Figure 4-12. Child Items Inherit Fields from Parent



This type of inheritance allows items of any *child* type to be used anywhere in TrackRecord where an item of its *parent* type is expected. For example, Figure 4-12 illustrates that the Team Member type inherits from the Person type, so all Team Member items are also Person items. Therefore, Team Members and Testers can be used to fill fields where Person items are expected. Note, however, that the reverse is *not* generally true: Person items that are not Team Members cannot be used in fields where Team Members are required.

One of the benefits that type inheritance provides is the ability to add another identity to an item of one type. For example, a person in a database changes roles on a project and must be changed to a Team Member. If Team Member is a child type of the Person type, an administrator does not need to remove and reenter information about that individual.

Type inheritance also permits querying on a parent type to retrieve information about all items of that type, including its child types. For example, a query that looks for Defects will retrieve items of the parent as well as the child.

Categories of Types

TrackRecord information types define the structure of the individual pieces of information stored in a project database. Information types, therefore, constitute the heart of TrackRecord. Manipulating types provides control over the structure of the database.

Although all types are equal from a technical point of view, administrators should think of types as falling into two categories:

*Tip: Administrators can determine whether a type is common or restricted with the Type Editor property **Display to All Users,** as described in "Creating and Modifying Types" on page 82. Common types will be displayed to all users; restricted types will not.*

- **Common**— Privileges set so that most users can create and modify items of these types.
- **Restricted**— Privileges set so that only administrators can manipulate them. There are two restricted types:
  - **Administrative—** Items that are behind the scenes, such as the Priority and Resolution types, for creating items that are building blocks of other structures.
  - **Contained**— Created from within another item and have no use outside the context of the containing item. For instance, the Phone Number field within a Team Member item actually holds a Phone Number item. Phone Number items, however, have no meaning as stand-alone items.

Note: Administrators can use the **Show All** check box on the Item Browser to view both common and restricted types. Clearing the **Show All** check box changes the Item Browser display to common types only. ****This function is not available in TrackRecord during a CARS engagement.****

Altering Access Rights for Types and Fields

Access rights can be set for either an entire type or the individual fields within a type.

Type permissions use the same names as global permissions: read, write, and add. Refer to "Planning Group Membership and Access Rights" on page 47 for more information on access rights.

Abbreviations

When an item is displayed in a report, TrackRecord must display at least one field from the item. The field that is always displayed is called the Default Abbreviation. Each type must identify one Default Abbreviation. In addition, each type can have one Short and one Long Default Abbreviation.

Caution: Use caution when defining abbreviations as it is possible to create self-referencing abbreviations that may result in an unstable database.

TrackRecord creates a Default Abbreviation based on the first field in a type. You can change this default, and you can specify a Short and Long abbreviation, with the Type Properties dialog box, as described in "Creating and Modifying Types" on page 82.

Identifying Duplicate Imported Items

When items are imported into the TrackRecord database, TrackRecord must determine if the items are duplicates of existing items. Each type must contain one field or combination of fields to be compared with imported items to determine duplicates. This field or fields is identified with the **Dupe** option on the Type Properties dialog box, as described in "Creating and Modifying Types" on page 82. If you do not specify a Dupe abbreviation, the Default Abbreviation will be used.

ActiveLink Tags

Use ActiveLink tags only when you create new types, or delete fields from an existing type. For more information on tags, consult the online help, Appendix B, "Using Tags", or "Creating and Modifying Types" on page 82.

TrackRecord administrators can name types, and label the fields on types, to match their company's development process. For example, if the organization calls software defects "bugs," they can name a defect type "Bug."

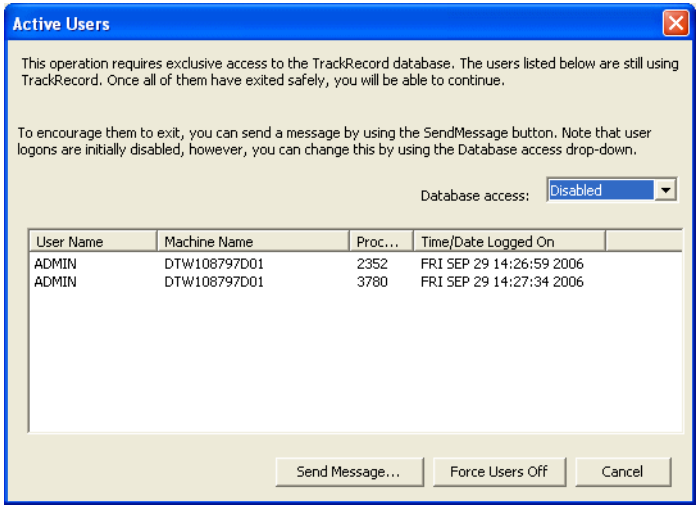
Caution: The Workflow Editor depends on certain ActiveLink tags. If these tags have been removed or altered, the Milestone Status and Workflow Editor option might not be available.

Products that integrate with TrackRecord depend on certain types, and fields within types, to properly integrate its developer tools with the TrackRecord database to gather information automatically. To provide for this dependency, TrackRecord supplies ActiveLink *tags* that let administrators link the types and fields they create with the underlying structure that ActiveLink expects. If you use the types shipped with the sample TrackRecord database, you can change type and label names without concern for ActiveLink tags.

Active Users in the Database

Certain changes to the database, such as creating or modifying types, cannot be made while users are logged on to the database. If you attempt these changes while the database is in use, a dialog box will be displayed listing all the active sessions and current users. Figure 4-13 shows an example of the Active Users dialog box. This dialog box provides an opportunity for the administrator to advise them to log off.

Figure 4-13. Active Users Dialog Box.



In Figure 4-13, note that the user name Admin appears twice because there are two sessions of TrackRecord open with this user name. For more information on redundant users, see "Duplicate Users in the Database" on page 80.

To send a message requesting that users log off, click the **Send Message** button, edit the default message if desired and click the **Send** button. The message is displayed in the TrackRecord client of all logged on users, including those accessing the database through the WebServer. The user list is refreshed every few seconds. Once all users have logged off, this dialog box is dismissed and you can make your desired changes.

Tip: The timing of the message display depends on each user's polling frequency setting.

By default, new logins are disabled when this dialog box appears. You can use the drop-down menu to enable logins; clicking **Cancel** will also re-enable logins.

Note: If a user attempts to log in while logins are disabled, a message box will inform them that logins are disabled. They can choose to log in using a different database. If a user tries to execute a manual task or it is time for an automated task to run, the task will come back "Failed" until the administrator is finished making changes and closes the Type Editor.

Duplicate Users in the Database

Multiple instances of users and machine names occur in the Active Users dialog box when there are two or more sessions of TrackRecord open with this user name. This can occur in the following instances:

- If a user attempts to log into TrackRecord multiple times unsuccessfully, the error message generated each time is interpreted by TrackRecord as an active user. For instance, if five error messages are open, the user and machine name will be listed five times as an active user. In order to remove the redundant users, simply close the login dialogs.
- Any active integration with TrackRecord will add an *automation user* to the Active Users dialog box. This automated user is invoked when using integrated products such as *QA*Director and DevPartner Studio's CodeReview. This will occur even if TrackRecord is not open on the machine running those tools.
- Multiple instances of a user will be generated in the Active User dialog box if the user is logged in more than once through multiple sessions of the Windows Client or the WebServer.

Note: The total number of active users listed represents the currently logged-in users, but does not necessarily reflect the total number of licenses being used.

Planning the Composition and Layout of Forms

To assist teams in their understanding of information types, TrackRecord ships with a sample database populated with information types that were

created to set up one kind of development process. By studying these types, administrators can develop an understanding of how types integrate to implement a process. This is an important step because once items have been added, it is more difficult to customize them.

Tips For Planning Forms

- Use the Type Editor to open a type and view its fields, including its properties and access rights. For information on using the Type Editor, see "Using the Type Editor" on page 85 for more information.

Note: View possible values for each field in the Item Browser. In order to view a complete list of data types, select the **Show All** check box. ****This function is not available in TrackRecord during a CARS engagement.****

- Identify the fields in the sample database you wish to keep, rename, add, delete, and move.
- Specify which groups should be able to view each field. TrackRecord allows you to hide certain fields from users.
- Identify all possible values of each field you will be using.
- Note the fields in the sample database which should have properties changed. This depends upon the possible values for a field. Some considerations include:
  - Is the field required?
  - Should entries to a field be free-form, selected from a drop-down list, or true/false? For example, if the number of values is unlimited, the field should be a free-form entry.
  - If selected from a list, are there multiple entries or a single entry?
- If you want to prevent items from being accidentally deleted by users, create a hidden field within the item type and make the field Read-only for all the groups. This will prevent users from being able to delete any items of that type.

Reserved Type Names

There are eighteen basic types whose names are reserved by the database. The following reserved names are not available when creating or renaming a type:

- Name
- String
- clump
- Time/Date

- Address
- File Name
- Numeric
- List
- Boolean
- Time Span
- Notepad
- Attachment
- Phone
- Check Box
- Radio Button
- Date
- Executable
- Query

Creating and Modifying Types

While companies can use the sample database shipped with TrackRecord as a production database, many organizations will see the need to add to or modify the information types supplied. The procedures that follow explain type creation and modification. To create or modify a type:

1 Log on to TrackRecord as an administrator and close all items and views (defects, reports, etc.).

Refer to "Starting TrackRecord as an Administrator" on page 18.

2 From the **Administrator** menu, choose **Edit Types**.

A warning is displayed advising you to back up your database before modifying types. Refer to "Scheduling Backups" on page 30 if you would like to back up your database at this time.

Note: If users or other TrackRecord clients, such as the WebServer, are logged on using this database, a dialog box will be displayed listing current users. Refer to "Active Users in the Database" on page 79 for information about this dialog box.

3 If you are creating a new type, on the Choose a Type dialog box click the **New** button and proceed to step 5.

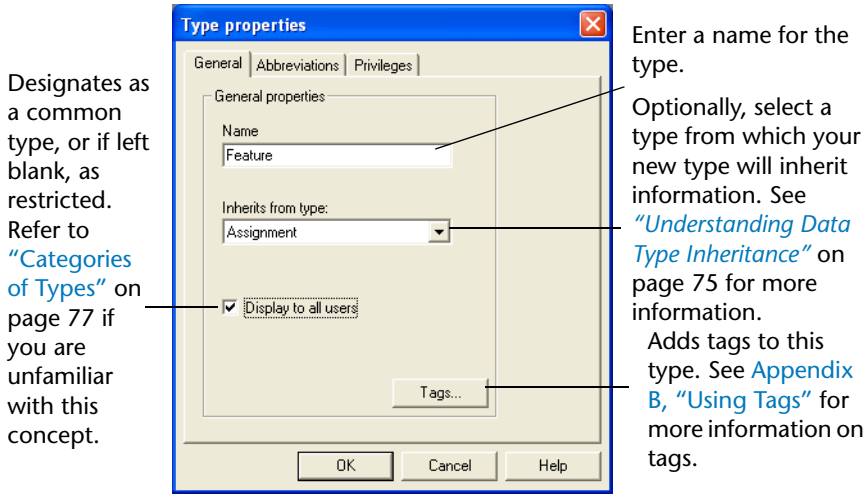
If you are modifying a type, click on the Type and click the **Edit** button.

4 To modify type properties, click the **Type Properties** button.

If changing field properties only, skip to "Using the Type Editor" on page 85.

5 On the **Type Properties General** tab (Figure 4-14), identify the name and properties of the type. Before naming the type, consult "Reserved Type Names" on page 81.

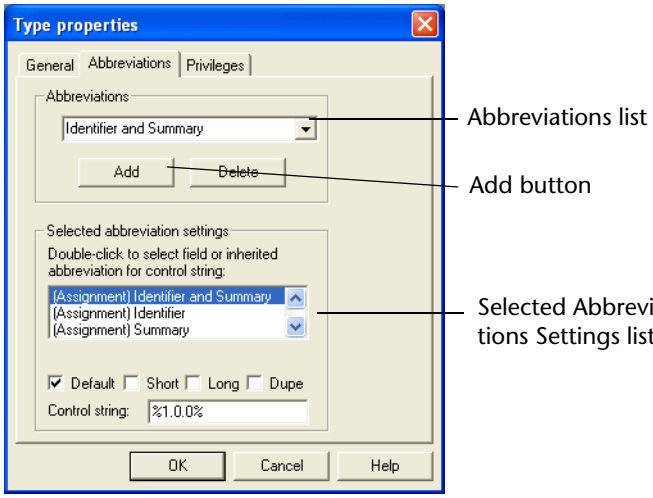
Figure 4-14. Type Properties General tab



6 Click the **Abbreviations** tab (Figure 4-15) to identify the field(s) to be used as Default Abbreviation(s).

Note: If you are editing a subtype (or child type) and want to display it to all users, you must also check the "Display to all users" checkbox of the parent type as well as the subtype.

Figure 4-15. Type Properties Abbreviations Tab



If you are creating a type that does not inherit from another type, the list of fields will be empty until you create fields, as described in later

steps. You can return to this tab after you add fields to the form. If this type inherits from another type, the inherited fields are displayed.

- To change the Default abbreviation, select **Default** in the **Abbreviations** list. In the **Selected Abbreviation Settings** list, select a field (or combination of fields), and check the **Default** option.
- To set the Short and Long abbreviations, click the **Add** button and add a name for this abbreviation, then select the field or combination of fields from the **Selected Abbreviation Settings** list and check the **Short** or **Long** option.
- To specify a field to be used to identify duplicate items when importing data, select the field or combination of fields and check the **Dupe** option. If you do not specify a Dupe abbreviation, the Default Abbreviation will be used.

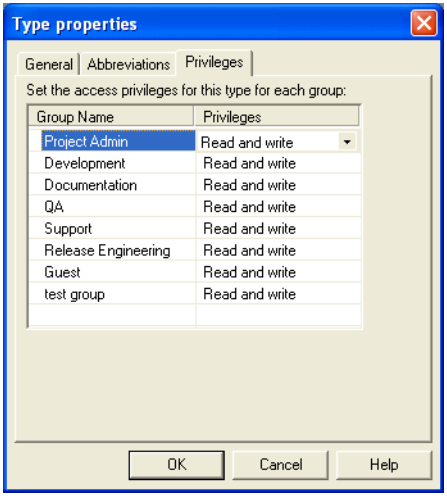
The control string is similar to a C-style control string. It can contain alphanumeric characters, which are duplicated in the abbreviation, and one or more control sequences, which are used to put information from the item into the abbreviation. TrackRecord automatically creates the control string.

Note: Any time a change is made to an abbreviation, the changes will ONLY take effect in the individual items, if they are opened for editing, then closed and saved. To update all the items with the new changes, perform a check database.

7 To restrict access to this type, click the **Privileges** tab. Select a group and use the drop-down list to select the appropriate privilege for this group.

Tip: While you can change the privilege setting for any group, you cannot increase its privileges beyond the upper limit established through Group Administration.

Figure 4-16. Type Properties Privileges Tab



Refer to "Implementing Group Access Rights" on page 49 for a general discussion of access privileges.

8 When you have completed entering information on these three tabs, click **OK** to open the Type Editor.

You can return to make changes on this dialog box by using the **Type Properties** button in the Type Editor.

Using the Type Editor

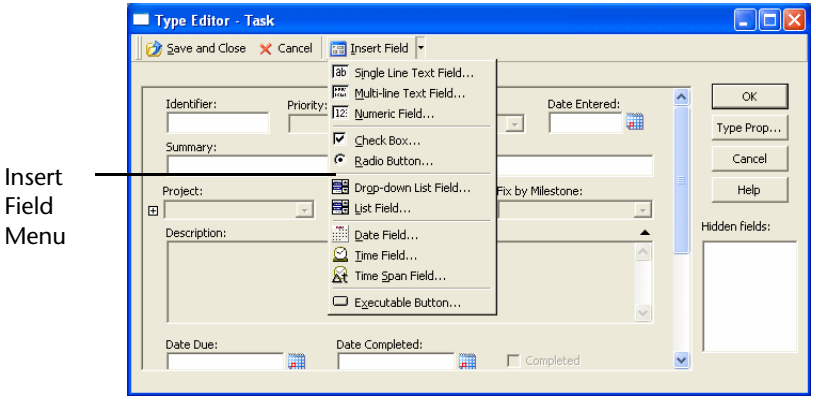
You use the Type Editor dialog box to insert fields into a form, modify fields on a form, or to modify settings on the Type Properties dialog box.

Note: When making modifications to the types via the Type Editor, it is always good practice to: make a backup or image of the database before using the Type Editor, ensure all users have exited the database, and run a Check Database after using the Type Editor. See Chapter 2, "Creating and Maintaining TrackRecord Databases" for more information on running Check Database.

Caution: If you are using the WebServer, be sure that there are no spaces at the end of a field name. This precaution will avoid errors when opening your Web page.

Follow the steps in "Creating and Modifying Types" on page 82 to display the Type Editor.

Figure 4-17. Type Editor



To modify Type properties, click the **Type Properties** button. Refer to "Creating and Modifying Types" on page 82 for information about type properties.

- To move a field, click and hold that field and drag it to another location.

- To modify the properties of a field, right-click on the field and select **Open**. The Field Properties dialog box appears. See "Field Properties-General Tab" on page 87.
- To insert a field into the type, choose the **Insert Field** menu (Figure 4-17) and select a field category from the drop-down list. A pointer is displayed in the form, indicating where the field will be added. Move the pointer around the form and click the mouse to position the field.
- To insert an existing item type into the type you are editing, choose the **Insert Field** menu. From the drop-down list, select **Single Item Combo Box** or **Multiple Item List**. A pointer is displayed in the form, indicating where the field will be added. Move the pointer around the form and click the mouse to position the field.

If the type inherits from a parent type and you place the new field in the inherited section, the new field will be added to all types inheriting from that parent.

When you drop the field into a position, a Field Properties dialog box appears. There are three tabs on the Field Properties dialog box, each of which is described in the following sections.

Field Properties- General Tab

The Field Properties General tab (Figure 4-18) displays the following options:

Figure 4-18. Field Properties - General Tab

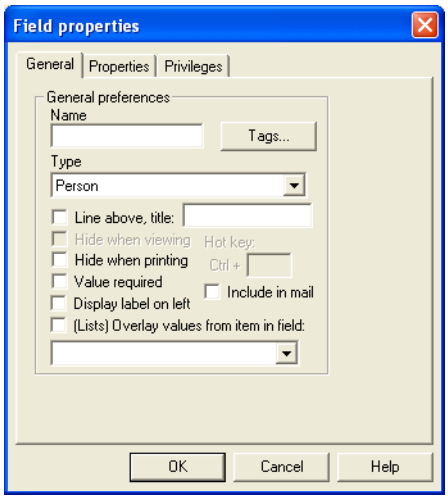


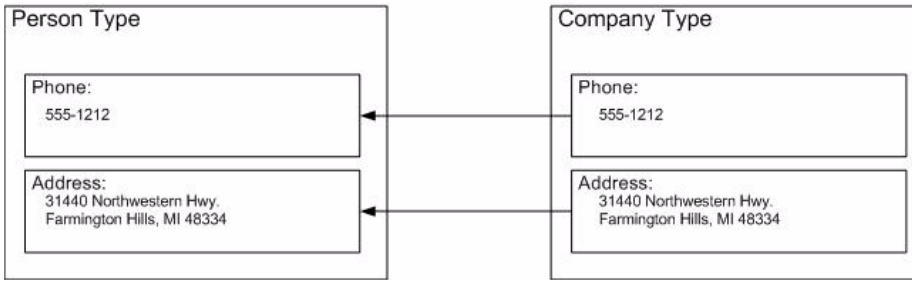
Table 4-2. Field Properties- General Tab

Option	Description
Name	Required
Type specification	In general, you cannot change the field type of an existing field. If you have no database items of the edited type, you can change its type. Selectable field types depend on the palette tool used to create the field. For instance, a field created with the date tool cannot be specified as a Phone info field type.
Line above, title	Divides form into labeled sections and allows static text by setting an optional title for the line.
Hot key	(Optional). Used to edit a field directly from within other views, without using the Item view.
Value required	Determines if it is a required field.
Display label on left	Displays label on left of field rather than above.
Include in mail	Determines if field should be included in the mail message if the AutoAlert utility is used.

