Rational® IBM Rational Robot

IBM

Version 7.0.0.1
Windows

Documentation Supplement

GI11-6387-00

Rational® IBM Rational Robot

IBM

Version 7.0.0.1
Windows

Documentation Supplement

GI11-6387-00

Note:
Before using this information and the product it supports, read the information under "Notices," on page 39.

2nd edition (December 2006)

This edition applies to **version 7.0.0.1 of IBM Rational Robot (product number 5724-G24)** and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this document

This documentation supplement describes IBM® Rational® Robot features that have been introduced in the past several service releases, from version 2003.06.12 to version 7.0.0.1. The information in this document supplements the information in the Help and the *IBM Rational Robot User's Guide*.

IBM Rational Robot 7.0.0.1 is integrated with ClearQuest Test Manager 7.0.0.1. For more information about working with ClearQuest Test Manager, refer to the ClearQuest Test Manager Help.

Who should read this document

The information in this document is intended for Rational Robot users.

Typographical conventions

This manual uses the following typographical conventions:

- *ccase–home–dir* represents the directory into which Rational ClearCase, Rational ClearCase LT, or Rational ClearCase MultiSite has been installed. By default, this directory is /opt/rational/clearcase on the UNIX system and Linux, and C:\Program Files\Rational\ClearCase on Windows.
- *cquest-home-dir* represents the directory into which Rational ClearQuest has been installed. By default, this directory is /opt/rational/clearquest on the UNIX system and Linux, and C:\Program Files\Rational\ClearQuest on Windows.
- **Bold** is used for names the user can enter; for example, command names and branch names.
- A sans-serif font is used for file names, directory names, and file extensions.
- **A serif bold font** is used for GUI elements; for example, menu names and names of check boxes.
- *Italic* is used for variables, document titles, glossary terms, and emphasis.
- `A monospaced font` is used for examples. Where user input needs to be distinguished from program output, **bold** is used for user input.
- Nonprinting characters appear as follows: <EOF>, <NL>.
- Key names and key combinations are capitalized and appear as follows: Shift, Ctrl+G.
- [ ] Brackets enclose optional items in format and syntax descriptions.
- { } Braces enclose a list from which you must choose an item in format and syntax descriptions.
- | A vertical bar separates items in a list of choices.
- ... In a syntax description, an ellipsis indicates you can repeat the preceding item or line one or more times. Otherwise, it can indicate omitted information.

  Note: In certain contexts, you can use "**...**" within a pathname as a wildcard, similar to "*" or "?". For more information, see the **wildcards_ccase** reference page.
- If a command or option name has a short form, a "slash" (/) character indicates the shortest legal abbreviation. For example:

  `lsc/heckout`

Contacting IBM Customer Support for Rational software products

If you have questions about installing, using, or maintaining this product, contact IBM Customer Support as follows:

The IBM software support Internet site provides you with self-help resources and electronic problem submission. The IBM Software Support Home page for Rational products can be found at http://www.ibm.com/software/rational/support/.

Voice Support is available to all current contract holders by dialing a telephone number in your country (where available). For specific country phone numbers, go to http://www.ibm.com/planetwide/.

Note: When you contact IBM Customer Support, please be prepared to supply the following information:

- Your name, company name, ICN number, telephone number, and e-mail address
- Your operating system, version number, and any service packs or patches you have applied
- Product name and release number
- Your PMR number (if you are following up on a previously reported problem)

Downloading the IBM Support Assistant

The IBM Support Assistant (ISA) is a locally installed serviceability workbench that makes it both easier and simpler to resolve software product problems. ISA is a free, standalone application that you download from IBM and install on any number of machines. It runs on AIX®, (RedHat Enterprise Linux® AS), HP-UX, Solaris, and Windows® platforms.

ISA includes these features:

- Federated search
- Data collection
- Problem submission
- Education roadmaps

For more information about ISA, including instructions for downloading and installing ISA and product plug-ins, go to the ISA Software Support page.

IBM Support Assistant: http://www.ibm.com/software/support/isa/

Chapter 1. Login

This chapter describes the login procedure of Rational Robot Standalone mode. Rational Robot is enhanced so that it can be used in a standalone mode, without any integration with Rational Test Manager or any other Rational Test Management solutions. Although the Rational Robot changed, Rational Test Manager functions and login procedure. For more information, see the Rational Test Manager Help.

Before you read this chapter, you should be familiar with the functions of Rational Test Manager and Rational Robot.

To log in to Robot Standalone mode, perform the following steps:

1. Start the IBM Rational Robot application by selecting **Programs** -> **IBM Rational** -> **IBM Rational Robot** or click**IBM Rational Robot** icon on your desktop.

   The **Select a Mode** dialog box appears.
2. Select the Robot Standalone mode from the **Mode** list. The **Select a Project** dialog box appears.

   Note:

   To set the default mode to Standalone Robot, select **Use this as default and do not ask again** check box.
3. Do one of the following in the **Select a Project** dialog box:
   - Select an existing Robot project from the **Project** list.
   - Select an unregistered project, by selecting **Browse** from the **Project** list and the required project from the **Select Robot Project** dialog box.
   - Manage projects by clicking **Manage**. For more information, see Chapter 2 Managing Projects.

Chapter 2. Managing Projects

This chapter describes how Robot Standalone projects are managed using the Robot Project Administrator. By using the Robot Project Administrator, you can do the following:

- Create projects
- Register and unregister projects
- Import assets from Rational Suite Projects to Robot project

You can delete the registered Robot projects from the location in which they are created. For more information, see Deleting a Project.

Using Robot Project Administrator

You use the Robot Project Administrator to manage projects. To use the Robot Project Administrator, perform one of the following steps:

- Log into Robot Standalone mode and click **Manage** in the **Select a Project** dialog box. The Robot Project Administrator window appears.

  OR
- From the Rational Robot menu, select **Tools -> Manage Robot Projects**. The Robot Project Administrator window appears.

The Robot Project Administrator window is composed of two parts, the Project pane and the Details pane. The Project pane displays a list of Rational Robot projects and the Details pane displays the details of an item such as a project that you selected in the Project pane.

Creating a project

To create a new project, perform the following steps:

1. Right-click on **Projects** and select **New Project**.

   The **New Projects - General** page appears.
2. Enter the project name and the location (in UNC format).

   Note: The following is an example of a possible path name (in UNC format): \\computer_name\directory\sub_directory, where the subdirectory would be an empty directory. If the directory is not empty, an error message is displayed asking you to specify an empty directory.
3. Select **Customize TestAssets Path** checkbox, to customize the TestAssets path.

   Note: By customizing the test assets path, you are storing the test assets in a path other than the default one.
4. Click **Browse** to select the TestAssets path.
5. Click **Next**.
6. Read the summary of the project settings provided in the **New Project - Summary** page.
7. Click **Finish** to accept the settings, or click **Back** to change the settings.

   The project that you created is displayed in the Projects list in the Robot Project Administrator window with the following information:
   - Location - The location of the Robot project.

- Path - The complete pathname of the Robot project.

Registering and unregistering a project

If you plan to administer a Rational Robot project on a computer other than the one on which it was created, register the project after creating and configuring it. You can also register any unregistered project. You register the Rational Robot project on the same computer you used to create and configure the project. To register a project, perform the following steps:

1. Right-click on **Projects** and select **Registering Existing Project**.
2. Select the required project and click **Open**.

   After a project is registered, the project name appears in the project hierarchy in the Rational Robot Administrator’s left pane.

Unregistering a project makes it disappear from the project hierarchy. To unregister a Robot project, perform the following steps:

1. Expand the **Projects** folder from the Robot Project Administrator's left pane. The registered projects are displayed.
2. Select a project to unregister, right-click , and select **Unregister Project**.

   The project is unregistered and removed from the project hierarchy in the Rational Robot Administrator’s left pane.

Importing assets from Rational Suite Projects

When you import assets from a Rational Suite Project to an existing or new Robot project, the GUI assets and datapools are imported into the Robot Project. To Import assets from a Rational Suite Project to an existing or a new Robot project, perform the following steps:

1. Right-click on **Projects** and select the option, **Importing Assets from Rational Suite Project**.
2. Select a Rational Suite Project and a Robot Project in the **Import Assets (Page 1 of 2)** page.
3. (Optional) Click **New** to create a new Robot project and then import test assets from a Rational Suite Project.
4. Select **Overwrite Assets** to overwrite the existing assets.
5. Click **Next** and read the instructions and project summary in the **Import Assets (Page of 2 of 2)** page.
6. Click **Finish** if you are satisfied with the settings or click **Back** to change the settings.

   The assets are imported into the Robot project.

Deleting a project

You can delete the registered Robot projects only from the location in which they are created. Once the project is deleted from its location, you can unregister it from the Robot Project Administrator or it will be removed from the projects hierarchy the next time you open the Robot Project Administrator window.

To delete a project, perform the following steps:

1. Go to the project folder and delete the entire project folder.
2. Unregister the project from Robot Project Administrator. For more information, see Registering and Unregistering a project.

Chapter 3. Managing test scripts and associated assets

This chapter describes how you can manage test scripts and associated assets in Robot Standalone mode. The test scripts and associated assets are now stored in Robot projects instead of in any test management solutions and can be managed in the following ways:

- Creating a GUI script, GUI shell script, Project Header file, or SQABasic file
- Editing a GUI script, Project Header file, or SQABasic file
- Deleting a GUI script

Creating a GUI script

In Robot Standalone, you can only create a GUI script but not a VU script. To create a GUI script, perform the following steps:

1. Select **File** -> **New** ->**Script** from the Rational Robot menu.
2. Type a name (40 characters maximum) and optionally, a description in the **New Script** dialog box.
3. Click the radio button **GUI**.
4. Click **OK**.

   The GUI Script window displays the script that you created. For more information on GUI scripts, see the Rational Robot Help.

Creating a GUI shell script

To create a GUI shell script, perform the following steps:

1. Select **File** -> **New** ->**GUI Shell Script** or **File -> Record GUI** from the Rational Robot menu. Or click CTRL+N or CTRL +R.
2. Type a name (40 characters maximum) and optionally, a description in the **New GUI Shell Script** dialog box.
3. Click **OK**.

   The GUI Shell Script window displays the shell script that you created. For more information on GUI shell script, see the Rational Robot Help.

Creating a Project Header file

To create a Project Header file, select **File** -> **New** ->**Project Header File** from the Rational Robot window.

The Project Header file window opens with a default name. You can specify the name of the file when you save it. The new project header file is saved with the file extension .sbh. For more information on project header files, see the Rational Robot Help.