Table 4-2. Field Properties- General Tab

Option	Description
(Lists) Overlay values from item in field:	If field is empty, checks the item for any fields with duplicate names. If found, its contents are displayed. However, if the contents are changed in any way, this action is ignored. The effect is similar to that of applying a template; the values from the selected item are used, but no implicit links are created. For example, this property is used to overlay the address and phone fields from a Company into any Person items that contain that Company. See Figure 4-19.

Figure 4-19. Example of Overlay Values



Field Properties- Properties Tab

The Properties tab contents will vary depending on the type of field you are creating or editing. There are four types of field properties:

- Compound Type Fields — Drop-down and multi-item list fields
- String/Notepad Fields — Text string and notepad fields
- Check box/Radio Buttons — Check box and radio button fields
- Date/Time Fields — Date- and time-related fields

Compound Type Fields

The Compound Type Properties (Figure 4-20) tab appears for any field that contains information from another type. For instance, a drop-down

list or a multi-item list box would be a compound type property. The Compound Types Properties tab displays the following options:

Figure 4-20. Field Properties for Compound Types- Properties Tab

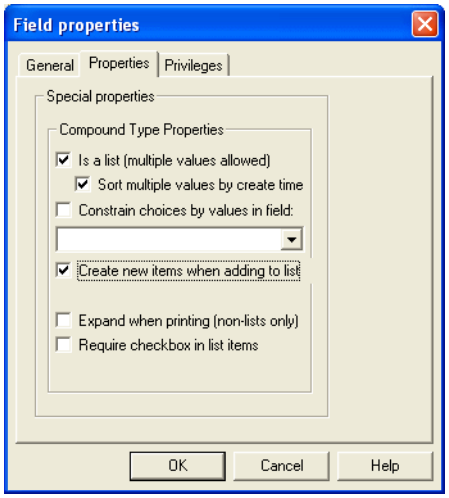


Table 4-3. Field Properties for Compound Types- Properties Tab

Option	Description
Is a list	Indicates if the field offers a list of items using a standard Windows list box. Once the list box property has been applied to a field, it cannot be removed.
Sort Multiple Values By Create Time	Select this check box to have items in a multi list field remain in the order in which they were created. These items can not be moved up and down in the field.
Constrain choices by values in field	When choosing an item for this field, TrackRecord filters the items shown according to this item. For example, the Fix by Milestone field in a Defect Report could be constrained by the Project field. Then, a list of Fix by Milestone Defects would be limited by Project.
Create new items when adding to list	(Only applicable for list fields) If selected, a new item of the labeled type is automatically created when the Add or Ins button is clicked, and deleted from the list when the item of that type is removed.
Expand when viewing	By default, non-list fields should be expanded inline, showing all of the sub-fields when viewing an item.
Expand when printing	By default, non-list fields should be expanded inline, showing all of the sub-fields when printing an item.

Table 4-3. Field Properties for Compound Types- Properties Tab

Option	Description
Require check box in list items	By default, list fields do not require a check box.

String/Notepad Fields

The String/Notepad Properties tab (Figure 4-21) appears for all string and Notepad fields. The String/Notepad Properties tab displays the following options:

Note: The control string is a text string that is used to create the identifier. You should use a pound sign (#) in the string where you want the number to appear. For example, to get identifiers of the form "Track-Record-00001", you would use the control string "TrackRecord-#". The optional integer specifies the current value for the counter. You can change this value at any time.

Figure 4-21. Field Properties for Strings/Notepads- Properties Tab

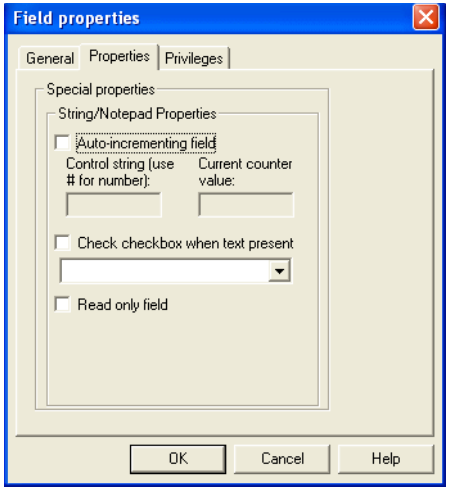


Table 4-4. Field Properties for Strings/Notepads- Properties Tab

Option	Description
Auto-increment-ing field	Automatically assigns field value when an item containing the field is initially created. This attribute is useful for creating fields like task identifiers, when you want a unique number to identify each new task.
Control string	Text string used to create the identifier. Use a pound sign (#) in the string where you want the number to appear. For example, to get identifiers of the form "TrackRecord-00001", you would use the control string "TrackRecord-#".

Table 4-4. Field Properties for Strings/Notepads- Properties Tab

Option	Description
Current counter value	Optional integer that specifies the current value for the counter number. You can change this value at any time.
Check check box when text present	Named field should be checked if any text is present in the field, and should be unchecked if no text is present.
Read only field	Makes field read-only. Use only with auto-incrementing fields, or with fields that will automatically be filled from templates in all cases.

Check box/Radio Buttons

Check box/Radio Button Properties (Figure 4-22) tab appears for all check box and radio button fields. The Check box/Radio Button Properties tab displays the following options:

Figure 4-22. Field Properties for Check boxes/Radio Buttons- Properties Tab

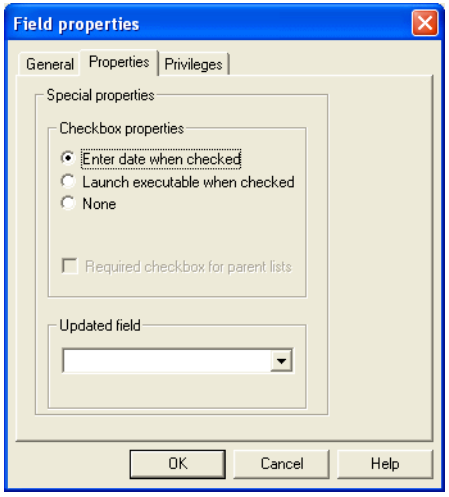


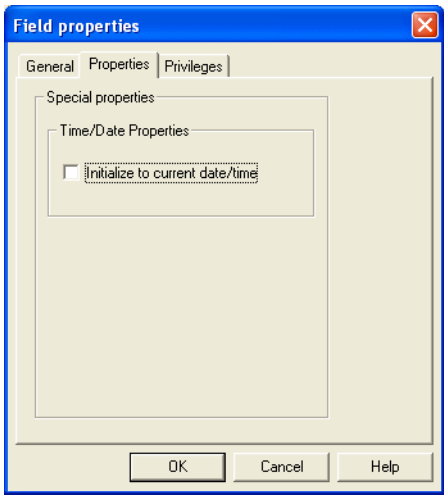
Table 4-5. Field Properties for Check boxes/Radio Buttons- Properties Tab

Option	Description
Enter date when checked	Named field, which must be a Date or a Time/Date field, should contain the current time and date when checked.
Launch executable when checked	Launches specified executable file. The list of executable files you can specify appears under Updated Field.
None	Clears previously check options.
Updated fields	Indicates which fields will be updated.

Date/Time Fields

The Date and Time Properties (Figure 4-23) tab appears for all date- and time-related fields: dates, times, and time spans. The **Initialize to Current Date/Time** check box initializes the value in the field to the current date and time when a new item with this field is created, or duplicated from an existing item.

Figure 4-23. Field Properties for Dates- Properties Tab



Executable Buttons

The Special Properties (Figure 4-24) tab appears for all Executable button fields. Administrators can add an executable button to run custom built applications such as an application that uses the TrackRecord ActiveX interface. This executable button will launch any executable or batch file.

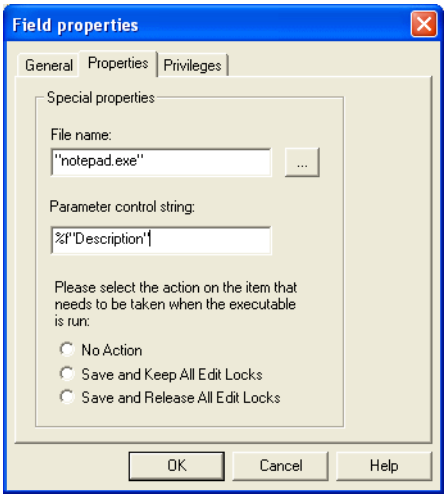
- In the **File name** field, enter the file name of the program that the executable button will open. For example, enter "notepad.exe" to have the button launch the Windows Notepad accessory.
- In the **Parameter control string** field, specify the command line parameters with which the executable file can be called. For example, the entry `"test.txt"` will launch Notepad and open the file named `test.txt`, or will create a new `test.txt` if none exists.

  You can also enter the `%f "<fieldname>"` parameter, where <*field-name*> is the name of the field you wish to pass. In Figure 4-24, the button would launch the Windows Notepad accessory, go to the Description field and pass the contents of the field as a parameter to the executable.

*Tip: Using double quotes ("") in the **File name** field will allow TrackRecord to start the parameter using the program that recognizes the associated file extension. For example, when creating an executable button that specifies a URL, type "" as the filename and enter %f"name of the field containing the URL" as the parameter control string. This enables the executable button to launch the browser and retrieve the specified URL.*

Another option is the `%i "<identifier>"` parameter. The identifier of the current item would be passed as a parameter.

Figure 4-24. Field Properties for Executables- Properties Tab

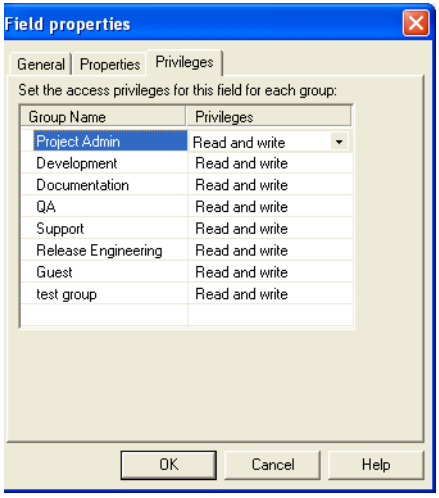


Field Properties- Privileges Tab

The Privileges tab allows you to specify which groups will have access to this field. Refer to "Altering Access Rights for Types and Fields" on page 77 for a discussion of access privileges.

Figure 4-25. Field Properties - Privileges Tab

Tip: While you can change the privilege setting for any group, you cannot increase its privileges beyond the upper limit established through Group Administration.



1. In the **Group Name** column, select the group whose access rights you want to modify.
2. In the **Privileges** column, select the new privileges setting from the drop-down list and click **OK**.

   You can use the General tab to hide a field and control access rights to a field. Making a field invisible does not disable the field and a template could supply a value to that field, but the template supplied field would not be visible.
3. When all Field Properties have been set, click **OK** to see the field on the type form. When all fields have been added, you can set Abbreviations, if desired, as described in step 6 on page 83.
4. When all type properties and fields have been set, click **Save** and then **Close**. The Choose a Type dialog box appears, showing the newly created or modified type.

Restrictions to Modifying Types

TrackRecord places some restrictions on the modifications that administrators can make to information types.

- You can delete only the last field added to a type if items of this type already exist in the database.
- You cannot move an inherited field past the boundary of the parent fields when a type inherits from another type. (Note that you can, however, add a new field to the inherited section.)

Custom Items

The TrackRecord sample database ships with ten Choice types, including Department, Priority, Severity, and others, and several choice items for each of these types. For example, the Priority type includes the choices Critical, High, Normal, Low, and Suggestion. You can use these Priority items, or you could, for example, delete the Suggestion choice item and create a new choice item called Wish List.

Note: Status and Action choice items should be set through the Workflow Editor rather than through this procedure. Refer to "Workflow Administration" on page 60.

Creating or Deleting a Custom Item

1. From the **Tools** menu, choose **Item Browser**.
2. Select the **Show All** check box. ****This function is not available in TrackRecord during a CARS engagement.****

3 Under the **Choice Fields** type, select the type for which you want to create a custom item. For example, you could select the Priority type to create a custom item. A list appears under Items.

4 To delete an item, right-click on the item and select **Delete**. For example, you could right-click on the "5 - Suggestion" item of the Priority type, click **Delete** and click **Yes** in the confirmation dialog box.

5 To create a new item, click the **New** button in the upper left section of the Item Browser toolbar. An item form appears.

6 Enter the information for the new item. For example, for a Priority item you could enter "5" in the Name field, and "Wish List" in the Description field. Other item types, such as Department, only contain a name field.

7 Click **Save and Close**.

Modifying a Custom Item

1 From the **Tools** menu, choose **Item Browser**. The Item Browser dialog box appears.

2 Select the **Show All** check box. ****This function is not available in TrackRecord during a CARS engagement.****

3 Under **Choice Fields**, select a type to modify. For example, select Department in the Types list to modify a Department item.

4 Double-click the item to be modified. For example, double-click the Documentation item.

5 On the item form, make any modifications. For example, you could replace the Name "Documentation" with "Publications."

6 Click **Save and Close**.

Rules Administration

TrackRecord provides a Rules Engine that permits creating policies that govern the behavior of information types. Rules require exclusive database access, as do Check Database, Edit Types and Edit Workflow. Administrators can use rules to require that certain fields be completed always or when specific conditions are met. For example, an administrator might create a rule that states, "If Priority is equal to Critical, then Fix by Date is required."

This allows you to further customize your workflow by implementing rules that disable the standard workflow in certain situations. Disabling the workflow causes the Action button to be disabled, which will prevent a user from changing the status of an item even if the standard workflow would normally permit them to do so.

Note: **This function is not available in TrackRecord during a CARS engagement.**

Notes About Rules

- Administrators create rules for individual types, and can create multiple rules for any given type.
- Rules apply either globally, or to users who belong to specified groups.
- Rules use the same operators as TrackRecord queries, except for the "is between" operator.
- A child type inherits any rules created for its parent. To avoid the creation of child data types that are identical to parent types, ensure that there are additional fields in the child type.
- Inherited rules are applied before a type's own rules.
- A type's own rules are applied in the order they appear within the Rules editor, and administrators can rearrange this order.
- Multiple conditions within a single rule provide an AND logical operation and all the conditions must be true for the rule to be applied. Separate rules provide the logical OR operation.
- Rules are applied upon: a loss of focus on a field, a change to a radio button or check box, and the execution of a workflow action based on the contents of the item before the action.
- Rules take the form:

  ```
  IF {condition}, THEN {action}
  ```

- Rules are not invoked via a change produced by another rule. For example, if you create one rule that states that selecting check box one automatically selects check box two, and another rule which states that selecting check box two automatically selects check box three, then when check box one is selected by the user, only check box two will be automatically selected, not check box three.

Adding a Rule

Use the following procedure to create a rule:

1 From the **Administrator** menu, choose **Edit Rules**. The Choose a Type dialog box appears.

Note: It is always good practice to make a backup or image of the database before creating a rule.

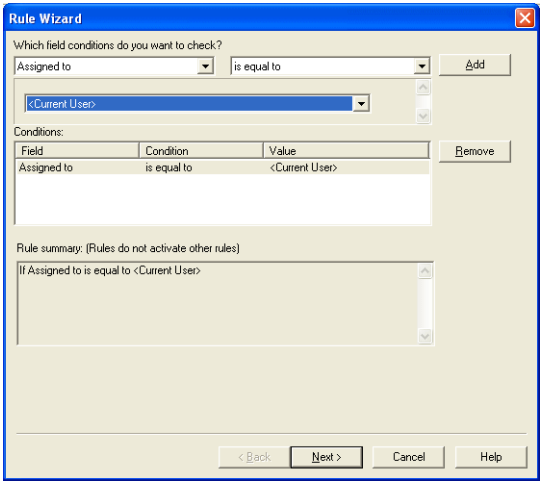
2 Double-click a type to display the Rules dialog box.

3 Click the **New** button to add a new rule and open the Rule Wizard.

4 Use the Rule Wizard (Figure 4-26) to build the rule.

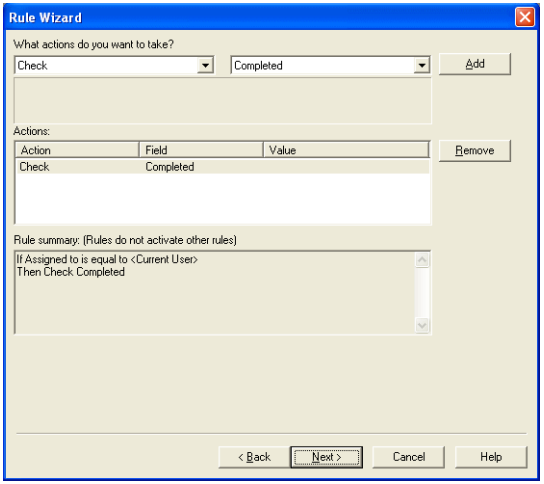
5 Create the IF condition by selecting an attribute from each of the drop-down lists. Click the **Add** button when it is complete. The new rule will appear in the Conditions list. Click **Next**.

Figure 4-26. Creating an IF Condition



6 Create the THEN condition designating the action that you want taken by selecting an attribute from each of the drop-down lists (Figure 4-27).

Figure 4-27. Creating the THEN Condition



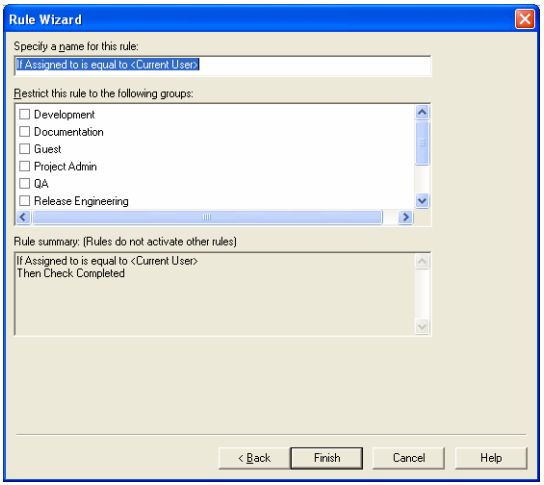
THEN options include:

- **Check**— Add a check to a check box
- **Clear**— Clear contents in a field
- **Disable**— Make a field unavailable
- **Enable**— Make a field available
- **Require**— Item cannot be saved unless the specified field has an entry
- **Set**— Assign a value to a specified field. This is only applicable for single choice fields where the possible values are predetermined.
- **Store Date/Time**— Enter a timestamp in a specified field
- **Uncheck**— Clear a field with the property check box

7 Click the **Add** button. The new condition will appear in the Action list. Click **Next**.

8 Enter a name for the rule, select the groups to which the rule will apply, and click **Finish** to close the Wizard (Figure 4-28).

Figure 4-28. Selecting Groups

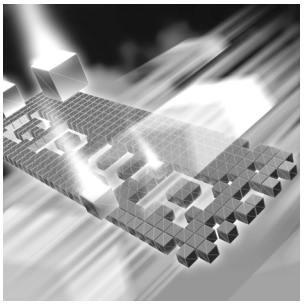


9 When multiple rules exist, use the **Move Up** and **Move Down** buttons on the Rules dialog box to rearrange the order in which rules are applied.

10 Click **OK** to exit and save your work.

Chapter 5

Administering AutoAlert



AutoAlert Administration Overview

TrackRecord includes many features to help simplify, accelerate, and improve team communications. One of these features is the TrackRecord AutoAlert utility. It provides email notification of reported defects and any important information that enters the TrackRecord database. AutoAlert allows you to define flexible criteria for notifying each user of changes that might be of interest to them. AutoAlert can also be used to:

- notify off-site developers about new problems
- alert testers and developers when the status of their reported defects change
- update managers when critical defects are found

Required: After the AutoAlert software is installed, you must create a mailbox and mail profile for AutoAlert's use. Refer to your mail service documentation for information on creating a mailbox and profile.

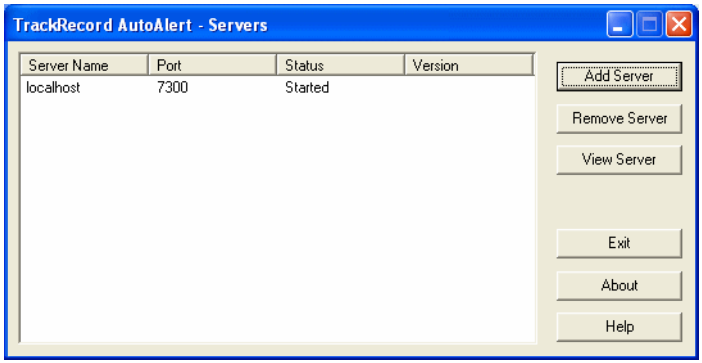
About the AutoAlert Administration Utility

The AutoAlert Administration Utility allows you to add and remove databases to be polled by the AutoAlert service, add and remove servers, show the users that are AutoAlert enabled, and set a variety of options.

Starting the AutoAlert Administration Utility

On a machine on which the AutoAlert Administration Utility has been installed, click the taskbar's **Start** button. From TrackRecord's program files, choose **AutoAlert Administration Utility**. The TrackRecord AutoAlert-Servers dialog box (Figure 5-1) appears.

Figure 5-1. Servers Available for AutoAlert (Example)



Adding and Removing Servers

1. Start the AutoAlert Administration Utility. See "Starting the AutoAlert Administration Utility" on page 102.
2. From the TrackRecord AutoAlert-Servers dialog box (Figure 5-1), you may:
   - ◇ Click the **Add Server** button to add an AutoAlert server. A dialog box will appear prompting you to enter the AutoAlert server name and port number on which the AutoAlert server is listening. Click **OK** when finished.
   - ◇ Select a server from the list and click the **Remove Server** button. A confirmation screen will appear allowing you to continue or cancel the server removal process. The server is only removed from the available server's list.

3 Click the **Exit** button to close the AutoAlert Administration Utility or continue to configure the server options.

AutoAlert Server Options

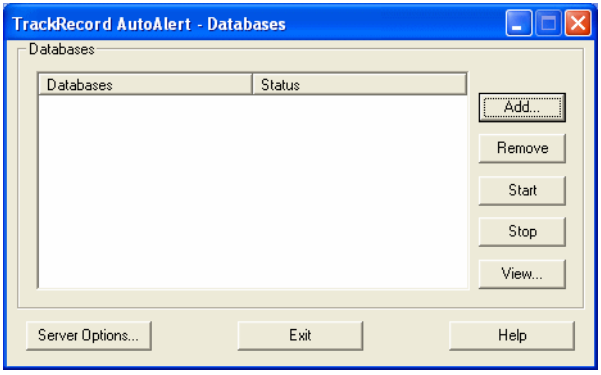
1 Start the AutoAlert Administration Utility as described on page 102.

2 Select one of the available AutoAlert servers and click the **View Server** button (Figure 5-1).

Note: If password protection was selected, enter the password for the AutoAlert server.

The TrackRecord AutoAlert-Databases dialog box appears (Figure 5-2). From this dialog box, you may add, remove, start, stop, and view databases associated with the TrackRecord server on this machine as well as configure server options.

Figure 5-2. TrackRecord AutoAlert-Databases Dialog Box



Adding Databases

1 Follow the procedures for "AutoAlert Server Options" on page 103.

2 Click the **Add** button to identify the database that the AutoAlert server is to poll. AutoAlert will search for items in the database matching the TrackRecord queries. The TrackRecord AutoAlert-Add Database dialog box appears.

3 In the **Database** field, enter the database name.

4 Click **OK** when you are finished. The associated TrackRecord server will verify the existence and accessibility of the TrackRecord database.

5 Repeat step 2–4 for each database.

6 Click the **Exit** button on all remaining dialog boxes to close the AutoAlert Administration Utility.

Removing Databases

1 Follow the procedures for "AutoAlert Server Options" on page 103.

2 To remove the database from the list, select the database name and click the **Remove** button.

3 A confirmation dialog box will appear to allow you to continue or cancel the database removal process. Click **OK** to remove the database from the list. If there is a query processor running for this database, the query processor will be stopped.

4 Click the **Exit** button on all remaining dialog boxes to close the AutoAlert Administration Utility.

Starting Database Polling

1 Follow the procedures for "AutoAlert Server Options" on page 103.

2 Select the database on which you wish to start AutoAlert service and click the **Start** button. Note that the status changes to **Started**.

3 Click the **Exit** button on all remaining dialog boxes to close the AutoAlert Administration Utility.

Stopping Database Polling

1 Follow the procedures for "AutoAlert Server Options" on page 103.

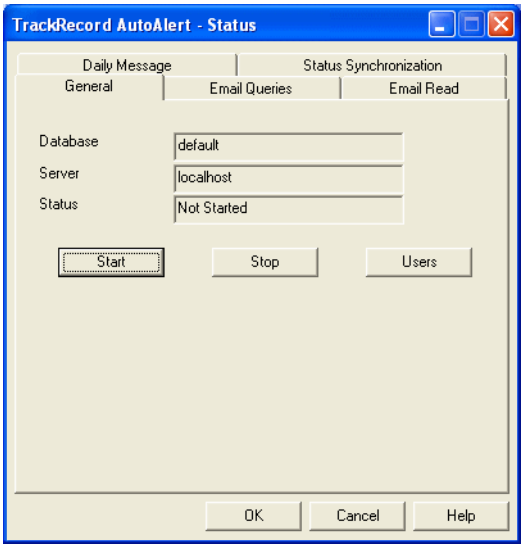
2 Select the database you wish to stop and click the **Stop** button. Note that the status changes to **Not Started**.

3 Click the **Exit** button on all remaining dialog boxes to close the AutoAlert Administration Utility.

Viewing Database Options

1. Follow the procedures for "AutoAlert Server Options" on page 103.
2. Select the database you wish to configure and click the **View** button. The TrackRecord AutoAlert-Status dialog box appears (Figure 5-3) with the contents of the **General tab** displayed.

Figure 5-3. TrackRecord AutoAlert-Status Dialog Box (General Tab)

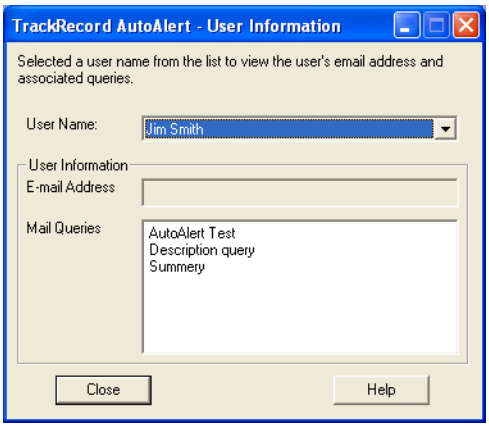


- ◇ Click the **Start** button to begin AutoAlert's polling of the database. The status changes to **Scheduling Started**.
- ◇ Click the **Stop** button to stop AutoAlert's polling on the database. The status changes to **Scheduling Not Started**.

Required: In order to further configure the AutoAlert database polling, the status must be set to **Scheduling Started**.

◇ Click the **Users** button to display the list of users enabled for email processing. The TrackRecord AutoAlert-User Information dialog box appears (Figure 5-4).

Figure 5-4. TrackRecord AutoAlert-User Information Dialog Box



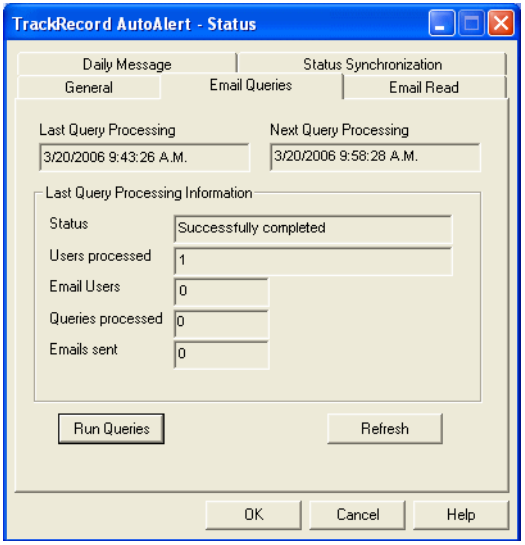
– Select a user name from the drop-down list to display the associated user's email address and email queries.

– Click **Close** to return to the General tab.

Note: Any changes to the **User name** list, or to the properties of any mail enabled user, must be made in TrackRecord. No user information can be changed from the AutoAlert interface. Once a status change is made in TrackRecord, AutoAlert automatically updates the User list.

3 Click the **Email Queries** tab (Figure 5-5).

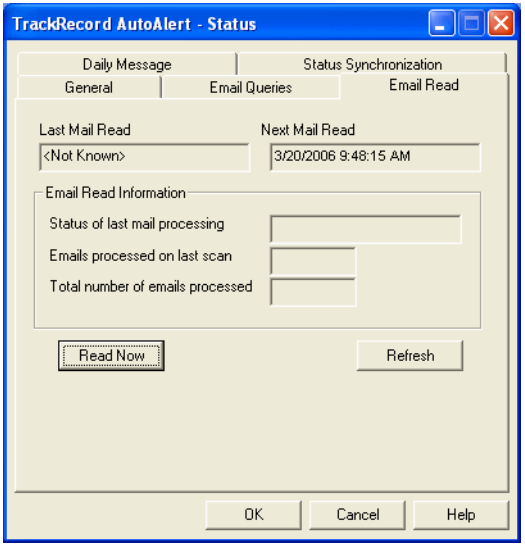
Figure 5-5. TrackRecord AutoAlert-Status Dialog Box (Email Queries Tab)



- ◇ Click the **Run Queries** button to force the immediate execution of email queries. An error message will appear if the database processor is stopped.
- ◇ Click the **Refresh** button to refresh the dialog box with current information from the database.

4 Click the **Email Read** tab (Figure 5-6).

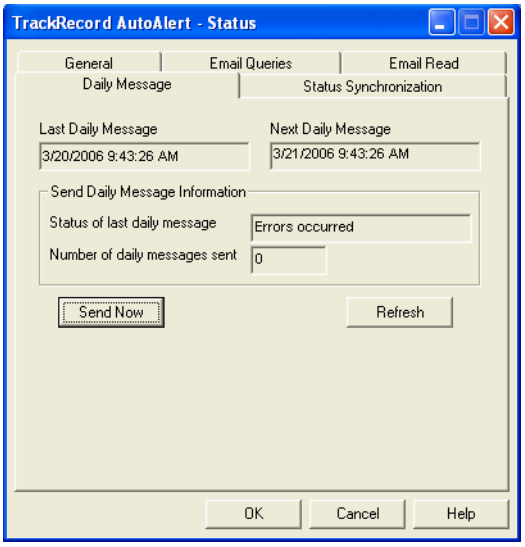
Figure 5-6. TrackRecord AutoAlert-Status Dialog Box (Email Read Tab)



- ◇ Click the **Refresh** button to refresh the dialog box with current information from the database.

5 Click the **Daily Message** tab (Figure 5-7).

Figure 5-7. TrackRecord AutoAlert-Status Dialog Box (Daily Message Tab)

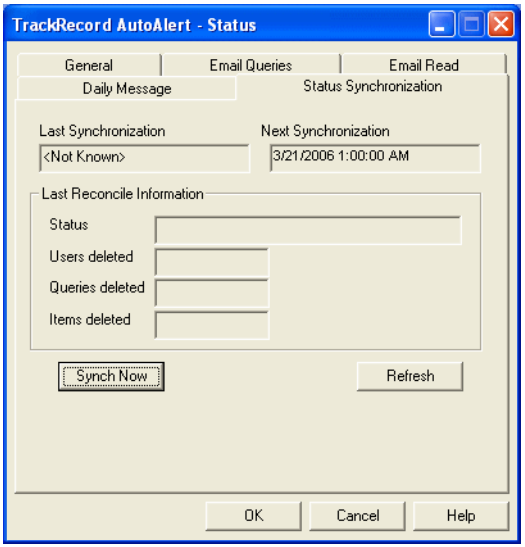


◇ Click the **Send Now** button to send the daily message immediately. For more information on configuring your daily message, see "Configuring AutoAlert Server Options" on page 109. An error message will appear if the database processor is stopped.

◇ Click the **Refresh** button to refresh the dialog box with current information from the database.

6 Click the **Status Synchronization** tab (Figure 5-8).

Figure 5-8. TrackRecord AutoAlert-Status (Status Synchronization Tab)



- ◇ Click the **Synch Now** button to synchronize the status file and the database. An error message will appear if the database processor is stopped.
- ◇ Click the **Refresh** button to refresh the dialog box with current information from the database.

7 Click **OK** when you are finished.

Configuring AutoAlert Server Options

1 Follow the procedures for "AutoAlert Server Options" on page 103.

2 Click the **Server Options** button. The TrackRecord AutoAlert-Options dialog box appears.

3 On the **Mail Frequency** tab, use the slider bar to specify the frequency (from ten minutes to eight hours) that AutoAlert checks for new or changed items. By default, AutoAlert will send mail every 15 minutes. A higher setting will result in longer delays before mail notification is sent.

Note: For larger groups (20 or more mail users), it is recommended that you use a polling frequency of 30 minutes or higher. TrackRecord and your mail package may also have polling frequencies that could result in additional delays before AutoAlert sends messages.

4 Click the **Message Threshold** tab to define the threshold at which the AutoAlert server will consolidate multiple query results into a single email message.

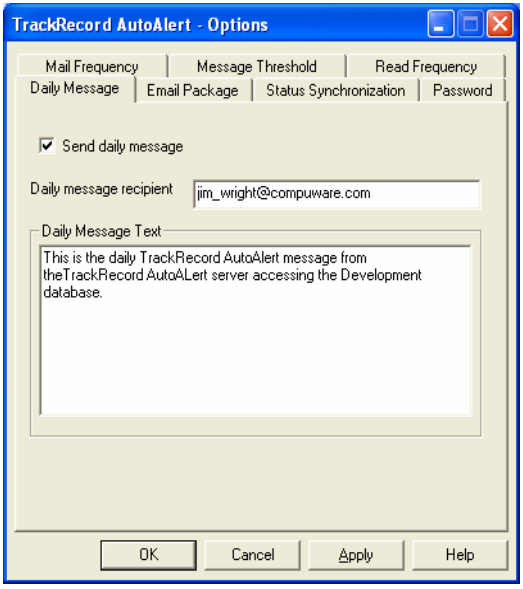
The value can be set from 1 to 100 messages, with the default setting of 15 messages. A lower setting means you will receive fewer mail messages, but some messages will have multiple item notifications in them. A higher setting makes it more likely that each notification will result in a separate mail message.

5 Click the **Read Frequency** tab.

- ◇ Use the slider bar to specify the frequency at which AutoAlert checks for new mail entries. These mail entries contain new information if users add new defects to TrackRecord. The default is every five minutes. If using AutoAlert with a SMTP server, the frequency must be set to 0 for this feature to function properly.

6 Click the **Daily Message** tab (Figure 5-9).

Figure 5-9. TrackRecord AutoAlert-Options Dialog Box (Daily Message Tab)

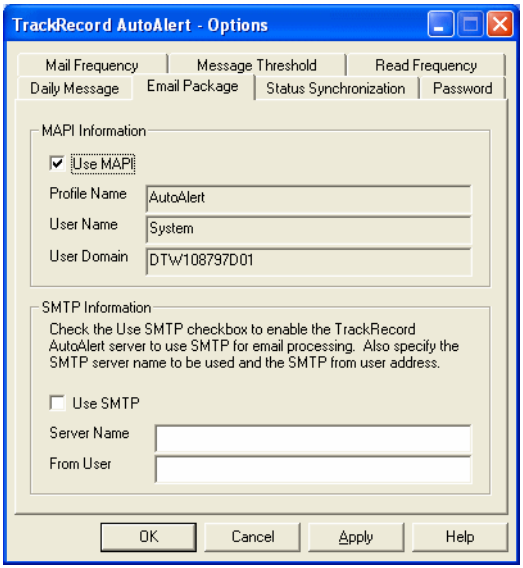


- ◇ Select the **Send daily message** option to send a daily TrackRecord AutoAlert status message.
- ◇ Enter a recipient's email address (commonly the administrator).
- ◇ Enter a message in the **Daily Message Text** field.

Note: If a user does not specify a message, the default text will state: This is the daily automated message from TrackRecord AutoAlert running on machine: xxxxxx.

7 Click the **Email Package** tab to define what email package will be used to send email messages (Figure 5-10).

Figure 5-10. TrackRecord AutoAlert-Options Dialog Box (Email Package Tab)



- ◇ **MAPI** — Select the **MAPI** check box. Enter the profile, user name, user domain, and password.
- ◇ **SMTP** — Select the **SMTP** check box. Enter the name of the SMTP server and the address of the user sending the email.

8 Click the **Status Synchronization** tab.

- ◇ Select the appropriate radio button to designate the interval at which the AutoAlert server should synchronize data in the status file and the associated database.
- ◇ Select the time for synchronization.

9 Click the **Password** tab to set up a required password to gain access into the associated AutoAlert server.

- ◇ In the **Password** field, enter a password. If a password is not specified, no password will be required.
- ◇ In the **Confirm Password** field, enter the same password.

10 Click **Apply** and **OK** when your settings are complete.

Configuring AutoAlert Users

For users to receive email notification, their email address must be included in their TrackRecord login information. If an email address was not entered when the user's Team Member item was created, do the following:

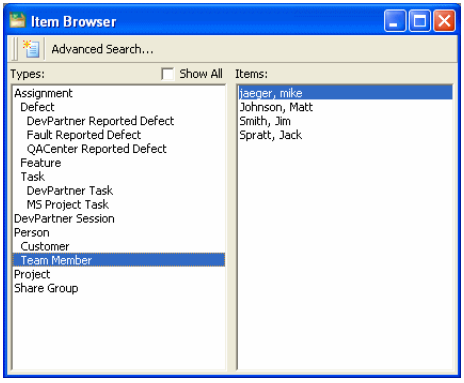
1 Start TrackRecord and log on as a user with administrator privileges as described on page 18.

2 Open the Item Browser.

3 Select **Team Member** from the Types list (Figure 5-11).

Tip: Refer to "Creating Team Members" on page 51 for additional information.

Figure 5-11. Using the Item Browser



◇ If the user is listed in the item list, click on the user's name.

◇ If the user is not listed, click the **New** button in the tool bar. Follow the steps for "Creating a User" on page 53.

Setting Up Mail Queries

A mail query is a TrackRecord query in which the **Email Me When New Items Match This Query** option has been selected. Each AutoAlert recipient must construct a set of queries that describe the criteria for email notification. For example, most users will probably want a mail query that selects any items assigned to them. The query editor provides an Options dialog box with a setting for sending out email whenever a new or changed item matches the query.