Creating a SQA Basic file

To create a SQA Basic file, perform the following steps:

1. Select **File** -> **New**-> **SQA Basic File** from the Rational Robot menu.
2. Select the file type as Header File or Library Source File and click **OK**.

The SQA Basic File type window appears with the header or library source file. For more information on SQA file types, see the Rational Robot Help.

Editing a GUI script, Project Header file or SQABasic file

You can edit the existing GUI scripts, header files, or SQA file types.

To edit, select **File -> Open** and select GUI script, Project Header file, or SQA Basic file option.

The appropriate window is displayed to edit the script, header file, or SQA file type. For more information, see Rational Robot Help.

Deleting a GUI script

You can delete a GUI script from a project in Robot Standalone mode. To delete GUI script, perform the following steps:

1. Select **File -> Delete -> GUI Script** from the Rational Robot menu.
2. Select a script from the list in the **Delete Script** window.
3. Click **Delete**. Click **OK** to confirm the deletion.
4. Click **Close**.

Deleting a GUI script from the project also deletes its corresponding script file (.rec), executable file (.sbx), verification points, and low-level scripts.

Chapter 4. Managing datapools

This chapter describes how datapools are managed in Robot Standalone mode. Datapools let you automatically pump test data to virtual testers under high-volume conditions that potentially involve hundreds of virtual testers performing thousands of transactions.

Typically, you use a datapool so that:

- Each virtual tester that runs the script can send realistic data (which can include unique data) to the server.
- A single virtual tester that performs the same transaction multiple times can send realistic data to the server in each transaction.

To make the managing of datapools easier, Robot Standalone offers similar functionality as Test Manager for managing datapools. You can manage datapools by doing the following:

- Creating a datapool
- Editing datapools
- Exporting and importing datapools

Creating a datapool

To create and automatically populate a datapool, perform the following steps:

1. From the Robot Standalone menu, select **Tools > Manage Datapools**.
2. Click **New**.
3. Type a name for the datapool (40 characters maximum).
4. Click **OK**.
5. Click **Yes** to acknowledge that you want to define the datapool now.

   The **Data Type Specification** dialog box appears, in which you define the columns (that is, fields) in the datapool file. Datapool column definitions are listed as rows in this dialog box.
6. Click **Insert before** or **Insert after** to add a datapool column to the datapool.
7. Type a name for the new datapool column (40 characters maximum).
8. Assign a data type to the Type datapool column plus any other settings required for the column you are creating. For more information on datatypes, see About datatypes.
9. Repeat steps 6 through 8 until you have defined all the columns in the datapool.
10. Type a number in the **No. of records to generate** field. Alternatively, if you do not want to generate any data now, click **Save** to save your datapool column definitions, and then click **Close**.

    Note: If you are generating unique values for an Integers - Signed data type, Length, Minimum, Maximum, and No. of records to generate must be consistent. For example, if you want unique numbers from 0 through 999, errors may result if you set Length to 1, Maximum to 5000, and/or No. of records to generate to a number greater than 1000.
11. When you are finished defining datapool columns, click **Generate Data**.

Note: You cannot automatically generate data to a datapool that has more than 150 columns defined for it in the **Data Type Specification** dialog box.

12. Optionally, click **Yes** to see a brief summary of the generated data. If there are errors, an error message appears, you can correct the error in the Datapool Fields grid.

    Note: You cannot automatically generate data to a datapool that has more than 150 columns defined for it in the Datapool Specification dialog box.

To see the generated values, close the **Data Type Specification** dialog box. In the **Manage Datapools** dialog box, select the datapool you just created, click **Edit**, and then click **Edit Datapool Data**.

About datatypes

A datapool data type is a source of data for one datapool column. For example, the Names - First data type (shipped with Rational Test as a standard data type) contains a list of persons' first names. Suppose you assign this data type to the datapool column FNAME. When Robot automatically generates the datapool, it populates the FNAME column with all the values in the Names - First data type.

The two kinds of datapool data types are:

- Standard data types that are included with Rational Test. These data types include commonly used, realistic sets of data in categories such as first and last names, company names, cities, and numbers. For more information, see About Standard data types.
- User-defined data types that you create. In Robot Standalone mode, you cannot create user-defined data types however, you can create them in Test Manager. For more information, see the Test Manager Help.

About standard datatypes

Data types supply datapool columns with their values. You assign data types to datapool columns when you define the columns in the Datapool Specification dialog box. The standard data types listed in the following table are included with your Rational Robot software. Use these data types to help populate the datapools you create. The standard data types are listed in the Datapool Specification dialog box under the heading **Type**, and the other datapool column definitions (such as **Length** and **Interval**) referenced in the following table are some of the definitions you set in this dialog box.

Table 1.

Type	Description
Address – Street	Street numbers and names. No period after abbreviations.
Cities – U.S.	Names of U.S. Cities
Company Name	Company names (including designations such as Co and Inc where appropriate).
Date – Aug 10, 2006	Dates in the format shown. **Note:** To set a range of dates from January 1, 1900 through December 31, 2050, set Minimum to 01011900 and Maximum to 12312050. The day portion of the string is always two characters. Days 1 through 9 begin with a blank space. To include the comma (,) as an ordinary character rather than as the .csv file delimiter, the dates are enclosed in double quotes when stored in the datapool.