Queries and Non Admin Users

If an owner of a query selects the **Email me** option, AutoAlert will send email notification only to the owner. These queries can only be sent to the owner and not to other group members.

Queries and Admin Users

Only Admin users may share mail queries and send notifications to groups of users, or all users. This prevents users without administrative authority from making changes that could affect all of the TrackRecord users. It also prevents the sending of extraneous emails. The Admin options are as follows:

Table 5-1. Admin Options for Sharing a Mail Query

Options	Behavior
Email me option selected, query is Not Shared	AutoAlert will send notification to the Admin only
Email me option selected, query is Shared to All	AutoAlert will send notification to every user
Email me option selected, query is Shared to one or more groups	AutoAlert will send notification to members of that group

Mail Queries for Individual Users

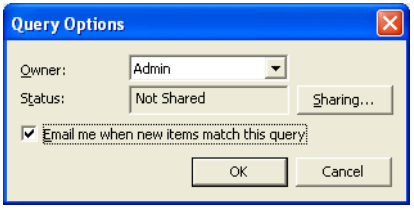
1 Start TrackRecord and log in as the user you are configuring. To log on as a specific user, close all open windows, select **Tools>Change Login**.

Tip: The name of the current user is located to the right of the status line.

2 Create a query that selects items that this user would be interested in seeing.

3 Click the **Options** button. The Query Options dialog box (Figure 5-12) appears.

Figure 5-12. Query Options Dialog Box



4 Select the **Email Me When New Items Match This Query** option.

5 Repeat step 2–4 for any other queries that you want to create.

For example, most users will probably want a mail query that selects any items assigned to them.

6 Repeat this process for each AutoAlert recipient.

<current user> Feature

If the query includes the *<current user>* criterion, the email will be specific to each TrackRecord user. This feature is available only to the Admin, not to any other user with administrator privilege.

For example, assume that there are three TrackRecord users: Joe Smith, Kathy Simpson, and Jane Brown. The Admin creates the query to retrieve all defects where *[assigned to]=<current user>*, shares it to all, and selects the **Email me** option. AutoAlert will run the query in the following manner:

1 AutoAlert runs query "Retrieve all [defects] where [assigned to] = Joe Smith. Joe will receive an email with the results.

2 AutoAlert runs query "Retrieve all [defects] where [assigned to] = Kathy Simpson. AutoAlert will email Kathy with the results.

3 AutoAlert runs query "Retrieve all [defects] where [assigned to] = Jane Brown. Jane will receive an email with the results.

Changing the Contents of Mail Messages

The message sent by AutoAlert consists of the default abbreviation for the type of item that triggered the notification and any fields from that item that are marked with the **Include in Mail** property. You can change the fields included in AutoAlert messages.

Changing Fields in Mail Messages

1 Shut down the AutoAlert service and log on to TrackRecord as an Administrator.

2 Close all TrackRecord items and views (Defects, reports, and so on).

3 From the **Administrator** menu, choose **Edit Types**.

A warning is displayed advising you to backup your database before modifying types. Refer to "Scheduling Backups of Databases" on page 30 if you would like to backup your database at this time.

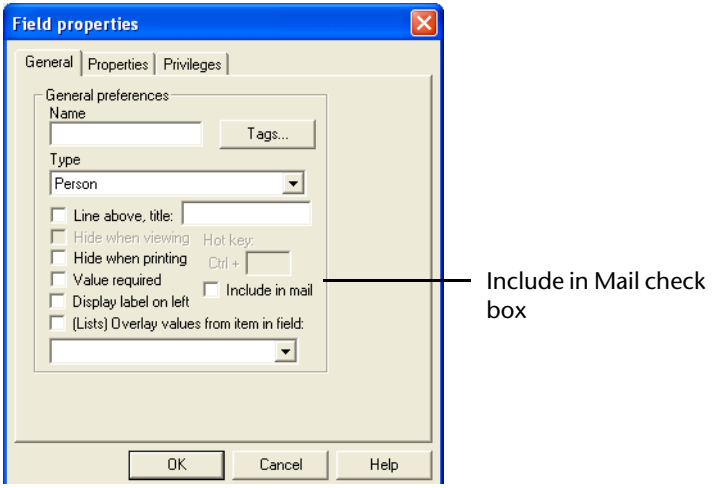
If users are logged on using this database, a dialog box will be displayed listing current users. You cannot edit types while users are using the database. Refer to "Active Users in the Database" on page 79 for information about this dialog box. Once all users have logged off, this dialog box is dismissed and the Choose a Type dialog box appears.

4 Open the type you want to change by double-clicking its name.

5 Double click on the field you want included in mail messages. The Field Properties dialog box appears.

6 On the **General** tab (Figure 5-13), select the **Include in Mail** check box to have this field included in mail messages sent for this type.

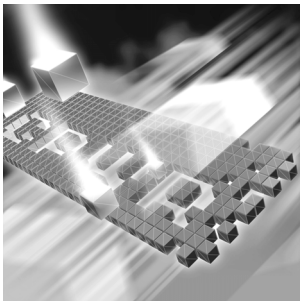
Figure 5-13. Field Properties Dialog Box (General Tab)



7 Exit all dialog boxes and restart the AutoAlert service.

Chapter 6

Administering WebServer



WebServer Administration Overview

WebServer allows users to access a TrackRecord database using a Web browser such as Microsoft Internet Explorer. The WebServer user interface is described in the *TrackRecord User's Guide*. The following sections describe features available only to users with Administrator privilege.

Note: All users authorized to use the Windows TrackRecord client are also authorized to use the WebServer client. See "Team Member Administration" on page 50 to create a new Web user who is not already established in TrackRecord.

Tip: *Refer to the Installation Guide for instructions regarding installing the WebServer.*

The following sections contain information that will assist you in interpreting the statistics available through the WebServer Administrator's menu. A thorough understanding of this information is **not** required to successfully administer WebServer, but can help you assess system performance and requirements.

WebServer Processes

When a user logs in to WebServer, a session manager process—*TRWebSessionsHost*— is automatically created. TRWebSessionsHost assigns a session identifier to each logged in client and assigns a license to that session.

When the user makes a request to the database, WebServer creates a Database Host (DBHost) process, *TRWebDBHost*. Or, it adds the user's session to an existing DBHost process. A DBHost process caches values for a specific database. When a WebServer reaches its threshold for the number of sessions it can handle, the WebServer creates another DBHost.

Some important notes about DBHost:

- Each DBHost services requests for only one database.
- There can be multiple DBHosts for each database.
- Each DBHost can serve multiple sessions.

As users log off, their sessions are terminated and their licenses become available for use by other users. When there are no more user requests, the TRWebSessionsHost and TRWebDBHost processes will shut down automatically.

Permanent and Transient Sessions

The session identifier assigned by the TRWebSessionsHost process is kept for the entire life of the client session. This is the user's permanent session. It identifies this user's requests until the user logs off to terminate the session.

When the user makes a first request, WebServer creates a *transient* session. Transient sessions are data structures that cache data from users for a short period of time. They are created for efficiency. If the transient session is unused for ten minutes (i.e., there have been no requests from this client), the transient session times out and is removed. If the client subsequently makes a request, a transient session is recreated from the information in the permanent session.

Monitoring permanent sessions and transient sessions separately allows you to accurately assess the requirements of your server and users.

WebServer Log Files

The Web server (IIS, Personal Web Server) on which you are running the TrackRecord WebServer creates a log file. Analysis of this log file can help you ensure that your system is sized properly.

Since all WebServer requests are logged as CGI requests, be sure to enable CGI request logging in your log file.

Note: Logging of HTML page and GIF file requests are generally enabled; CGI requests may not be enabled by default.

Ending a Session

There are several ways for a user to end a WebServer session including clicking the **Logout** link, closing the browser, accessing a different URL, and clicking a **Home** button. In these cases, the session is terminated and eventually all system processes are stopped.

To ensure that sessions are terminated and processes stopped in situations such as a loss of power in the client or server machine, WebServer maintains a license beat between the WebServer application client and the WebServer. If the license beat is interrupted in one of the following ways, sessions are terminated and system processes are stopped:

- If the client's attempt to send a license beat fails four times in a row, the client will stop sending license beats.
- If a license beat response contains an invalid session ID error, the client will stop sending license beats. (This error could be generated, for example, if the NT server running the TrackRecord WebServer has crashed.)
- If the server has not received a request (minimally, a license beat) from a client for an extended time, the server will terminate the session.

If a user attempts to access the database after their session is terminated, they will receive a message that the application has terminated and be prompted to log in again.

WebServer Administration Menu

Using the WebServer interface, the WebServer administrator can monitor server performance and perform other administrative functions. To perform administrative functions, follow these steps:

1. Open a Web browser, such as Microsoft Internet Explorer.
2. Enter the following address:

   `http://www.`*`servername`*`/TrackRecord/Welcome.html` where *servername* is the name of the server on which WebServer is installed.

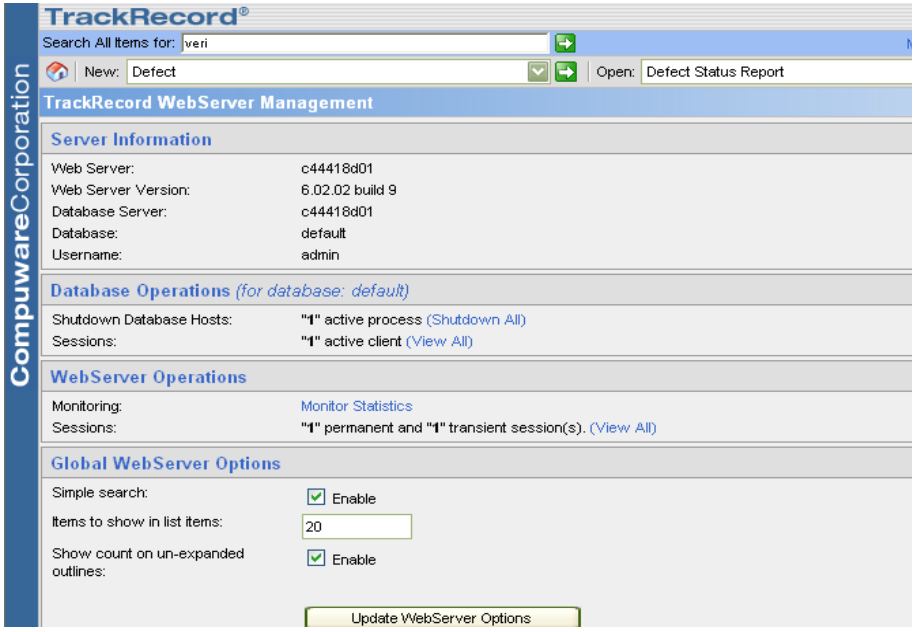
3 On the TrackRecord WebServer Login page, enter the name of your TrackRecord database, your user name and your password.

Note: Be sure to log in as a user with Administrator privilege.

4 Click **OK** to open the TrackRecord WebServer main window.

5 To perform administrative functions, click the **Manage** link in the top frame of the window. The Administration options are displayed, as shown in Figure 6-1.

Figure 6-1. WebServer Administrative Options



Administrator Options

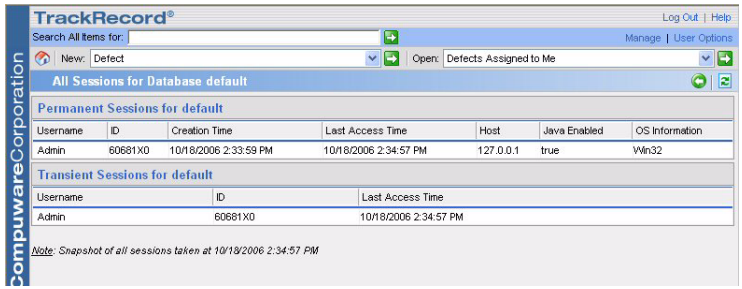
Under Database Operations, the options are:

- **Shutdown All** - Click this to shut down the Database Hosts for the named database. (Refer to "WebServer Processes" on page 118 for a description of database hosts.) After clicking, a confirmation window appears. Confirm that all processes for the database default should be shut down. Database Hosts will be recreated when a request, such as a license beat, is received. User sessions are not terminated.

Tip: In addition to the options described in this section, the Web Monitor utility can be used to shut down all WebServer processes in the event of a catastrophic failure. Refer to "WebMonitor" on page 122 for information about the Web Monitor utility.

- **View All** - Click this to view a list of the users currently accessing this database through the TrackRecord WebServer (Figure 6-2).

Figure 6-2. View Sessions

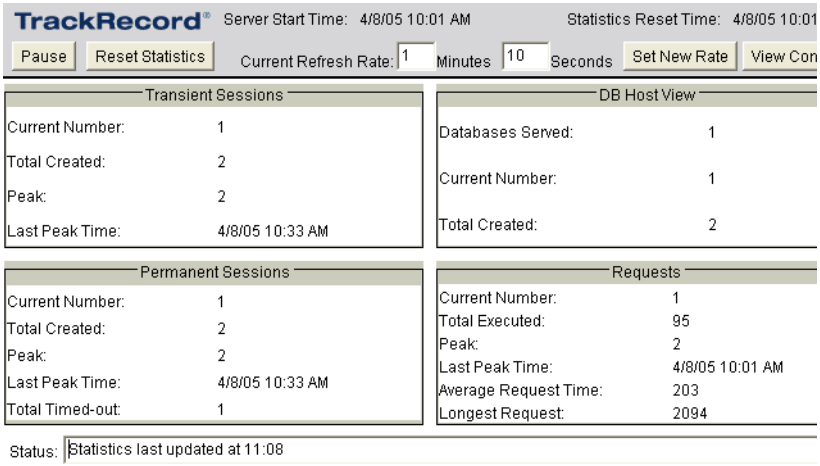


All permanent and transient sessions accessing this database are listed, as well as the time of the last request from each session to the WebServer. Click the **Refresh** button to update the information displayed. Click **Close** to return to the WebServer Administrator menu.

Under WebServer Operations, the options are:

- **Monitor Statistics** - Click this option to monitor permanent and transient sessions, database hosts, and the requests being issued to the WebServer. The monitor window appears, as shown in Figure 6-3.

Figure 6-3. Monitor Window



  - Click the **Pause** button to prevent the Monitor window from being refreshed. The button text changes to Resume. Click the **Resume** button to resume monitoring. Since every update to the monitor is a request, constantly updating the monitor can load the server unnecessarily.

- Click the **Reset Statistics** button to reset values to zero. This allows you to view an accurate representation of the activity occurring at the current time.
- To change the rate at which the monitor window is refreshed, change the time displayed and click the **Set New Rate** button.
- By default, statistics about Permanent Sessions, Transient Sessions, DBHosts and Requests are displayed. To remove one or more of these from the Monitor display, click the **View Control** button and clear the check box associated with the items to be removed.

- **View All** - Click this to see who is currently using the TrackRecord WebServer, regardless of the database they are accessing. Permanent and transient sessions for all databases are listed, as well as the time of the last request from each session to the WebServer.

Under Global WebServer Options, options are:

- **Enable** - Click this checkbox to have the Simple Search Field available.
- **Items to show in list items** - Enter the number of items to be displayed when a search is executed.
- **Enable** - Click this checkbox to include the number of items for each header in a report. If performance problems occur when pulling up the initial report, disable this feature.
- **Update WebServer Options** - Click this button to update the options.

WebMonitor

WebServer contains a process (TRWebSessionsHost) that maintains all currently active user sessions. This process contains a monitoring component called the Web Monitor. Assuming that the installer has set the appropriate DCO options (as described in the *Installation Guide*), this monitoring component is visible on the system on which the WebServer is installed. You cannot display it on all Administrator systems.

The Web Monitor is equivalent to the View All Sessions option on the WebServer Administration dialog, but allows two additional areas of functionality: it allows the administrator to view the internal database, and it contains a **Shutdown Now** button. This button shuts down the TRWebSessionsHost process, which will invalidate all user sessions.

Adding a List of Available Databases to the WebServer Login Window

You can add a drop-down list of available databases to the WebServer login window. To do so, follow these steps:

1. On the WebServer machine, open Windows Explorer and navigate to the directory `x:\Inetpub\cgi-win`, where x is the drive where Windows is installed.
2. Create a text file entitled `TRDatabaseList.txt` if one does not already exist.
3. In the `TRDatabaseList.txt` file, list the names of the databases available on your server. Enter one database name per line. For example, for databases Sample, Demo, and ProjectX, you would list:

   ```
   Sample
   Demo
   ProjectX
   ```

4. Save and close the file.
5. Log into the WebServer. In the **Database** field of the login page, verify that a drop-down list appears containing the databases you entered in step 3.

Troubleshooting Common WebServer Issues

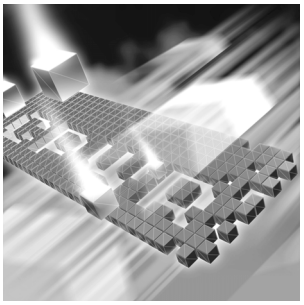
For a detailed listing of troubleshooting and configuration information for WebServer issues, refer to “Troubleshooting” in the TrackRecord online help.

For the most current TrackRecord support information, please visit the Compuware Web site at: http://www.compuware.com1From this location, you will have access to our online KnowledgeBase and the most current patches and updates for the product.

Appendix A

Using ActiveX

Reference material



ActiveX Interfaces

TrackRecord's ActiveX (formerly OLE) Automation interface allows any program with an ActiveX or OLE-compliant extension language to access a database, run queries, extract items, and perform other functions.

Using this information, you can write programs that can perform many useful functions, including:

- Reports and calculations based on live TrackRecord data without exporting
- Automatic exporting of daily data, and transfer of the exported data to an external site
- Parsing and submission, by importing, of user-submitted bugs or feature requests

You have virtually unlimited access to TrackRecord's types and items from ActiveX; through some clever programming, TrackRecord can be customized to do almost *anything*!

Basic Design

Through automation, TrackRecord exposes a number of objects that represent parts of its internal applications. Apart from the main application, each object has a singular form, representing a single instance of

the object, and a plural form, representing a collection of objects. Collections can be iterated over using standard automation methods.

Simple Examples

*Tip: For more examples, open Windows Explorer and navigate to **Program Files**>**Compuware**>Track Record>**Examples**.*

Visual Basic Example 1

Here's a simple Visual Basic program that retrieves the items that match a particular query, and displays the results:

Note: Since Automation controllers may use different extension languages, this example may not work exactly as presented here. Despite that, the general structure and usage of TrackRecord objects, properties, and methods remains the same.