Table 1. (continued)

Type	**Description**
Date – August 10, 2006	Dates in the format shown. **Note:** To set a range of dates from January 1, 1900 through December 31, 2050, set Minimum to 01011900 and Maximum to 12312050. The day portion of the string is always two characters. Days 1 through 9 begin with a blank space. To include the comma (,) as an ordinary character rather than as the .csv file delimiter, the dates are enclosed in double quotes when stored in the datapool.
Date – MM/DD/YY	Dates in the format shown. **Note:** To set a range of dates from January 1, 1900 through December 31, 1999, set Minimum to 010100 and Maximum to 123199. You can only specify a range of dates in the same century (that is, the year in **Maximum** must be greater than the year in **Minimum**). To include the slashes (/) as ordinary characters rather than as the .csv file delimiter, the dates are enclosed in double quotes when stored in the datapool.
Date – MM/DD/YYYY	Dates in the format shown. **Note:** To set a range of dates from January 1, 1900 through December 31, 2050, set Minimum to 01011900 and Maximum to 12312050. To include the slashes (/) as ordinary characters rather than as the .csv file delimiter, the dates are enclosed in double quotes when stored in the datapool.
Date – MMDDYY	Dates in the format shown. **Note:** To set a range of dates from January 1, 1900 through December 31, 1999, set Minimum to 010100 and Maximum to 123199. You can only specify a range of dates in the same century (that is, the year in **Maximum** must be greater than the year in **Minimum**).
Date – MM-DD-YY	Dates in the format shown. **Note:** To set a range of dates from January 1, 1900 through December 31, 1999, set Minimum to 010100 and Maximum to 123199. You can only specify a range of dates in the same century (that is, the year in **Maximum** must be greater than the year in **Minimum**).
Date – MMDDYYYY	Dates in the format shown. **Note:** To set a range of dates from January 1, 1900 through December 31, 2050, set Minimum to 01011900 and Maximum to 12312050.
Date – MM-DD-YYYY	Dates in the format shown. **Note:** To set a range of dates from January 1, 1900 through December 31, 2050, set Minimum to 01011900 and Maximum to 12312050.
Date – YYYY/MM/DD	Dates in the format shown. **Note:** To set a range of dates from January 1, 1900 through December 31, 2050, set Minimum to 19000101 and Maximum to 20501231. To include the slashes (/) as ordinary characters rather than as the .csv file delimiter, the dates are enclosed in double quotes when stored in the datapool.
Date – YYYYMMDD	Dates in the format shown. **Note:** To set a range of dates from January 1, 1900 through December 31, 2050, set Minimum to 19000101 and Maximum to 20501231.
Date, Julian – DDDYY	Dates in the format shown. **Note:** To set a range of dates from January 1, 1900 through December 31, 1999, set Minimum to 00100 and Maximum to 36599. DDD is the total number of days that have passed in a year. For example, January 1 is 001, and February 1 is 032.

Table 1. (continued)

Type	**Description**
Date, Julian – DDDYYYY	Dates in the format shown. **Note:** To set a range of dates from January 1, 1900 through December 31, 1999, set Minimum to 00001 and Maximum to 99365. DDD is the total number of days that have passed in a year. For example, January 1 is 001, and February 1 is 032.
Date, Julian – YYDDD	Dates in the format shown. **Note:** To set a range of dates from January 1, 1900 through December 31, 2050, set Minimum to 1900001 and Maximum to 2050365. DDD is the total number of days that have passed in a year. For example, January 1 is 001, and February 1 is 032.
Date, Julian – YYYYDDD	Dates in the format shown. **Note:** To set a range of dates from January 1, 1900 through December 31, 2050, set Minimum to 1900001 and Maximum to 2050365. DDD is the total number of days that have passed in a year. For example, January 1 is 001, and February 1 is 032.
Float – X.XXX	Positive and negative decimal numbers in the format shown. Set Length to the number of decimal places to allow (up to 6). Set Minimum and Maximum to the range of numbers to generate. To generate numbers with more than 9 digits (the maximum allowed with the Integers - Signed data type), use the Float - X.XXX data type and set Decimals to 0. For example, credit card numbers typically have more than nine digits. To generate credit card numbers with the Float - X.XXX data type: 1. Set Decimals to 0. 2. Set Length to the number of digits in a credit card number. 3. Set Sequence to Random. 4. Set Repeat to 1 to ensure unique credit card numbers. 5. Set Minimum and Maximum to the smallest and largest possible credit card numbers.
Float – X.XXXE+NN	Positive and negative decimal numbers in the exponential notation format shown. Set **Length** to the number of decimal places to allow (up to 6). Set **Minimum** and **Maximum** to the range of numbers to generate.
Gender	Either M or F, with no following period.
Hexadecimal	Hexadecimal numbers.
Integers – Signed	Positive and negative whole numbers. This is the default data type. To include negative numbers in the list of generated values, set Minimum to the lowest negative number you want to allow. Maximum range: • Minimum = -999999999 (-999,999,999) • Maximum = 999999999 (999,999,999) For larger numbers, use a float data type. If you do not specify a range, the default range is 0 through 999,999,999. Use this data type to generate unique data in a datapool column (for example, when you need a ″key″ field of unique data). You can also use Read From File and user-defined data types to generate unique data.
Name – Middle	Masculine and feminine middle names. If the middle name is preceded by a field with masculine or feminine value (such as a masculine or feminine first name), the middle name is in the same gender category as the earlier field.

Table 1. (continued)

Type	Description
Name – Prefix (e.g., Mr)	Mr or Ms, with no following period. If the name prefix is preceded by a field with masculine or feminine value (such as a masculine or feminine gender designation), the name prefix is in the same gender category as the earlier field.
Names – First	Masculine and feminine first names. If the first name is preceded by a field with masculine or feminine value (such as a masculine or feminine name prefix), the first name is in the same gender category as the earlier field.
Names – Last	Surnames.
Name – Middle Initial	Middle initials only, with no following period.
Packed Decimal	A number where each digit is represented by four bits. Digits are non-printable. Note that commas and other characters that may be used to represent a packed decimal number may cause unpredictable results when the datapool file is read.
Phone – 10 Digit	Telephone area codes. To generate correct area code lengths, set Length to 3.
Phone – Area Code	Telephone area codes. To generate correct area code lengths, set Length to 3.
Phone – Exchange	Telephone exchanges. To generate correct exchange lengths, set Length to 3.
Phone – Suffix	Four-digit telephone numbers (telephone numbers without area code or exchange). To generate correct telephone number suffix lengths, set Length to 4.
Random Alphabetic String	Strings of random upper case and lower case letters and digits. Length determines the number of characters generated.
Random Alphanumeric String	Strings of random upper case and lower case letters and digits. Length determines the number of characters generated.
Read From File	Assigns values from an ASCII text file to the datapool column. For example, you could export a database column to a text file, and then use this data type to assign the values in the file to a datapool column. You can use this data type to generate unique data. You can also use the Integers - Signed and user-defined data types to generate unique data.
Space Character	An empty string.
State Abbrev – U.S.	Two-character state abbreviations.
String Constant	A constant with the value of Seed. The datapool column is filled with this one alphanumeric value.
Time – HH.MM.SS	Times in the format shown. Hours range from 00 (midnight) through 23 (11 pm). **Note:** To set a range of times from midnight to 2 pm, set Minimum to 0 and Maximum to 140000.
Time – HH:MM:SS	Times in the format shown. Hours range from 00 (midnight) through 23 (11 pm). To include the colons (:) as ordinary characters rather than as the .csv file delimiter, the dates are enclosed in double quotes when stored in the datapool. **Note:** To set a range of times from midnight to 2 pm, set Minimum to 0 and Maximum to 140000.

Table 1. (continued)

Type	Description
Time – HHMMSS	Times in the format shown. Hours range from 00 (midnight) through 23 (11 pm). **Note:** To set a range of times from midnight to 2 pm, set Minimum to 0 and Maximum to 140000.
Zip Code – 5 Digit	Five-digit U.S. postal zip codes. To generate the correct zip code lengths, set Length to 5.
Zip Code – 9 Digit	Nine-digit U.S. postal zip codes.
Zip Code – 9 Digit with Dash	Nine-digit U.S. postal zip codes with a dash between the fifth and sixth digits.
Zoned Decimal	Zoned decimal numbers.

Editing datapools

When you create a .csv file and import it as a datapool, Robot Standalone automatically assigns column names (that is, datapool field names) to each datapool column. You can edit the data in the columns of an existing or an imported datapool.

To edit datapools columns, perform the following steps:

Note: Datapool columns are also called fields.

1. From the Robot Standalone menu, select **Tools > Manage Datapools**.
2. Click the name of the datapool whose names you want to edit.
3. Click **Edit**.
4. Click **Define Datapool Fields**.
5. Modify the datapool column names. Datapool column names appear in the **Name** column of the grid.
6. When you finish editing datapool column names, click **Save**, and then click **Close**.

Importing datapools

Robot Standalone imports data from .csv files and automatically creates and populates datapools for you. You can import the data using the Robot Standalone Import feature.

To import a datapool perform the following steps:

1. From the Robot Standalone menu, select**Tools > Manage Datapools**.
2. Click **Import**.
3. In the **Look in** box, specify the directory where the datapool you created is located.
4. In the **File name** box, specify the name of the datapool .csv file.
5. Click **Open**. The **Import Datapool** dialog box appears, containing the datapool name.
6. Accept the default datapool name or type a new name (40 characters maximum).
7. In the **Field Separator** box, make sure the character(s) displayed are the same as the field separator used in the .csv file you are importing as a datapool.

8. Optionally, type a description of the datapool (255 characters maximum).
9. Click **OK**, and then click **Close**.

Exporting datapools

You can export a datapool to any directory on your computer's directory structure. When you export a datapool, the original datapool remains in its project.

To export a datapool:

1. From the Robot Standalone menu, select **Tools > Manage Datapools**.
2. Click the datapool to export.
3. Click **Export**.
4. In **Save In**, specify the directory where you want to copy the datapool.
5. Click **Save**.
6. Click **OK** to acknowledge that the datapool was exported to the correct location.

Chapter 5. Logs

In IBM Rational Robot Standalone mode, the logs are generated in .xml format. You can customize the appearance of the XML file by attaching a template.

When you play back a script the test results are generated as raw XML data and are published in a log file. By default, these log files are saved in <Project Name>\TestAssets\Log.