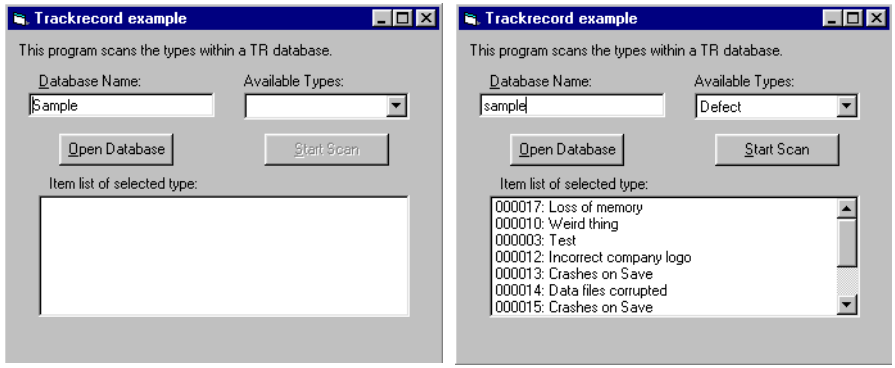
```
Dim TR As New TRDatabase.application
Dim MyQuery As TRDatabase.Query
Dim Results As TRDatabase.Items
Dim Item As TRDatabase.Item

Set MyQuery = TR.GetQuery("Tasks Due Today")
Set Results = MyQuery.Run

'If your controller has iteration support, you can use:

For Each Item In Results
          Print Item
Next Item

'Otherwise, you can explicitly iterate using:

NumItems = Results.Count

For i% = 0 To NumItems - 1
          Print Results.Item(i%)
Next i%
```

Visual Basic Example 2

A developer may wish to create a simple Visual Basic program to open a specific database and create a list of items for available TrackRecord types in the database. In this example, you might create a form with two text

boxes, a command button, and a combo box such as the one shown in Figure A-1. The code would be written as follows:

Figure A-1. Example of Program Created Using a Visual Basic Program



Note: Since Automation controllers may use different extension languages, this example may not work exactly as presented here. Despite that, the general structure and usage of TrackRecord objects, properties, and methods remains the same.

```
'A general declaration in the project

Option explicit
Dim trapp As TRDatabase.Application

_____________________________________________________

Private Sub cmdOpenDB_Click()

'Declare Variables
       Dim trdb As New TRDatabase.Database
       Dim trtypes As TRDatabase.Types
       Dim trtype As TRDatabase.Type

'Initialize
       Set trapp = Nothing
       Set trapp = trdb.OpenDatabase(txtDBName)
       Set trdb = Nothing

'open database
       If trapp Is Nothing Then
           txtDBName = txtDBName & "- Unable to open."
           txtDBName.SetLength = Len(txtDBName)
           txtDBName.SetFocus
           Exit Sub
       Else
```

```
            txtDBName = trapp.DatabaseDirectory
        End If

        'iterate through the types in database.
        cboType.Clear
        Set trtypes = trapp.GetTypes

        For Each trtype In trtypes
            'List types in combobox cboType
            cboType.AddItem trtype.Value
        Next trtype

        'Clean up workspace
            Set trtypes = Nothing
            Set trtype = Nothing

        cboType.SetFocus

End Sub
```

```
Private Sub cmdStartScan_Click()

        Dim trtype As TRDatabase.Type
        Dim tritems As TRDatabase.Items
        Dim tritem As TRDatabase.Item

        Set trtype = trapp.GetType(cboType.Text)
        Set tritems = trtype.Items

        List1.Clear
        For Each tritem In tritems
            List1.AddItem tritem
        Next tritem

        Set trtype = Nothing

End Sub
```

Transactions in COM Applications

To insure compatibility with COM applications created in previous versions of TrackRecord, TrackRecord 06.03.00 now automatically starts a transaction when interacting with the TrackRecord database and commits the transaction before exiting the API. Some examples of APIs that are wrapped in transactions are Application::ProcessItemchanges, Item::DeleteItem, User::ChangeEmail, and User::ChangePassword. To insure database integrity, it is recommended that COM API users updating the TrackRecord database use methods such as Application::GetEditItem, Database::startTransaction, Database::getEditLock, Database::abortTransaction, Database::commitTransaction, and so on to explicitly implement transactions.

Figure A-2 and Figure A-3 show the creation and modification of an item that comprises a transaction.

Figure A-2. Creating a New Item

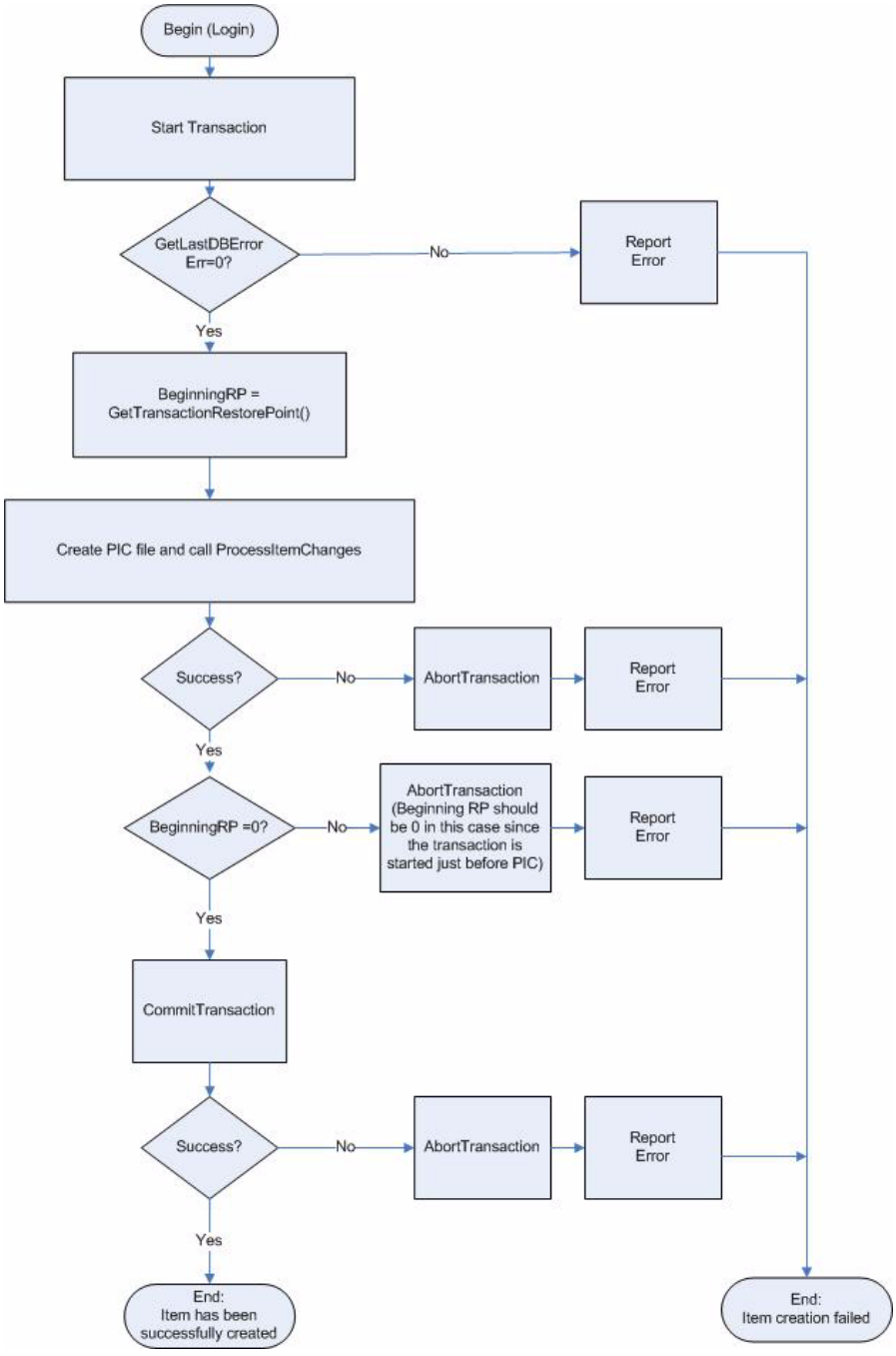
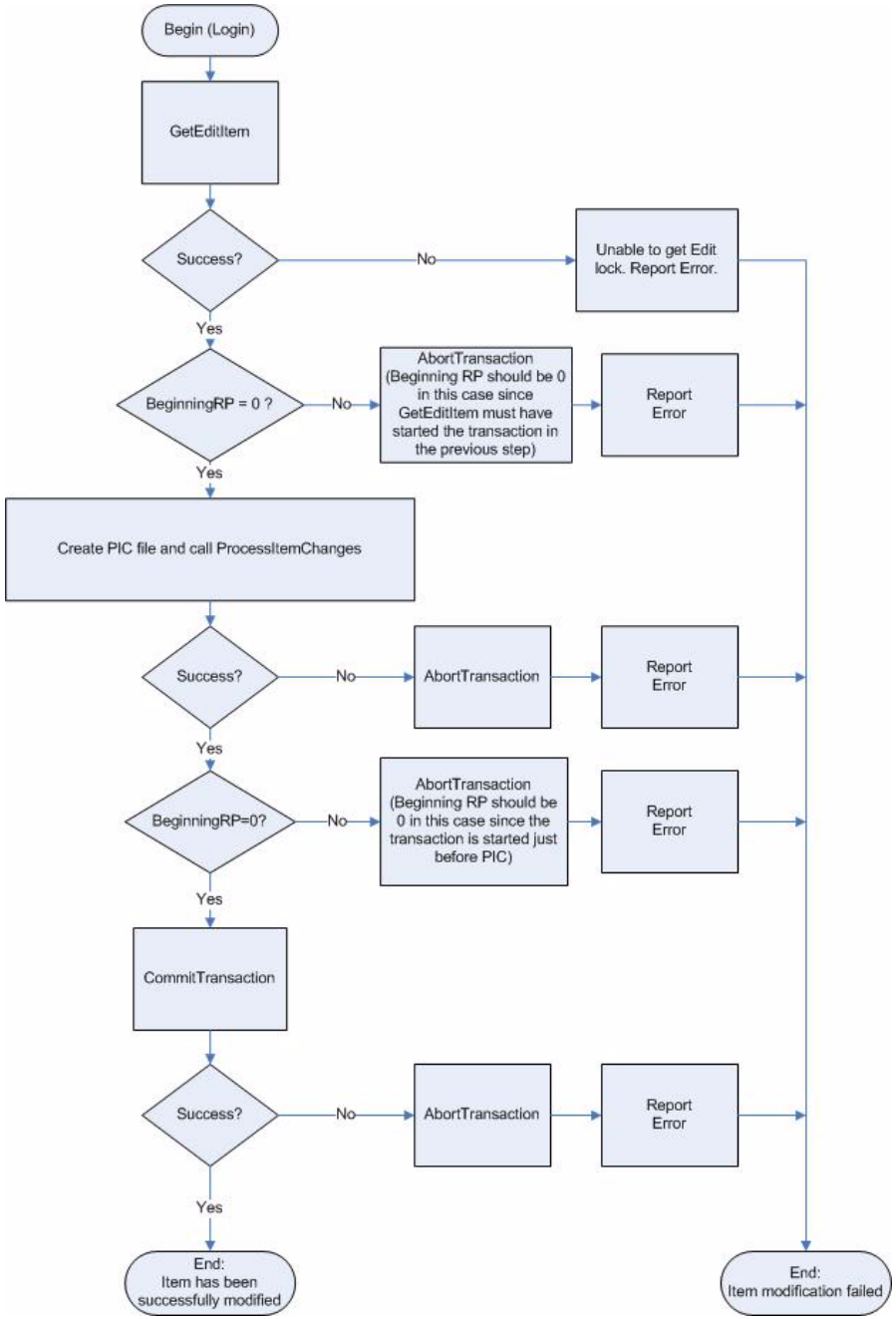


Figure A-3. Edit an Existing Item

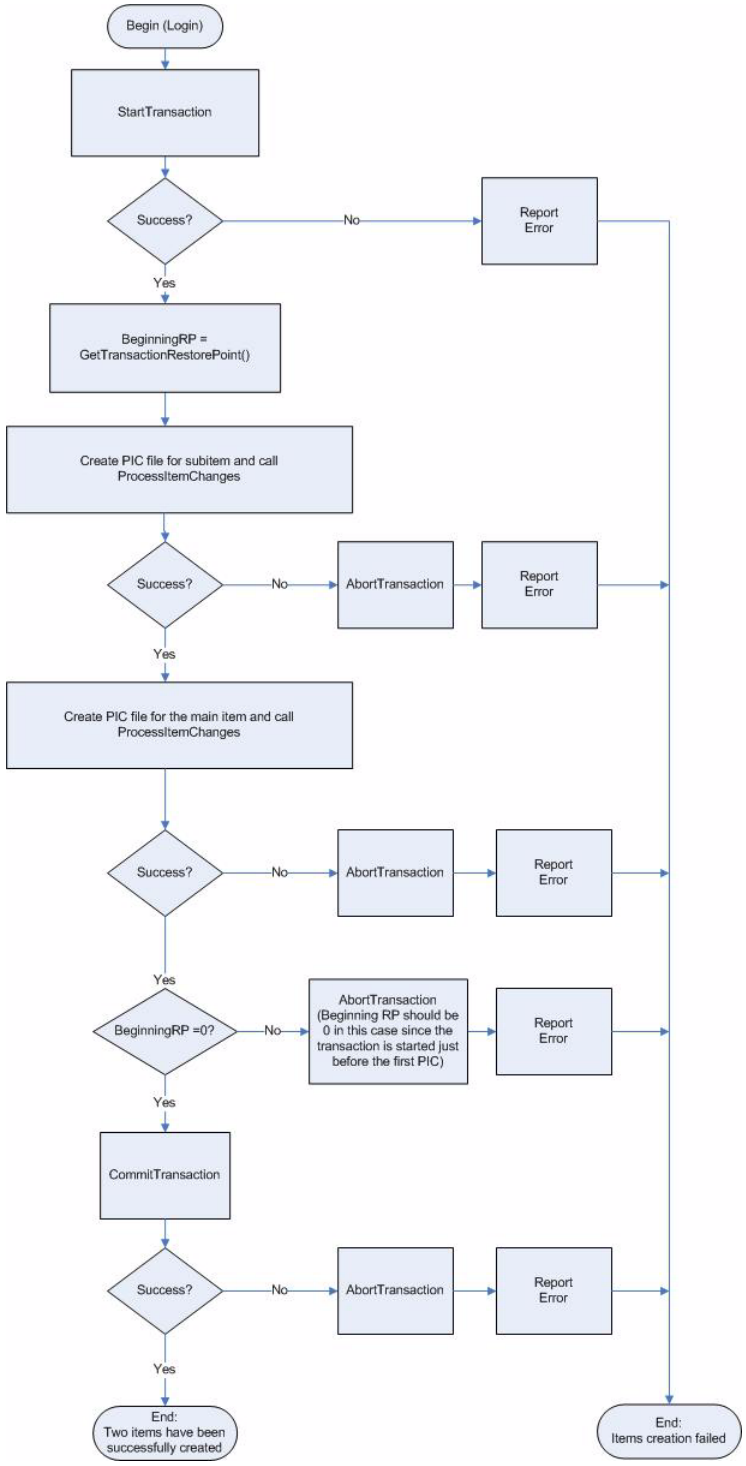


The flowcharts shown in Figure A-2 and Figure A-3 illustrate the creation and modification of an item that comprises a transaction. Simple flowcharts can be combined to create complex flowcharts, as in

Figure A-4. Checks for Restore Point may be unnecessary in a simple flow chart, but in actual usage, when a user might cancel part or all of a transaction, they became very important.

For example in Figure A-4 if a Main Item and SubItem were not created in the same transaction, an orphaned database item could occur if the user created a SubItem in one transaction, and then failed to create the Main Item.

Figure A-4. Creating a SubItem and Main Item



Object Reference

The following sections provide object properties and methods.

Application Object

The Application object is the basis for most of the fundamental operations that you will perform. Before doing anything, you must create an instance of the application object, using a line like:

```
Set TR = CreateObject ("TrackRecord.Application")
```

Once created, you can call any of the methods in the object to perform operations on your TrackRecord data.

Properties

Table F-2 Application Object Properties

CurrentUser	CurrentUser As Object
The user name of the currently logged in user.	
DatabaseDirectory	DatabaseDirectory As String
Current database directory.	
ServerName	ServerName As Boolean
Name of the server machine.	
TypesModifiedDate-andTime	TypesModifiedDateandTime As String
Read only. When the type definitions were last changed.	

Methods

Table F-3 Application Object Methods

CreateItems	Function CreateItems() As Items
Returns an empty Item collection.	
Export	Function Export (OutputFile As String, TemplateFile As String) As Items
Exports all items from the database. TemplateFile is optional; if not specified, all items are exported to a TrackRecord exchange file. If specified, all items matching the template specification are exported to OutputFile in delimited ASCII format. Returns an Items collection containing all the items that were exported.	
GetEditItem	GetEditItem(ItemIdentifier as String, ParentItemIdentifier String, mode as Integer) as EditItem

	Returns a single Item from the database, based on the Item's unique identifier. This method can be called to open an item in edit mode since calling this method starts a transaction. The returned EditItem object contains the item object, transaction restore point and error information if any error occurred. Parameters: • *ItemIdentifier* - The internal identifier of the item to be retrieved. This id string should be in hexadecimal format. • *ParentItemIdentifier* - The internal identifier of the parent item of the item to be retrieved. This id string should be in hexadecimal format. If the item to be opened does not have any parent, NULL or 0 can be passed. Default value: NULL • *Mode* - A bit mask specifying what edit lock to break. Default value - ASK_EDIT. The values are as follows: ◇ ASK_EDIT - (0x00) - Check if the item is currently locked (by any one). An error is set if the lock does not exist. ◇ ASK_USER_EDIT - (0x01) - Check if the item is currently locked by the current user. An error is set if the lock does not exist. ◇ GET_EDIT - (0x10) - Lock item if possible, but don't break existing edit lock. ◇ FORCE_EDIT - (0x11) -Lock item, and break existing edit lock if the logged in user has administrator privileges Returns: An EditItem object. Errors are set for the following cases: • If no items exists with the given id • ASK_EDIT or ASK_USER_EDIT modes: An error is set if the lock does not exist. • GET_EDIT or FORCE_EDIT: Error is set if the user does not have write permissions or if another user currently has an edit lock on the item or if the current user does not have edit lock on the parent item
GetItem	Function GetItem (ItemIdentifier As String) As Item

Table F-3 Application Object Methods

Returns a single Item from the database, based on the Item's unique identifier. This identifier can be passed to an external program via an Executable button. In the field properties of an executable button, specify the file name and enter **%i** in the **Parameter Control String** field. If the first three characters of an identifier match a valid database type, GetItem will return a new valid item.	
GetMaxUsers	Function GetMaxUsers() As Integer
Get the maximum number of users allowed.	
GetQueries	Function GetQueries (LocalQueriesOnly As Boolean) As Queries
Returns all of the Queries in the database, as a Query collection. LocalQueriesOnly is optional, and defaults to FALSE. If TRUE, all private queries for the currently logged in user are returned. These queries can also be retrieved from the User object; see "User Object" on page 162 for more information.	
GetQuery	Function GetQuery (QueryNameOrIdentifier As String) As Query
Returns a single Query from the database, based on its name or unique identifier. If a name is specified, and more than one query has the same name, the first one created is returned. See "Query Object" on page 158 for more information on Query objects.	
GetType	Function GetType (TypeName As String) As Type
Returns a single Type object, based on its name.	
GetUser	Function GetUser(UserName As String) As User
Retrieve a single user by name.	
GetUserByUserID	Function GetUserByUserID (UserID As Long) As User
Retrieve a single user by UserID.	
GetViewItem	GetViewItem(ItemIdentifier as String, reload as Boolean) as EditIt Returns a single Item from the database, based on the Item's unique identifier. This method can be called to open an item in view or read-only mode. If no item exists with the given id then an error will be set. The returned EditItem object contains the item object and error information if any error occurred. Parameters: • *ItemIdentifier* - The internal identifier of the item to be retrieved. This id string should be in hexadecimal format. • *reload* - The item is reloaded for the database if this is true. "reload" is optional and defaults to False. Returns: An EditItem object

Table F-3 Application Object Methods

GetTypes	Function GetTypes () As Types
Retrieves all of the Types in the database, as a Types collection.	
GetUsers	Function GetUsers () As Users
Returns a collection of User objects corresponding to the list of Users defined for this database. The User collection returned is “active”, and will update as users are added and deleted by the database Administrator.	
GetUserSetting	Function GetUserSetting(sectionName As String, variable-Name As String) As String
Get a preferences setting for the currently logged in user.	
Import	Import (InputFile As String, FavorDatabase As Boolean, TemplateFile As String) As Items
Imports items into the database from the InputFile. FavorDatabase is optional, and chooses how items are handled when they duplicate items already in the database. If TRUE (the default), the items in the database are retained if duplicates are imported from InputFile. TemplateFile is also optional; if specified, InputFile is assumed to be an ASCII delimited file. All the items that have been imported are returned, as an Item collection, when the import is complete. If the set is empty on return, there was an error.	
IsDBOpen	Function IsDBOpen () As Boolean
Is a database currently open?	
LogIn	Function LogIn([UserName As String],[Password As String],UserID As Long) As Boolean
Log in as a specific user.	
LogOut	Function LogOut() As Boolean
Logs out of TrackRecord, or rather, Logs in as the Admin user.	
ProcessItem-Changes	Function ProcessItemChanges(ScriptFile As Filename, [UserID As Long]) As Items
Process item change script. Item change scripts can be used to make changes to existing items, and to create new items. Returns the items that were changed. The format for the script is as follows: The first line must always be either [<identifier>] or [“<type name>”] where <identifier> is an identifier string for an existing item, or <type name> is the name of the item type that should be used for creating a new item. Each of the following lines is of the form: <field-name>\t<field-value> where <field-name> is the name of the field, and <field-value> is the value to set for the field. The \t represents a hard tab character in the file (ASCII 9), which can be entered with most text editors. Certain types of fields have additional modifiers other than the name of the field:	

Table F-3 Application Object Methods

- Name — For a field called **Name**, the options are **Name.First, Name.Last,** and **Name.Middle**.
- Address — For a fields called **Address**, the options are **Address.Address1, Address.Address2, Address.City, Address.State, Address.Country,** and **Address.Zip**
- Timespan — For a field called **Timespan,** the options are **Timespan.Start Time** and **Timespan.End Time,** which contain the combined date and time separated by a null character, and **StartDate, StartTime, EndDate,** and **EndTime** for date or time individually.
- Date/Time Options — For a field called **Date/Time**, the options are **Date/Time.Date** and **Date/Time.Time**
- Check boxes — For check boxes and radio buttons, the <field-value> should start with a space if the check box/radio button should be unchecked, and an 'X' if it should be checked.
- Lists — For a field called **Reported By**, the options are **Reported By.Add** and **Reported By.Delete**. For either one, the <field-value> should be the identifier of the item to add or delete.
- Compound type lists — For compound type fields, the <field-value> is the identifier of the item to select.