Chapter 6. Customizing IBM Rational Robot modes

IBM Rational Robot provides APIs to extend different Robot functions. With these APIs, you can create your own mode in IBM Rational Robot.

To create a new mode in IBM Rational Robot, perform the following:

- Create the AssetLibrary and a LogLibrary DLLs using the APIs
- Configure the AssetManager.xml file located in the IBM Rational Robot install directory to include the AssetLibrary and LogLibrary DLLs.

Note: You have an option to use the AssetLibrary provided by Robot and create your own LogLibrary to customize the log options.

Asset management

You can use the following APIs to manage assets in Standalone Robot:

Table 2.

Function	**Description**
Initialize()	Initializes the Asset Manager
LoginToProject()	Logs in to project
CreateScript()	Creates new script
GetScriptByName()	Gets script by name

Initialize()

This command initializes the Asset Manager.

Syntax

INT Initialize()

Return Values

- If the function succeeds, the return value is nonzero
- If the function fails, the return value is zero

LoginToProject()

This command encapsulates all the steps you need to perform to log in to a Robot project.

Syntax

*INT LoginToProject (INT paramCount, NamedValue *param)*

Table 3.

Parameters	**Description**
paramCount	Specifies the number of rows in the parameter array.
param	An array containing parameter name/value pairs, where *param[n].name* is the property name and *param [n].value* is its value.

Return Values

- If the function succeeds, the return value is nonzero
- If the function fails, the return value is zero

Remarks

NamedValue is defined as follows:

```
typedef struct
{
    char *Name;
    char *Value;
} NamedValue;
```

Example

```
LoginToProject(int propertyCount, NamedValue *property, PROJECTINFORMATION** pProjectInfo)
```

CreateScript()

This command creates a new script.

Syntax

*INT CreateScript(TCHAR *scriptName)*

Table 4.

Parameters	Description
scriptName	scriptName is a pointer to the string that specifies the script name

Return Values

- If the function succeeds, the return value is nonzero
- If the function fails, the return value is zero

Example

```
createScript(int iOption, SCRIPTPROPERTY *pScriptProperty)
```

GetScriptByName()

This command returns the location of the scripts in a project.

Syntax

*INT GetScriptPath(TCHAR * scriptPath)*

Table 5.

Parameters	Description
scriptPath	[out] scriptPath is a pointer to the string that returns the script path

Return Values

- If the function succeeds, the return value is nonzero
- If the function fails, the return value is zero

Example

```
getScriptByName(char * szScriptName, SCRIPTPROPERTY **pScriptProperty)
```

Logging

You can use the following APIs for logging test results in Standalone Robot:

Table 6.

Function	Description
SetLogParam()	Sets the log info from user
GenericLogEvent()	Publishes the event data
LaunchViewer()	Launches the log viewer

SetLogParam()

This API is called before the playback engine starts.

Syntax

*INT SetLogParam (TCHAR** logparams)*

Table 7.

Parameters	Description
logparams	Array of string values

Return Values

- If the function succeeds, the return value is nonzero
- If the function fails, the return value is zero

Remarks

If *logparams* does not contain all the necessary values then the API can open a dialog and collect the data form the user.

Example

```
setLogParam(NamedValue *property)
```

GenericLogEvent ()

This command publishes the event data.

Syntax

*INT GenericLogEvent(int eventCategory, int eventType, int result, int reason, tchar *description, int propertyCount, NamedValue*property);*

Table 8.

Parameters	Description
eventCategory	Category of events
eventType	Type of event
result	Result
reason	Reason
description	Description of the event
propertyCount	Number of properties
property	Array of property

Return Values

- If the function succeeds, the return value is nonzero
- If the function fails, the return value is zero

Example

```
genericLogEvent(s16 eventCategory, s16 eventType, s16 result, s16 reason, char *description, s32 prop
```

LaunchViewer()

This command launches the log viewer.

Syntax

*INT LaunchViewer(TCHAR * logfile)*

Table 9.

Parameters	Description
logfile	The name of the log file that needs to be opened in the viewer

Return Values

- If the function succeeds, the return value is nonzero
- If the function fails, the return value is zero

Example

```
launchViewer(char* pszDefaultLog)
```

Chapter 7. Configuring Robot with test management solutions

IBM Rational Robot Standalone provides COM Interface APIs for integration with any test management solution.

COM Interface APIs

You can use the following COM Interface APIs to perform tasks in IBM Rational Robot Standalone mode:

Table 10.

Function	**Description**
LoginToProject()	To log in to a Robot project
ExecuteScript()	To playback a recorded script
ExecuteScriptEx()	Extension of ExecuteScript()
Logoff()	To log off from the project
GetState()	To get the current state of Robot
AbortPlayback()	To abort playback
OpenScript()	To open a script from the project that you are logged in to

LoginToProject()

This command logs you in to a Robot project.

Syntax

HRESULT LoginToProject([in] BSTR sProjectPath, [in] BSTR sUserName, [in] BSTR sPassword, [in] BOOL bUseLoginService, [in, optional] SAFEARRAY NamedValue)

Table 11.

Parameters	**Description**
sprojectPath	The full path of the Robot project file
sUsername	The user name (if any) for authentication
sPassword	The password (if any) for authentication
bUseLoginService	
NamedValue	The extra login parameters

Return Values

- If the function succeeds, the return value is S_OK
- If the function fails, the return value is E_FAIL or failure HRESULT

ExecuteScript()

This command executes a recorded script from the project you are logged in to.