After all the fields are listed, the item is ended with a line:

[enditem]

Normally, items with an empty default abbreviation are not saved by ProcessItemChanges. However, if you want to force all items to be saved, whether they are complete or not, use the following line to end the item:

[enditem save]

To ensure that items are unique, end the item with:

[enditem unique]

Duplicates are determined by comparison of the Dupe field. The new or matching item will be returned.

Each item can be followed by the end of the file, or by another item.

A sample script might be:

["person"]

Name.Last Strange

Name.First Mickey
Company a000000

[enditem]

Table F-3 Application Object Methods

Notes:

1. Notepad field data requires double quotes.

2. Any quote character (“) within a string field requires a preceding backslash (\) character.

PurgeCache	Sub PurgeCache()
Purge the cache.	
Quit	Sub Quit()
Obsolete. Provided for backward compatibility.	
ValidateID	Function ValidID(ID As String) As Boolean
Returns TRUE if the identifier is valid. Returns FALSE if the item no longer exists in the database.	

Abbreviation Object

An Abbreviation represents a specific way of formatting the output of an Item. The Abbreviations in ActiveX are the same as the abbreviations used in ***regular*** TrackRecord.

Properties

Table F-4 Abbreviation Object Properties

Application	Application (Read only)
Application object.	
IsDefault	Boolean (Read only)
Is this the default abbreviation?	
IsDup	Boolean (Read only)
TRUE if this is the abbreviation used for duplicate elimination.	
IsLong	Boolean (Read only)
TRUE if this is the long default abbreviation.	
IsShort	Boolean (Read only)
TRUE if this is the short default abbreviation.	
Name	String (Read only)
The name of the abbreviation.	
Number	Integer (Read only)
The identifying abbreviation number: typically, you pass in the abbreviation number to other TrackRecord methods.	

Abbreviations Object (Collection)

A collection of Abbreviation objects, stored in a standard iterating form.

Methods

Table F-5 Abbreviations Object (Collection)

Count	Function Count() As Integer
Returns the number of Abbreviation objects in the collection.	
Item	Function Item (Index As Integer) As Abbreviation
Returns the Abbreviation at Index in the collection.	

Database Object

Permits the opening of a specified database.

Methods

Table F-6 Database Object Methods

abortTransaction	Function abortTransaction() As Boolean

Aborts the current database transaction. If no transaction is currently executing, the method does nothing. Any edit locks owned by this user will be released when this method is called.

Returns:

True - A transaction was aborted or no existing transaction existed.

False - The transaction abort failed.

It is recommended to call this method to abort a transaction in case of a failure to modify the database so it is brought to a new consistent state or in case the user cancels such update.

abortTransactionRestorePoint	Function abortTransactionRestorePoint (transactionPoint as Integer) as Boolean

Abort the transaction back to the restore point argument passed. The value of transactionPoint is a value returned from Database::startTransactionRestorePoint(). If a transaction restore point has not been started with Database::startTransactionRestorePoint() or the value for transactionPoint is out of range, an error value will be returned. All transaction restore points created after the passed transactionPoint will also be aborted.

Parameters:

transactionPoint - The transaction restore point to abort back to.

Returns:

True - A transaction was started.

False - The transaction start failed or an existing transaction was already in progress.

AutoLogIn	Function AutoLogin(UserName As String,Password As String,databaseDir As String, serverName As String, forceLogin As Boolean, appObject as Application) As Long

Automatically log in to TrackRecord using previously captured user information.

clearLastDBError	Function clearLastDBError() as Void

Clear the last database error. This method is provided to ensure the last DB error is cleared prior to calling another Database method.

closeDatabase	Function closeDatabase(forceClose Boolean) as Boolean

Table F-6 Database Object Methods

Closes the database and releases License. Various resources used by the database are also released.

Parameters:

- ◇ forceClose [NOT USED] - suppresses all prompts and chooses default answers, forcing the database to be closed with no user intervention. Default value of this parameter is false.

Returns 'true' if the database resources were released. Returns 'false' if an error occurred or the database was previously closed.

commitTransaction	Function commitTransaction () As Boolean

Commits the current database transaction. If no transaction is currently executing, the method does nothing and no error code will be returned. If there is an error committing the transaction an error will be returned.

Returns:

true - A transaction was committed or no existing transaction existed.

false- The transaction commit failed.

It is recommended to call this method to commit a transaction after calling a group of API methods that successfully modified the database, bringing it to a new consistent state.

GetAppFromHandle	Function GetAppFromHandle(sessionHandle As Long]As Application

Gets an application object from its session handle.

getDBErrorMessage	Function getDBErrorMessage (errorNumber as Long, idValue as Long) as String

Returns an error string for the errorNumber parameter passed. This is useful for displaying error messages to the user. If an errorNumber is not passed, then the last database error will be used.

Parameters:

errorNumber- The error number to return the error string for.

idValue - The item causing the error.

Returns: An error string for the passed error number.

getEditLock	Function getEditLock (index as Long, mode as Integer, user_id as Integer) as Boolean

Table F-6 Database Object Methods

	Attempts to get an edit lock for the item passed. If an edit lock can not be obtained (either the user does not have write permissions or another user currently has an edit lock) then an error will be returned. The Database::getLastDBError or Database::getDBErrorMessage can be used to obtain the actual error number or error message for the error. Parameters: *index* - The internal identifier of the item to obtain an edit lock. *mode* - A bit mask specifying what edit lock to break. Default value is GET_EDIT. The values are as follows: • ASK_EDIT (0x00) - Check if the item is currently locked. • ASK_USER_EDIT (0x01) - Check if the item is currently locked by the current in user. • GET_EDIT (0x10) - Lock item if possible, but don't break existing edit lock. • FORCE_EDIT(0x11) - Lock item, and break existing edit lock if the logged in user has administrator privileges. *user_id* - The user to check for lock. Default value is Current User. Returns: If mode is ASK_EDIT or ASK_USER_EDIT: • true if the item is locked • false if the lock does not exist. If mode is GET_EDIT or FORCE_EDIT: • true if we are able to obtain an edit lock on the item • false if an edit lock can not be obtained
getLastDBError	Function getLastDBError() as Long
	Returns the last database error that occurred. Many times an error occurs, but the user is not able to determine why the error occurred. This method can be called to get more information on the last database error that occurred. The return value from this method can be passed to Database::lastDBErrorMessage(long errorNumber) to obtain an error string. Returns: The last stored error number.
getTransactionRestorePoint	Function getTransactionRestorePoint() as Long
	Return the current database transaction restore point created by Database::startTransactionRestorePoint(). Returns: The current transactionRestorePoint.
OpenDatabase	Function OpenDatabase(Database As String, (Servername As String)) As Application
	Open a database and retrieve an application object.

Table F-6 Database Object Methods

releaseEditLock — Function OpenDatabase(Database As String, (Servername As String)) As Application

This will release the user's edit lock for the item passed. If an edit lock exists for the logged in user, the edit lock will be released.

Parameters:

index - The internal identifier of the item to release.

---

setSilentMode — Function setSilentMode(enable as Boolean) As Boolean

Calling this method with a boolean true value will supporess popping up of any dialog messages while running a TR COM application.

Parameters:

enable

- true will suppress the dialog messages
- false will not suppress any dialog messages

Returns:

- true if silent mode is set successfully
- false if unable to set the silent mode on the database

---

startTransaction — Function startTransaction() As Boolean

Start a new database transaction. If a current transactin is currently in progress, than an error will be returned.

Returns:

True - If a transaction was started, false if unable to start transaction or a transaction was already started.

It is recommended to cal this method to start a transaction before calling any of the following methods which modify the database:

- Application::ProcessItemChanges
- Application:GetItem
- FieldLayout::ValuesFromTemplates
- Item::DeleteItem
- User:ChangeEmail
- User::ChangePassword

Table F-6 Database Object Methods

startTransactionRestorePoint	Function startTransactionRestorePoint() as Long
Starts a transaction restore point and returns the transaction point number that was created. The value returned can be used with Database::abortTransactionRestorePoint (long transactionPoint). If a transaction is not currently executing, a new transaction will be created. If an error occurs creating the transaction restore point, an error value will be returned. Returns: The last created transaction restore point.	

EditItem Object

EditItem is like a wrapper on Item Object. It holds information about the Item, such as current Transaction restore point and user information. GetViewItem and GetEditItem API calls of the Application Object return this EditItem object.

Properties

Table F-7 EditObject Properties

Item	Item as Item
This represents the item itself along with its properties.	
TransactionRestore-Point	TransactionRestorePoint As Long
Contains the Transaction Restoration Point. If Application::GetEditItem is called, this property can be used to obtain the Transaction restoration point.	
LastDBError	LastDBError As Long
Contains the database error code if any error occurred during GetViewItem or GetEditItem calls.	
LastDBErrorMes-sage	LastDBErrorMessage as String
Returns 0, which indicates NO_ERROR occurred. If avalue other than 0 is returned, use LastDBErrorMessage to get the text associated with the error mes-sage.	
Contains the database error message if any error occurred during GetViewItem or GetEditItem calls.	

Field Object

Represents a single field of an item.Properties

Table F-8 Field Object Properties

AccessLevel	AccessLevel As Long (Read only)
Access control level.	
Application	Application (Read only)
Application object.	
DefaultValue	String (Read only)
Returns the name of the field.	
FieldValid	Boolean (Read only)
Returns TRUE if the current field object is valid.	
GetAbbreviations	Abbreviations
Retrieve the field abbreviation.	
IsGroupStart	Boolean
Is this the start of a radio button group?	
IsHidden	Boolean
Is this field hidden?	

Table F-8 Field Object Properties

IsList	Boolean (Read only)
Returns TRUE if the Field is a list, FALSE otherwise.	
IsMail	Boolean
Is this field marked for mail?	
IsReadOnly	Boolean
Is this field read-only?	
IsRequired	Boolean
Is this field required?	
ItemValid	Boolean (Read only)
Returns TRUE if there is an item associated with the current field, and that item is valid.	
LineAbove	String
Returns the line above a field.	
Modified	Boolean (Read only)
Returns TRUE if the field contents could have been modified since they were last examined. This property automatically resets if you retrieve Value or DefaultValue.	
Type	Type (Read only)
Returns a Type object that contains the type of the object stored in the field.	
Value	Items
Returns an Items collection containing the contents of the Field. If IsList is FALSE, there can only be one object in the collection. Otherwise, the collection will contain the contents of the list. Note that if the field is empty, the collection will be empty as well. Check the Count property of the Items collection before examining the collection's contents.	
Valid	Boolean (Read only)
Returns TRUE if the object is still valid. The object could become invalid if the item containing it was deleted while you still had an instance of the field.	

FieldLayout Object

The FieldLayout object is used to get back information about the layout of an item (fields and their ordering).

Properties

Table F-9 FieldLayout Object Properties

Application	Application (Read only)
Application object.	

Methods

Table F-10 FieldLayout Object Methods

Count	Function Count() as Long
Get the number of fields in the collection.	
Item	Function Item(Index As Long) As Field
Get specified field.	
GetFieldInfo	Function GetFieldInfo(Field As Field, Row As Long, HorizontalPosition As Long, Width As Long) As String
Returns information about screen layout for a field. The return value is the actual contents of the field.	
ValuesFromItem	Function ValuesFromItem(Item As Items) As Boolean
Sets source item(s) for field values. If this function is called, subsequent calls to GetFieldInfo will return field values for the specified item.	
ValuesFrom Templates	Function ValuesFromTemplates(Templates As Items) As Boolean
Sets source item(s) for field values. If this function is called, subsequent calls to GetFieldInfo will return values based on the set of templates passed in (combined with the default templates).	

Fields Object (Collection)

A collection of Field objects, stored in a standard iterating form.

Properties

Table F-11 Fields Object (Collection) Properties

Application	Application (Read only)
Application object.	

Methods

Table F-12 Fields Object (Collection) Methods

Count	Function Count() As Integer
Returns the number of Fields in the collection.	
Item	Function Item (Index As Integer) As Field
Returns the Field at Index in the collection.	
ItemByName	Function ItemByName (Name As String) As Field
Returns the Field in the collection named Name.	
ItemByTag	Function ItemByTag (TagToFind As String) As Field
Get specified field by tag.	

History Object

Represents a single entry in a Change History list.

Properties

Table F-13 History Object Properties

Action	String (Read only)
Contains a description of the what was done to the item. Possible values are: Changed Checked ItemCreated ItemDeleted ListItemAdded ListItemDeleted Unchecked	
Application	Application (Read only)
Application object.	
Field	Field (Read only)
Returns the Field that was modified.	
NewState	Item (Read only)
What is the new state?	
OldCheck	Boolean (Read only)
If the changed Field is a check box or radio button, and OldValueAvailable returns Check, OldCheck returns the previous state of the check box or radio button.	
OldDate	String (Read only)
If the changed Field is a Date field, and OldValueAvailable returns Date or DateAndTime, OldDate returns the date in the field before the user changed it.	
OldDateAndTime	String (Read only)
If the changed Field is a Time field, andOldValueAvailable returns DateAndTime, OldDateAndTime returns the date and time in the field before the user changed it.	
OldItem	Item (Read only)
If the changed Field is a reference to another item, and OldValueAvailable returns Item, OldItem returns the previous item in the field. For list actions, it returns the item added or deleted from the list.	
OldNumber	Long (Read only)
If the changed Field is a numeric field, and OldValueAvailable returns Number, OldDateAndTime returns the date and time in the field before the user changed it.	
OldState	Item (Read only)
What was the old state?	

Table F-13 History Object Properties

Type	Type (Read only)
Returns the type of the Field that was changed.	
User	String (Read only)
Returns the name of the user who modified the item.	
Value	String
Returns the value of the current history record.	

Methods

Table F-14 History Object Methods

DateAndTime	Function DateAndTime() As String
Returns the date and time the item was modified.	
OldValueAvailable	Function OldValuesAvailable() As String
Returns the type of "old" value available in this History object. Possible return values are: • Check • Date • DateAndTime • Item • Note • Number	

Histories Object (Collection)

A collection of History objects, stored in a standard iterating form.

Properties

Table F-15 Histories Object (Collection) Properties

Application	Application (Read only)
Application object.	
Value	string
Returns the value of all of the History information for the item, each entry on its own line, separated by "\n", as it would be displayed in the Change History column of the Outline view.	

Methods

Table F-16 Histories Object (Collection) Methods

Count	Function Count() As Integer
Returns the number of History objects in the collection.	
Item	Function Item (Index As Integer) As History
Returns the History object at Iqgh{ in the collection.	
Since	Function Since (DateAndTime As String) As Histories
Returns the History objects entered since the passed date and time.	

Item Object

An Item is a single object from the TrackRecord database.

Properties

Table F-17 Item Object Properties

Application	Application (Read only)
Application object.	
Default	Boolean
Contains TRUE if the item is a default template.	
DefaultValue	String (Read only)
Contains the default abbreviation for the Item.	
Fields	Fields (Read only)
Contains the collection of Fields available in this object.	
Identifier	String (Read only)
Contains the uniquely identifying string that represents this Item. This identifier can be saved and used later to retrieve the same object.	
IsReadOnly	Long 0=read/write, 1=read only, 2=read/add
Is this field read-only?	
Modified	Boolean (Read only)
Contains TRUE if the Item may have been modified since the last time you retrieved one of its properties.	
Private	Boolean
Contains whether or not the item is local to the current user. Can only be set if Template is TRUE.	
Template	Boolean
Contains TRUE if the item is a template.	
Type	Type (Read only)
Contains the Type object corresponding to the type of the current Item.	
Valid	Boolean (Read only)
Contains TRUE if the object is still valid. The object could become invalid if the item was deleted while you still had an instance of it.	

Methods

Table F-18 Item Object Methods

Cast	Function Cast (Type As Type) As Item
Attempts to cast this item to the passed Type. (For example, if there is an identity for this item of the specified type, it will return that identity.)	
Contents	Function Contents (Field As Field) As Items
Returns the contents of the specified Field.	
DeleteItem	Function DeleteItem() As Void
Deletes the specified item from the database. Does not apply access controls to the operation, so the item is always deleted, regardless of the privileges of the currently logged in user. This function can be very destructive, so be sure to carefully test any code that uses it.	
Downcast	Function Downcast() As Item
Casts the current Item to its "most derived" identity. For example, if you have retrieved a Task from the database, and the object also has a Bug Report identity, Downcast will return the Bug Report Item.	
Export	Function Export (OutputFile As String, TemplateFile As String) As Items
Exports the current Item to the specified OutputFile. TemplateFile is optional; if specified, the Item is written to a delimited ASCII file as specified in the template.	
ExtractAttachment	Function ExtractAttachment(ExtractToFile As String) As String
Extract attachment to a file.	
History	Function History (DateAndTime As String) As Histories
Returns the History objects associated with this Item. DateAndTime is optional: if not specified, all associated History items are retrieved. Otherwise, all History items since DateAndTime are retrieved.	
Identities	Function Identities () As Types
Returns the types of the other identities associated with this item, if any.	
Value	Function Value (Field As String, Abbreviation As Integer) As String
Returns the Abbreviation of the specified Field. Abbreviation is optional, and defaults to the default abbreviation (abbreviation 0).	
ValueByFieldAndAbbrev	Function ValueByFieldAndAbbrev (Field As Field, Abbreviation As Abbreviation) As String
Performs the same function as the previous command, using Field and Abbreviation objects instead of Strings.	