Syntax

HRESULT ExecuteScript(BSTR scriptName, [out,retval] long retVal)*

Table 12.

Parameters	Description
scriptName	The name of the script to be played back
retVal	Legacy param [not used]

Return Values

- If the function succeeds, the return value is S_OK
- If the function fails, the return value is E_FAIL or failure HRESULT

ExecuteScriptEx()

This command is an extension to ExecuteScript().

Syntax

HRESULT ExecuteScriptEx(BSTR scriptName, [in,optional] BSTR SMode)

Table 13.

Parameters	Description
scriptName	The name of the script to be played back

Return Values

- If the function succeeds, the return value is S_OK
- If the function fails, the return value is E_FAIL or failure HRESULT

Logoff()

This command closes Standalone Robot.

Syntax

HRESULT Logoff()

Return Values

- If the function succeeds, the return value is S_OK
- If the function fails, the return value is E_FAIL or failure HRESULT

GetState()

This command returns the current state of Standalone Robot.

Syntax

*HRESULT GetState([out,retval] int * nRetVal)*

Table 14.

Parameters	Description
nRetVal	The current state of Robot

Return Values

- If the function succeeds, the return value is S_OK
- If the function fails, the return value is E_FAIL or failure HRESULT

Remarks

Standalone Robot returns the following values:

Table 15.

STATE_IDLE	0x0000
STATE_PLAYBACK	0x0001
STATE_COMPILE	0x0004
STATE_ERROR	0x0008
STATE_OO_RECORD	0x0010
STATE_LL_RECORD	0x0020
STATE_VU_RECORD	0x0040
STATE_PAUSE_RECORD	0x0080

AbortPlayback()

This command stops the playback. If the playback option is set to close, then it will close Standalone Robot and logoff from the project.

Syntax

HRESULT AbortPlayback()

Return Values

- If the function succeeds, the return value is S_OK
- If the function fails, the return value is E_FAIL or failure HRESULT

OpenScript()

This command opens a recorded script from the project that you are logged in to.

Syntax

HRESULT OpenScript(BSTR scriptName)

Table 16.

Parameters	**Description**
scriptName	The name of the script to be opened

Return Values

- If the function succeeds, the return value is S_OK
- If the function fails, the return value is E_FAIL or failure HRESULT

Chapter 8. IBM Rational Robot 7.0 supplement documentation

This chapter describes IBM Rational Robot features that were introduced in the past several service releases, from version 2003.06.12 to version 7.0. The information in this chapter supplements the information in the Help and the IBM Rational Robot User's Guide.

General and functional testing

This section describes changes, enhancements to Rational Robot from version 2003.06.12 through version 7.0

Changes and enhancements

Alphanumeric verification points

The use of user-defined types for alphanumeric verification points (VPs) is now supported as described in the *IBM Rational Robot User's Guide*.

To use alphanumeric VPs:

1. Select an object to capture.
2. Click **Display GUI Insert Toolbar** on the GUI Record Toolbar (if recording) or on the standard toolbar (if editing).
3. Click **Alphanumeric Verification Point**.
4. In the Alphanumeric Verification window, select **Apply a User-Defined DLL Test Function**.
5. Click **OK**.
6. In the DLL Function Call window, enter values for a **Library Filename** and **Function**.
7. Click **OK**.

Icon and splash screen

There is a new product icon and splash screen for this release of Rational Robot.

Password encryption support

The password encryption API feature was previously undocumented.

InputEncKeys

Decrypts the input string and sends one or more keystrokes to the active window as if they had been entered at the keyboard. If the control has the password style set, Rational Robot encrypts the keystrokes and records the string. If the password style is not set, InputKeys is used. This feature works only with HTML text and standard edit controls.

Syntax: InputEncKeys *encryptedKeyText$*

encryptedKeyText$ is the encrypted form of the string that would be decrypted before sending it to the control.

Creating test datastores

There is a new wizard that streamlines the process of creating test datastores. You can also use it to add test assets, test users, and groups from an existing project to the datastore that you are creating. For more information, see the test datastore help in the Rational Administrator.

Validation and repair of test datastores

The Datastore Doctor application verifies the current status of your test datastores, and if necessary, repairs them. To access the Datastore Doctor, in Rational Administrator, click **Tools > Rational Test > Rational Datastore Doctor**.

SiteCheck and TestFactory

SiteCheck® and TestFactory® are no longer integrated with Rational Robot and are not supported at any level. References to them in product documentation and the Help should be ignored.

Mouse pause support

There is support for recording a mouse pause at a point.

API: `returnCode = MousePause RecString, ActionParams, milliSecondsToWait`

Description: Rational Robot does not record mouse movements. It records only mouse actions, such as a click, and keyboard actions. This is a common problem for AUT testing, which is sensitive to mouse hovers. This affects HTML pages with popup menus and certain hover help controls. Rational Robot fails to record these accurately.

With the provided fix, a mouse pause (configurable in milliseconds) over an object will be recorded by Rational Robot using the MousePause API. This API allows mouse positions to be captured. During playback, Rational Robot will pause over the object for the recorded time.

Parameters:

RecString
Object recognition string.

ActionParams
Object specific action items, such as coordinates.

milliSecondsToWait
Time in milliseconds that Rational Robot would pause over the object.