Items Object (Collection)

A collection of Item objects, stored in a standard iterating form. In addition to the typical methods, an Items collection can also have various set operations performed on it.

Properties

Table F-19 Items Object (Collection) Properties

Application	Application (Read only)
Application object.	
Modified	Boolean (Read only)
Are there any modified items in the collection?	

Methods

Table F-20 Items Object (Collection) Methods

Add	Function Add (ItemToAdd As Item) As Void
Adds ItemToAdd to the collection.	
AddIdentifier	Sub AddIdentifier (IdentifierOfItemToAdd As String)
Add an item to the collection by identifier.	
Clear	Function Clear () As Void
Empties the collection.	
Count	Function Count () As Integer
Returns the number of Items in the collection.	
Export	Function Export (OutputFile As String, TemplateFile As String) As Items
Exports all the Items in the collection to OutputFile. TemplateFile is optional; if specified, the Items are exported in delimited ASCII form, as specified in the template.	
GetModifiedItems	Function GetModifiedItems () As Items
Returns a collection containing all of the Items that may have been modified since the last time the function was called.	
Identifier	Function Identifier(Index As Long) As String
Get specified item identifier.	
Intersect	Function Intersect (ItemCollection As Items) As Items
Performs an intersection with the passed ItemCollection, returning the Items in common as the result.	
Item	Function Item (Index As Integer) As Item
Returns the Item at Index in the collection.	
Remainder	Function Remainder (ItemCollection As Items) As Items
Returns the collection Items that aren't in both the current Items collection and the passed ItemCollection.	
Remove	Function Remove (ItemToRemove As Item) As Void
Removes ItemToRemove from the collection, if there.	
Run	Function Run (QueryToRun As Query, DateSpecification As String,[Sort As Boolean]) As Item
Runs the passed Query on the Items in the collection, returning the result. DateSpecification is optional; if specified, the Query is run as a Calendar query, substituting the passed date as it would in the Calendar view.	

Table F-20 Items Object (Collection) Methods

Union	Function Union (ItemCollection As Items) As Items
Combines the items in the current collection with ItemCollection, returning the result.	

Query Object

A Query is a single query object from the database. It can be used to selectively extract Items based on their content.

Properties

Table F-21 Query Object Properties

Application	Application (Read only)
Application object.	
CalendarQuery	Boolean (Read only)
Returns TRUE if this Query can be used as a Calendar query.	
Identifier	String (Read only)
Contains the uniquely identifying string that represents this Query. This can be saved and used later to retrieve the same object.	
MailQuery	Boolean
Returns TRUE if the Query is a Mail query.	
Modified	Boolean (Read only)
Contains TRUE if the Query may have been modified since the last time you ran it.	
NeedsToRun	Boolean (Read only)
Contains TRUE if the database has changed in such a way that the results of the query might be different.	
Private	Boolean
Contains whether or not the item is local to the current user.	
Valid	Boolean (Read only)
Contains TRUE if the object is still valid. The object could become invalid if the item was deleted while you still had an instance of it.	
Value	String (Read only)
Contains the name of the Query.	

Methods

Table F-22 Query Object Methods

Export	Function Export (OutputFile As String, TemplateFile As String, DateSpecification As String) As Items
Exports the results of the Query to OutputFile. TemplateFile is optional; if specified, the Items are exported in delimited ASCII form, as specified in the template. If a DateSpecification is provided, the Query is run as a Calendar query, substituting the passed date as it would in the Calendar view.	
Run	Function Run (DateSpecification As String,[Sort As Boolean]) As Items
Runs the Query on the database, returning the result. DateSpecification is optional; if specified, the Query is run as a Calendar query, substituting the passed date as it would in the Calendar view.	

Queries Object (Collection)

A collection of Queries, stored in a standard iterating form.

Properties

Table F-23 Queries Object (Collection) Properties

Application	Application (Read only)
Application object.	

Methods

Table F-24 Queries Object (Collection) Methods

Add	Function Add (QueryToAdd As Query) As Void
Adds QueryToAdd to the collection.	
Clear	Function Clear () As Void
Empties the collection.	
Count	Function Count () As Integer
Returns the number of Queries in the collection.	
Item	Function Item (Index As Integer) As Query
Returns the Query at Index in the collection.	
Remove	Function Remove (QueryToRemove As Query) As Void
Removes QueryToRemove from the collection, if there.	

Type Object

Represents an item Type from the database. Every Item object has a Type, and a given Type can be derived from other Types, inheriting their properties, or have Types derived from it.

Properties

Table F-25 Type Object Properties

Application	Application (Read only)
Application object.	
Fields	Fields (Read only)
Retrieves the collection of Fields associated with this Type.	
GetAbbreviations	Abbreviations (Read only)
Returns the Abbreviations that can be used with this Type.	
GetInherited-Types	Types (Read only)
Contains the Types that this Type is derived from, if any.	
IsInShortList	Boolean (Read only)
Is this type in the short list?	
NewItems	Items (Read only)
Returns the collection of Items of the current Type that have been created since the Type object was retrieved, or since NewItems was last checked.	
Valid	Boolean (Read only)
Is this type a valid type?	
Value	String (Read only)
Returns the name of the Type.	

Methods

Table F-26 Type Object Methods

Field Layout	Function FieldLayout(LayoutType As Long) As FieldLayout
Retrieve FieldLayout objects for viewing this type. The LayoutType is reserved for future use; pass 3 for this value.	
GetDuplicates	Function GetDuplicates(dupeString As String) As Items
Returns the set of Items that have the specified Dupe abbreviation.	
GetItemInfo	Function GetItemInfo(LastID As String, NumItems As Long, StartWith As String) As String)
Retrieve next NumItems items of this type. The strings returned are in the form of the default abbreviation for the first item, followed by a newline character, followed by the identifier of the item, followed by another newline character, followed by the next abbreviation, and so on.	
Items	Function Items() As Items
Returns all the Items in the database of the current Type.	

Types Object (Collection)

A collection of Types, stored in a standard iterating form.

Properties

Table F-27 Types Object (Collection) Properties

Application	Application (Read only)
Application object.	

Methods

Table F-28 Types Object (Collection) Methods

Count	Function Count() As Integer
Returns the number of Types in the collection.	
Item	Function Item (Index As Integer) As Type
Returns the Type at Index in the collection.	

User Object

A User is an individual, created by the database administrator, who can log in to the database and perform operations. Each user has a number of properties associated with it that can be examined through this ActiveX object. In addition, the user's private Query and Template objects can be retrieved.

Properties

Table F-29 User Object Properties

Active	Boolean (Read only)
Returns TRUE if the administrator has activated this user, FALSE otherwise.	
Admin	Boolean (Read only)
Returns TRUE if the user is an administrative user and FALSE otherwise.	
Application	Application (Read only)
Application object.	
DefaultValue	String (Read only)
Returns the name of the User.	
MailAddress	String (Read only)
Contains the mail address defined for this User.	
MailEnabled	Boolean (Read only)
Returns TRUE if mail processing has been enabled for this user.	
Modified	Boolean (Read only)
The property is TRUE if this User may have been modified since it was retrieved.	
UserID	Long (Read only)
Returns an integer that can be passed to ProcessItemChanges to specify that the current user be referenced in any Change History entries that result from script processing.	
UserName	String (Read only)
Retrieves the name of the User.	
Valid	Boolean (Read only)
Returns TRUE if the User is still valid, FALSE otherwise. A User can become invalid if the administrator deletes it while an automation client still has an active reference to it.	

Methods

Table F-30 User Object Methods

GetQueries	Function GetQueries (IncludeNonMail As Boolean, IncludeMail As Boolean) As Queries
Retrieves all the private queries associated with this user. By providing different values for IncludeNonMail and IncludeMail, various combinations of mail and private queries can be returned. Note that the collection retrieved from GetQueries remains active. As private and/or mail queries are added/removed for this user, they're added to or removed from the collection.	
GetTemplates	Function GetTemplates () As Items
Retrieves all of the private templates associated with this user. Note that the collection retrieved from GetTemplates remains active. As templates are added/removed for this user, they're added to or removed from the collection.	

Users Object (Collection)

A collection of Users, stored in standard iterating form.

Properties

Table F-31 User Object (Collection) Properties

Added	Boolean (Read only)
Returns TRUE if there are new users in the collection. These new User objects can be retrieved with GetAddedUsers, documented below.	
Application	Application (Read only)
Application object.	
Modified	Boolean (Read only)
Returns TRUE if any users in the collection may have been modified. The potentially modified User objects can be retrieved with GetModifiedUsers, documented below.	

Methods

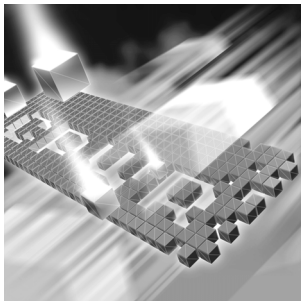
Table F-32 User Object (Collection) Methods

Count	Function Count () As Integer
Retrieves the number of users in the collection.	
GetAddedUsers	Function GetAddedUsers () As Users
Gets the collection of users that were added since the original Users collection was instantiated, or since this method was last called.	
GetModifiedUsers	Function GetModifiedUsers () As Users
Gets the collection of users that may have been modified since the original Users collection was instantiated, or since this method was last called.	
Item	Function Item (Index As Integer) As User
Returns the User at Index in the collection.	

Appendix B

Using Tags

Reference material



Overview of Tags

Many Compuware products use TrackRecord as an essential component. When these products integrate with TrackRecord, they depend on types, and fields within types, to properly integrate with the TrackRecord database. To provide for this dependency, TrackRecord supplies ActiveLink tags. *Tags* let administrators link the types and fields they create with the underlying structure. TrackRecord is shipped with a number of tags that allow integration. There are also some internal tags used to implement processes within TrackRecord itself, including workflow and share groups.

When to Use Tags

If you use the types shipped with the sample TrackRecord database, you can change type and label names without concern for ActiveLink tags. You only need to use the Tags dialog box when:

- Creating new types designed to exploit a Compuware technology, such as the build registration capability
- Changing previously tagged types or fields

- Deleting a field, which requires assigning its tag to another field

Caution: If you choose to modify types and fields, it is imperative that administrators verify that all tags are applied correctly to ensure integration between TrackRecord and DevPartner/QACenter/Fault Manager.

For example, an administrator deletes the Defect type and substitutes a new type called Bug Report. In order for the Bug Report type to serve the same function as the Defect type, it should be given the tag that by default belonged to the Defect type (QC_Defect). Any associated fields must also be properly tagged in order to work correctly with *QA*Center.

Cautions When Using Tags

If you do decide to change the tags, please note the following cautions:

- In order to maintain the integration of TrackRecord with *QA*Center and DevPartner, it is important that the tags are not changed unless it is absolutely necessary.
- TR_User is a special tag that identifies the type a user is associated with. In the default schema, this type is "Team Member". Do not create any types that inherit from the type with the TR_User tag associated with it. Doing so may cause problems with TrackRecord users.
- TrackRecord will not function properly if the TrackRecord type tags and their associated field tags do not exist or are not properly set up. For example, the Workflow Editor will not function properly if the three tags associated with the workflow: TR_Action, TR_State, and TR_Transition are not properly set up. To avoid this, ensure that the proper tags exist and are mapped to the correct types and fields.
- If you choose to modify types and fields, it is imperative that administrators verify that all tags are applied correctly to ensure integration between TrackRecord and DevPartner/*QA*Center/Fault Manager.

Categories of Tags

There are two categories of TrackRecord tags:

- **Type-level** — Tags associated with data types in the database. See "Type Tags" on page 169.
- **Field-level**— Tags associated with fields within types in the database. See "Field Tags" on page 171.

Any given type or field can be partnered with multiple tags. For example, the Project field of a Defect type has both the DC_Defect_Project and QA_Defect_Project tags associated with it.

Tag Naming Conventions

Type tags are written in the format: **XX_Description**. Where XX is the external product (or suite) that uses the tag. Some examples include: TR = TrackRecord, DC = DevPartner Controlled, QC = QACenter, and FM = Fault Manager. Description is usually the name of the type as it appears in your default schema. For example, Defect = Defect type and Hardware-Config = Hardware Configuration type.

Field tags are written in the format: **XX_TypeDesc_FieldDesc**. XX_TypeDesc represents the type tag name to which this field must belong. For example, DC_Defect_ prefixes all field tags within a DevPartner Reported Defect. FieldDesc is usually the name of the field as it appears in your default schema.

Type Tags

Tip: The first two letters of the tag name denote the external product (or suite) that uses the type: TR = TrackRecord, DC = DevPartner Controlled, QC = QACenter, and FM = Fault Manager.

TrackRecord information **types** define the structure of the individual pieces of information stored in a project database. These types are the heart of TrackRecord and make integration possible. Manipulating types provides control over the database structure, but can also result in integration problems. There are four type tags: TrackRecord, QACenter, Fault Manager, and DevPartner. The following tables list the tag name and the default type that is associated with.

Type Tags for TrackRecord

Table F-33 Type Tags for TrackRecord

Tag Name	Description	Default Affiliation
TR_Action	Action item	Action
TR_AttachedFile	Attached external file	Attached File
TR_Defect	Regular defect or bug items	Defect
TR_Person	Person items	Person
TR_Priority	Priority choice items	Priority

Table F-33 Type Tags for TrackRecord

TR_Requirement	Requirement item	Requirement
TR_State	Workflow state	Status
TR_Task	Regular task items	Task
TR_Transition	Transition information to next state in workflow	Workflow
TR_User	Login users	Team Member
TR_Workgroup	Workgroup for sharing view, templates, and queries	Share Group

Type Tags for QACenter

Note: For additional information on integrating QADirector with a customized TrackRecord database, see "Integrating a Customized TrackRecord Database With QADirector" on page 31.

Table F-34 Type Tags for QACenter

Tag Name	Description	Default Affiliation
QC_Defect	QACenter submitted defect	QACenter Reported Defect
QC_Product	QACenter product or test tool	Test Tool
QC_Test	QADirector test items	QADirector Test

Type Tags for DevPartner

Table F-35 Type Tags for DevPartner

Tag Name	Description	Default Affiliation
DC_Build	DevPartner registered build	Build
DC_Component	DevPartner registered application component	Application Component
DC_Defect	DevPartner submitted defect	DevPartner Reported Defect
DC_Executable	DevPartner registered executable file	Executable File
DC_FailSafe	FailSafe submitted incident	FailSafe Reported Incident

Table F-35 Type Tags for DevPartner

DC_HardwareConfig	DevPartner submitted hardware configuration	Hardware Configuration
DC_Incident	DevPartner submitted incident	Incident
DC_Milestone	Project milestone information	Milestone
DC_OS	Operating system	Operating System
DC_Person	DevPartner person	Person
DC_Product	DevPartner product	Product
DC_Project	Software project	Project
DC_Run	DevPartner submitted run	DevPartner Session
DC_Task	DevPartner submitted task	DevPartner Task
DC_TeamMember	DevPartner Team Member	Team Member

Type Tags for Fault Manager

Table F-36 Type Tags for Fault Manager

Tag Name	Description	Default Affiliation
FM_Defect	Fault Manager submitted defect	Fault Reported Defect
FM_Incident	Fault Manager submitted incident	Fault Reported Incident
FM_FaultType	Fault Manager type	Fault Type

Field Tags

Tip: The first two letters of the tag name denote the external product that uses the type: TR = TrackRecord, DC = DevPartner Controlled, QC = QACenter.

Each data type consists of **fields**. Fields hold specific facts about a type, such as a person's address or the name of a project that are connected to items in the TrackRecord database and other applications. Manipulating fields provides control over the database structure, but can also result in integration problems. The fields within a type also contain tags to make integration possible. There are three types of field tags: TrackRecord, DevPartner, and QACenter.

TrackRecord Field Tags

Table F-37 TrackRecord Field Tags

Tag Name	Description	Derived From	Data Type
TR_Action_Name	Name of action	TR_Action	String
TR_AttachedFile_Description	Description of file	DC_AttachedFile	Notepad
TR_AttachedFile_FileName	Name of attachment file	DC_AttachedFile	String
TR_Item_State	Status of an item		
TR_State_Name	Name of state	TR_State	String
TR_State_Transitions	Legal transitions to next state	TR_State	TR_Transition (List)
TR_Transition_Action	Action signaling transition	TR_Transition	TR_Action
TR_Transition_Description	Description of transition	TR_Transition	String
TR_Transition_NextState	State to change to on action	TR_Transition	TR_State

QACenter Field Tags

Table F-38 QACenter Field Tags

Tag Name	Description	Derived From	Data Type
DC_Defect_Status	Status of defect	DC_Defect	TR_State
QC_Defect_AssignedTo	Person defect is assigned to	QC_Defect	TR_User
QC_Defect_Build	Build where defect was found	QC_Defect	DC_Build
QC_Defect_Configuration	Configuration where defect was found	QC_Defect	DC_Hardware-Config

Table F-38 QACenter Field Tags

QC_Defect_DatabaseName	Name of *QA*Center database containing test	QC_Defect	String
QC_Defect_DateEntered	Date defect is entered	QC_Defect	String
QC_Defect_Description	Description of defect	QC_Defect	Date
QC_Defect_EnteredBy	Person entering defect	QC_Defect	TR_User
QC_Defect_ExecutionDateTime	Date and time of test execution	QC_Defect	Time
QC_Defect_ExecutionDetails	Detailed information about test run	QC_Defect	Notepad
QC_Defect_FailureReason	Reason for test failure	QC_Defect	String
QC_Defect_JobName	*QA*Director job associated with defect	QC_Defect	String
QC_Defect_OS	Operating system running when defect is submitted	QC_Defect	DC_OS
QC_Defect_Project	Project that defect occurred in	QC_Defect	DC_Project
QC_Defect_ReportedBy	Person(s) reporting defect	QC_Defect	TR_Person
QC_Defect_ScriptDescription	Description of *QA*Director Script	QC_Defect	String
QC_Defect_ScriptName	*QA*Director Script associated with defect	QC_Defect	String
QC_Defect_Status	Status of defect	QC_Defect	TR_State
QC_Defect_Synopsis	Defect summary	QC_Defect	String
QC_Defect_Test	*QA*Director test item associated with defect	QC_Defect	QC_Test
QC_Defect_TestTool	*QA*Center product that was running test	QC_Defect	QC_Product
QC_Product_Name	*QA*Center product name	QC_Product	String
QC_Product_Version	*QA*Center product version	QC_Product	String
QC_Test_Details	*QA*Director test details	QC_Test	Notepad

Table F-38 QACenter Field Tags

QC_Test_ID	Test ID for *QA*Director test	QC_Test	Numeric
QC_Test_ProcedureName	Procedure name for *QA*Director test	QC_Test	String
QC_Test_ProcedureSummary	Procedure summary for *QA*Director test	QC_Test	String
QC_Test_Purpose	*QA*Director test purpose	QC_Test	Notepad
QC_Test_Requirement	Requirement for *QA*Director test	QC_Test	String
QC_Test_SuiteName	Suite name for *QA*Director test	QC_Test	String
QC_Defect_Origination_Phase	Origination phase of software lifecycle	QC_Phase	Single Item Combo Box
QC_Defect Discovery_Phase	Discovery phase of software lifecycle	QC_Phase	Single Item Combo Box

DevPartner Field Tags

There are thirteen types of DevPartner field tags which are described in the next section.

Build Type Field Tags

Each build type is derived from DC_Build.

Table F-39 Build Type Field Tags

Tag Name	Description	Data Type
DC_Build_BuildNumber	Number of build	Numeric
DC_Build_Checksum	DevPartner checksum information from build	Attachment
DC_Build_Components	Components included in build	DC_Component (List)
DC_Build_DateTime	Date and time when build was created	Time

Table F-39 Build Type Field Tags

DC_Build_Name	Build name/description	String
DC_Build_ProjectVersion	Project version	String
DC_Build_Submitter	Person registering build	DC_Person

Component Type Field Tags

Each component type is derived from DC_Component.

Table F-40 Component Type Field Tags

Tag Name	Description	Data Type
DC_Component_DateTimeModified	Last modified date and time	Time
DC_Component_Name	Name of component	String
DC_Component_Size	Size (in bytes)	Numeric
DC_Component_Tag	Tag associated with component	String

Defect Type Field Tags

Each Defect type field tag is derived from DC_Defect.

Table F-41 Defect Type Field Tags

Tag Name	Description	Data Type
DC_Defect_AssignedTo	Person assigned to	DC_Person
DC_Defect_Build	Build where defect was found	DC_Build
DC_Defect_Components	Other system components excluding components from recognized builds	DC_Component
DC_Defect_Configuration	Configuration where defect was found	DC_HardwareConfig
DC_Defect_DateEntered	Date entered	Date
DC_Defect_Description	Description of defect	Notepad
DC_Defect_EnteredBy	Person entering defect	DC_Person
DC_Defect_Key	Key field for locating duplicate items	String

Table F-41 Defect Type Field Tags

DC_Defect_OS	Operating system running when DevPartner defect is submitted	DC_OS
DC_Defect_Product	DevPartner product submitting defect	DC_Product
DC_Defect_Project	Project that defect occurred in	DC_Project
DC_Defect_ReportedBy	Person reporting defect	DC_Person
DC_Defect_SessionFile		Attachment
DC_Defect_Stack	Stack traceback from DevPartner	Notepad
DC_Defect_Synopsis	Defect summary	String

Executable Type Field Tags

DC_Executable derives each of these types.

Table F-42 Executable Type Field Tags

Tag Name	Description	Data Type
DC_Executable_DateTimeLinked	Date and time file was linked	Time
DC_Executable_FileVersion	File version	String
DC_Executable_Instrumentation	Type of instrumentation in file	DC_Product (List)
DC_Executable_ProductVersion	Product version	String

Hardware Config Type Field Tags

All Hardware Config types are derived from DC_HardwareConfig.

Table F-43 Hardware Config Type Field Tags

Tag Name	Description	Data Type
DC_HardwareConfig_Country	Country setting for operating system	String
DC_HardwareConfig_CPU	CPU type	DC_CPU
DC_HardwareConfig_DBCSEnabled	Operating system is DBCS enabled	Check box
DC_HardwareConfig_Language	Language setting for operating system	String

Table F-43 Hardware Config Type Field Tags

DC_HardwareConfig_MachineName	Name of machine	String
DC_HardwareConfig_Memory	Amount of memory (in MB)	Numeric
DC_HardwareConfig_MidEastEnabled	Operating system is Middle Eastern enabled	Check box
DC_HardwareConfig_Monitors	Number of monitors	Numeric
DC_HardwareConfig_MouseButtons	Number of mouse buttons	Numeric
DC_HardwareConfig_MouseWheel	Indicates whether the mouse has a mouse wheel	Check box
DC_HardwareConfig_NumProcessors	Number of processors	Numeric
DC_HardwareConfig_OS	Operating system	DC_OS
DC_HardwareConfig_Palette	Display palette	Numeric
DC_HardwareConfig_Resolution	Display resolution	String

Milestone Type Field Tags

All Milestone type tags are derived from DC_Milestone.

Table F-44 Milestone Type Field Tags

Tag Name	Description	Data Type
DC_Milestone_Completed	Milestone has been completed	Check box
DC_Milestone_DateActual	Actual end date	Date
DC_Milestone_DateProjected	Projected end date	Date
DC_Milestone_DateStart	Starting date	Date
DC_Milestone_Description	Description of milestone	Notepad
DC_Milestone_ExcludeBuilds	Builds excluded from milestone	DC_Build (List)
DC_Milestone_Name	Name of milestone	String

Miscellaneous Type Field Tags

Table F-45 Miscellaneous Type Field Tags

Tag Name	Description	Derived From	Data Type
DC_Person_Name	Name of person	DC_Person	Name
TR_Item_State	State field in any type of item	Any	TR_State
TR_Workgroup_Members	Members of a sharing workgroup	TR_Workgroup	TR_User (List)

OS Type Field Tags

Each operating system type is derived from DC_OS.

Table F-46 OS Type Field Tags

Tag Name	Description	Data Type
DC_OS_AdditionalInfo	Additional operating system information	String
DC_OS_Build	Operating system build number	Numeric
DC_OS_Comments	Additional operating system comments	String
DC_OS_MajorVersion	Major operating system version	Numeric
DC_OS_MinorVersion	Minor operating system version	Numeric
DC_OS_Name	Operating system name	String

Product Type Field Tags

The Product types are derived from DC_Product.

Table F-47 Product Type Field Tags

Tag Name	Description	Data Type
DC_Product_Name	DevPartner product name	String
DC_Product_Version	DevPartner product version	String

Project Type Field Tags

The Project types are derived from DC_Project.

Table F-48 Project Type Field Tags

Tag Name	Description	Data Type
DC_Project_BuildFormatString	String used to control formatting of builds	Notepad
DC_Project_BuildNumber	Current build number counter	Numeric
DC_Project_Builds	Builds done for project	DC_Build
DC_Project_Description	Description of project	Notepad
DC_Project_Milestones	Milestones in project	DC_Milestone (List)
DC_Project_Name	Project name	String
DC_Project_TeamMembers	Team Members on project	DC_TeamMember (List)
DC_Project_VCSProject	Corresponding version control project	VCSProject
DC_Project_Version	Current version of project	String

Run Type Field Tags

Each Run type is derived from DC_Run.

Table F-49 Run Type Field Tags

Tag Name	Description	Data Type
DC_Run_Build	Build used for run	DC_Build
DC_Run_Comments	Additional user comments on run	Notepad
DC_Run_Components	Components loaded by run	DC_Component (List)
DC_Run_DateTime	Date and time of run	Time
DC_Run_HardwareConfig	Hardware configuration for run	DC_HardwareConfig
DC_Run_Product	DevPartner product used for run	DC_Product
DC_Run_Project	Project used for run	DC_Project

Table F-49 Run Type Field Tags

DC_Run_SessionFile	Session file associated with run	Attachment
DC_Run_User	User submitting run	DC_Person

Task Type Field Tags

All Task types are derived from DC_Task.

Table F-50 Task Type Field Tags

Tag Name	Description	Data Type
DC_Task_Build	Build running when task was submitted	DC_Build
DC_Task_Completed	Task has been completed	Check box
DC_Task_CompletedInBuilds	List of builds where task was completed	DC_Build
DC_Task_CompletedInRuns	List of runs where task was completed	DC_Run
DC_Task_DateCompleted	Date task was completed	Date
DC_Task_DateDue	Date due	Date
DC_Task_Description	Task description	Notepad
DC_Task_HardwareConfig	Hardware configuration when task was submitted	DC_Hardware Config
DC_Task_Identifier	Identifier of task item	String
DC_Task_Key	Key field for locating duplicate items	String
DC_Task_Priority	Priority	TR_Priority
DC_Task_Product	DevPartner product submitting task	DC_Product
DC_Task_Project	Project that task is logged against	DC_Project
DC_Task_RepeatEachBuild	Indicates whether tasks should be repeated for each build	Check box

Team Member Type Field Tags

Each Team Member type is derived from DC_TeamMember.

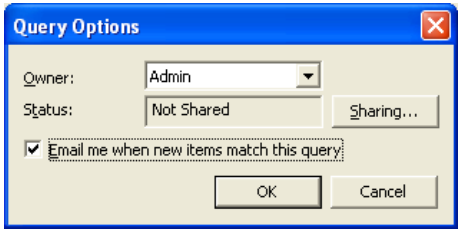
Table F-51 Team Member Type Field Tags

Tag Name	Description	Data Type
DC_TeamMember_Email	Email addresses	String
DC_TeamMember_Projects	Team Member projects	DC_Project (List)

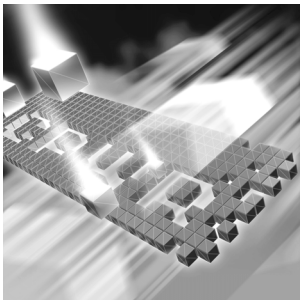
Troubleshooting Tags

For troubleshooting documentation, refer to “Troubleshooting” in the TrackRecord online help.

For the most current TrackRecord support information, please visit the Compuware Web site at: http://www.compuware.com. From this location, you will have access to our online Knowledgebase and the most current patches and updates for the product.


Query Options
Owner:
Admin
Status:
Not Shared
Sharing...
Email me when new items match this query
OK
Cancel

Glossary



Abend-AID Fault Manager: An automated fault management tool to help IT managers reduce downtime and optimize resources. Provides real-time and historical reports and graphs for monitoring the fault resolution process.

abbreviation: Text used to identify an item in the TrackRecord database.

absolute date: Entry in a date field that identifies a specific date; for example, May 19, 1999 or 2/14/87. Compare relative date.

access control: Function of an application that restricts user access to data according to permissions derived from the user's identity or membership in groups.

ActiveLink tags: A mechanism to allow administrators to link the types and fields they create with the underlying structure.

ActiveX control: An interface object on a form that enables or enhances a user's interaction with an application. ActiveX controls have an .ocx file name extension.

administrator: The person responsible for installing, configuring, and maintaining TrackRecord user accounts. The TrackRecord administrator may or may not be responsible for maintaining the network and database for a site.

API: An application program interface (API) is the specific method used by a computer operating system or by another application program by which a programmer writing an application program can make requests of the operating system or another application.

ASCII file: American Standard Code for Information Interchange (pronounced ASK-ee). Code in which numbers from 0 to 255 represent individual characters, such as letters, numbers, and punctuation marks. Standard for text representation.

audit trail: Record identifying each change made to an item or report and the identity of the person making the change. See change history.

AutoAlert: An optional component that provides email notification to users of the defects that pertain to them.

AutoAlert Administration Utility: A tool that allows you to add and remove databases that are being polled by the AutoAlert service, show users that have AutoAlert enabled, and set a variety of AutoAlert options.

available states: The Workflow Editor's list of states that have been defined, but are not yet used in the workflow.

bug report: See defect.

calendar: TrackRecord automatically displays a list of appointments and tasks.

CGI request: Common gateway interface (CGI) is a standard way for a Web server to pass a Web user's request to an application program and to receive data back to forward to the user. For example, when a user fills out a form on a Web page and sends it in, it usually needs to be processed by an application program. The Web server typically passes the form information to a small application program that processes the data and may send back a confirmation message. This method or convention for passing data back and forth between the server and the application is called the common gateway interface (CGI). It is part of the Web's Hypertext Transfer Protocol (HTTP).

change history: Record identifying each change made to an item or report and the identity of the person making the change. See audit trail.

child: An item that is subordinate to another item.

child header: A header that is subordinate to another header within an Outline Report. In the following example, Open Bugs is a child header of Bugs, and Priority 1 is a child header of Open Bugs. Compare parent header, sibling header.

```
Bugs
Open Bugs
Priority 1
Priority 2
Closed Bugs
```

ClientVantage: ClientVantage is designed to validate end-user response time from the end-user perspective and proactively measure application performance and availability to maintain high service levels.

cloning projects: Duplicating and reusing information from existing projects.

Command Line Interface: A CLI (command line interface) is a user interface to a computer's operating system or an application in which the user responds to a visual prompt by typing in a command on a specified line, receives a response back from the system, and then enters another command, and so forth.

coverage: Percentage of a project that has been tested, expressed as a percentage of lines tested or a percentage of the functions tested.

data type inheritance: TrackRecord uses a data hierarchy to develop related data types. When one data type (the child) is based on another data type (the parent), the child type inherits all the fields and controls of the parent type. See child and parent for more information.

database: A collection of data that is organized so that its contents can easily be accessed, managed, and updated. TrackRecord uses a relational database, a tabular database in which data is defined so that it can be reorganized and accessed in a number of different ways.

database lock-out: A error that will result if attempting to change the database (usually by creating or modifying types) while in use. A dialog will appear listing current users to allow the administrator to advise them to log off.

DBHost: Database Host. It caches values for a specific database.

defect: Report describing a problem found with software or hardware undergoing testing. A defect report is updated over time as a problem is reported, reproduced, fixed, and tested.

Defect: Item in the TrackRecord database describing a specific problem identified during project testing.

derived type: A data type that inherits all of the fields of its parent's type. For example, a Software User object inherits all the fields from the Person data type.

DevPartner Studio: TrackRecord is part of the Compuware DevPartner Studio suite of Windows-based, software debugging tools. Easily integrated into every team's development process, including Web-enabled, e-commerce, and distributed applications to help developers automatically detect, diagnose, and facilitate resolution of software errors; maximize code performance; and ensure optimum code coverage and testing.

export: To generate a formatted file containing data from the TrackRecord database for use by another application.

Fault Manager: See Abend-AID Fault Manager.

favorite reports: Frequently used reports or views.

fields: Holds specific facts about an item, such as a person's address or the name of a project.

filter: Process that selects information from the database according to user-specified criteria.

function coverage: A measure of how much of the code for a project has been tested, expressed as a percentage of functions tested across all Coverage Analysis sessions relative to the total number of functions in a project.

global favorite types: Favorite types that are shared with every TrackRecord user.

global Outline Report: Outline Reports that are shared with every TrackRecord user.

Global Preferences Administration: Creating queries, reports, and global templates, and identifying Home Pages and favorites for all users.

global query: Queries that are shared with every TrackRecord user.

global template: Templates that are shared with every TrackRecord user.

graph: A visual summary of information.

Group Administration: Creating groups and establishing group access rights.

header: Entry in an outline report that corresponds to a database query.

Header Wizard: TrackRecord utility that simplifies the process of defining and formatting outline reports. The Header Wizard was called the Header Engineer in previous TrackRecord releases.

hide: A field-level privilege setting that allows making a field invisible on a group-by-group basis.

Home Page: A report that opens automatically when TrackRecord is launched and can be accessed through the Home Page button.

import: To bring data generated by another application into the TrackRecord database.

inheritance: Property of data types that allow them to derive their structures from parent types.

item: Object or entity of a specific type in the TrackRecord database. Equivalent to a record in a relational database.

Item Browser: Provides access to information in the TrackRecord database and allows the creation of new items.

label: User-defined name you can attach to a specific software release or build in version control systems.

line coverage: A measure of how much of the code for a project has been tested, expressed as a percentage of code lines tested across all Coverage Analysis sessions relative to the total number of code lines in a project.

link: Relationship between items in the TrackRecord database. A link is created whenever one item references another. For example, attaching a note to a Person object creates a link between the two items.

mail query: A TrackRecord query in which an option is enabled to email users when new items match the query.

menu bar: Displays on the TrackRecord user interface below the window title and contains the following menus: File, Edit, Administrator, Favorites, Tools, Window, and Help.

merging: The process of compiling coverage information across multiple test sessions.

milestone: An event or point in a schedule that marks the transition of a project from one phase to another.

Milestone Status: Report that identifies the current status of a project with respect to a specified milestone, the volatility of a project from one build to another, data submitted from DevCenter or QACenter tools, and custom queries that summarize project information.

OLE-compliant: A set of APIs to create and display a document, OLE (Object Linking and Embedding) is Microsoft's framework for a compound document technology. Part of Microsoft's ActiveX technologies, OLE is part of a larger, more general concept, the Component Object Model.

Outline Report: Structured view of information in the TrackRecord database that is updated dynamically.

parent: An item that has one or more subordinate items.

parent header: A header that owns one or more subordinate headers within an outline report. In the following example, Bugs is the parent header of Open Bugs, and Open Bugs is the parent header of Priority 1 and Priority 2. Compare sibling header, child header.

```
Bugs
Open Bugs
Priority 1
Priority 2
Closed Bugs
```

password: Text string used to verify the identity of a user before granting the user read or write access to TrackRecord information.

priority: Arbitrary integer value assigned to a defect to indicate its significance relative to other defects. Depending on the standards of your site, you can use a higher or lower number to indicate that one defect is more important than another.

Project Administration: Determines the organizational framework for completing a deliverable including tasks required to build an application, people who work on those tasks, and dates on which the tasks must finish.

QADirector: QADirector is a powerful, extensible test management solution for full life cycle testing of distributed large-scale applications.

query: A means to specify criteria for data retrieval.

recurring events: Event scheduled to occur at regular intervals, such as Every Thursday.

relative date: Entry in a date field that identifies a date relative to the current system date; for example, Yesterday or Tomorrow. Compare absolute date.

Remainder Header: A None of the Above summary item used in an Outline Report. It is typically used to identify items belonging to a parent header that fail to match all criteria.

Rich Text Format (RTF): A file format that lets you exchange a text file between different word processors and different operating systems.

right-click: Pressing and releasing the right mouse button while positioning the cursor over an object or control on the screen. Typically opens a shortcut menu, which displays options applicable to the control or item over which the cursor is positioned.

share groups: Groups whose members share queries, Outline Reports, Milestone Status views, and templates.

shared: Property of a report that allows TrackRecord users other than the report's owner to view it. Only the owner of a report can modify a shared report, though other users can copy it (by saving it under a new name) and then modify their copy.

shortcut menu: Menu that appears when you right-click a control or item in a window.

sibling header: A header that is at the same level as another header within an outline report. In the following example, Open Bugs is a sibling header of Closed Bugs, and Priority 1 is a sibling header of Priority 2. Compare parent header, child header.

```
Bugs
Open Bugs
Priority 1
Priority 2
Closed Bugs
```

status bar: Located at the bottom left of the screen, this bar provides suggestions for the actions you can perform with a tool or a selected object.

subproject: A project that is one component of a larger or more complex project.

task: Any activity in a project that has a beginning date and an end date.

Task: An item in the TrackRecord database that identifies a task in a project.

Team Member: An individual assigned to work on a project.

template: A document or file having a preset format, used as a starting point for a particular application so that the format does not have to be recreated each time it is used.

TestPartner: An automated functional testing tool that has been specially designed for testing complex applications based on Microsoft, Java and Web based technologies.

toolbar: Contains shortcuts to the most commonly used tasks such as creating a new item or view, opening an existing item, displaying the Item Browser, opening the Home Page, or searching.

transient sessions: Efficient data structures that cache data from users for a period of ten minutes before it is removed.

type: An organizational system that provides the framework on which the TrackRecord database is built. Information is organized into categories such as Person, Company, Defect, or Project.

Type Editor: Inserts fields into a form, modifies fields on a form, and modifies settings.

type inheritance: See data type inheritance.

views: A specification for how data is displayed.

visible group layers: A list of all defined groups in the Workflow Editor.

volatility: A measure of the stability of code from one build to another, expressed as a percentage of the functions that changed between two builds.

WebMonitor: A monitoring component that maintains all currently active users sessions.

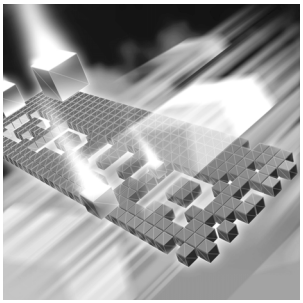
WebServer: An optional component that allows access to the TrackRecord database through a Web browser.

work area: Area where the workflow is created in the Workflow Editor.

workflow: A series of actions and the states resulting from these actions.

Workflow Editor: Creates and modifies the workflow and consists of these frames: available states, visible group layers, and work area.

Index



A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

Y