Note: returnCode is standard values returned by other APIs in Rational Robot.

Configuration: You can configure the minimum pause period that Rational Robot waits to detect mouse pauses. To do this, click **Tools > GUI record options > General** in the Rational Robot user interface. The GUI Record Options window opens, displaying the General page.

In the window, make sure that the **Enable Mouse Pause** check box is selected. In the **Pause Limit** field, type the time in milliseconds after which a mouse pause should be recorded.

You can also control recording using a Rational Robot hot key. The default hot key to enable or disable mouse pause recording is Ctrl + Shift + P. You can modify the hot key in the Robot Window page of the GUI Record Options window.

Because mouse pause recording can be slow at times, depending on the kind of AUT and the extensions enabled, this feature is designed so you can enable and disable it during the recording process.

Sample: The following script represents what might be recorded against popup menu items on a website after enabling mouse pause. Use the GUI Record Options window to enable or disable mouse pause support.

```
Sub Main
     Dim Result As Integer

     Window SetContext, "Caption=IBM Rational software -
     Microsoft Internet Explorer", ""
     MousePause "Type=HTMLLink;HTMLText=Products", "", 1172
     MousePause "Type=HTMLLink;HTMLText=Services", "", 4146
     MousePause "Type=HTMLLink;HTMLText=Support", "", 3596
     MousePause "Type=HTMLLink;HTMLText=Shop", "", 1162
     MousePause "Type=HTMLLink;HTMLText=Events", "", 3405

End Sub
```

After you apply the fix and enable mouse pause, playing the script will position the mouse over the specified menu items without clicking any of the items.

ScrollIntoView

There is support for an API to force an HTML element to scroll into view.

Syntax: `SQAScrollIntoView (recMethod$,viewCode&)`

Note: `viewCode` should be set to 5.

The CompileAll command-line option

Rational Robot has a new command-line option, `CompileAll`, to compile all the scripts and SQABasic library source files in a project.

Syntax: `rtrobo.exe [/user <userid>] [/password <password>] [/project <full path and full projectname>] /compileall`

Related options:

/user
User name for login.

/password
Optional password for login. Do not specify this option if there is no password.

/project
Name of the project that contains the script referenced, preceded by its full path.

Product support

This section describes the vendor software that Rational Robot supports

Oracle 9i and 11i

This version of Rational Robot supports testing Oracle 9i and 11i applications that were developed with Forms Developer, with the following restrictions:

- Testing Oracle ERP and CRM applications released with Oracle 11i is not supported.
- This version of Rational Robot does not have object-level support for tab control. Support is coordinate-based for this control.

To test Oracle Forms 9i and 11i applications:

1. If the Oracle container for Java™ service is not already running, start it by clicking **Start > Programs > Oracle 9i Developer Suite > Forms Builder > Start OC4J**.
2. For Rational Robot to detect Oracle 9i or 11i objects, you must install the Java Enabler. Make sure the Java Enabler is installed by performing these tasks:
   - To install the Java Enabler, click **Start > Programs > IBM Rational > Rational Test**.
   - Review the information about the Java Enabler in Chapter 15 of the *IBM Rational Robot User's Guide*.

To enable the client test machine:

1. Verify that the HTML and Java extensions are loaded in Rational Robot. For information, see Chapters 14 and 15 of the *IBM Rational Robot User's Guide* .
2. Install the Java Runtime Environment (JRE) on the client machine.

   Note: Oracle documentation does not specify any JRE requirements; however, Rational Robot support works best with Sun JRE 1.3.
3. Before starting Rational Robot for testing, make sure that the Java Enabler is installed and enabled. To install the Java Enabler, click **Start > Programs > IBM Rational > Rational Test**.

Internet Explorer 6.0

Rational Robot supports recording and playing back scripts on Internet Explorer 6.0.

Terminal Server

For functional testing, Rational Robot supports the following Terminal Server environments:

- Citrix MetaFrame (WIN2K)/Citrix MetaFrame client
- Microsoft® Terminal (WIN2K)/Microsoft Terminal Server Client
- Windows 2000 Server
- Windows Terminal Server (Windows NT® 4)

Both the application-under-test and Rational Robot are installed on the server. From within the client session you can start both the application-under-test and Rational Robot to run on the server. Rational TestManager Log Viewer edition can be run from the server or from within the client session.

Note:

- Rational TestManager does not support Citrix or Microsoft Windows Terminal Server.
- Rational Robot requires floating licenses for terminal servers.
- Make sure that the screen resolution for the client matches the screen resolution on the server.

Logging control: The main toolbar now has a **Log** button to turn logging results on and off. It displays automatically. If you are upgrading from an earlier release, you can add the button to the main toolbar by using the following procedure:

1. Click **View > Toolbars > Customize**, or right-click the toolbar and click **Customize**.
2. Click the **Commands** tab.
3. In the **File** category, drag the **Log** button to the main toolbar. Make sure that you release the mouse button within the toolbar.
4. Click **OK** to close the Customize window.

VS.NET support

Rational Robot provides complete support for VS.NET Windows Forms controls. Rational Robot has been enhanced to recognize these user interface controls at the object level. Rational Robot supports the following features:

- VS.NET 7.1
- Recording and playing back actions on all standard Windows Forms controls included with Visual Studio.NET
- VS.NET 1.0 SP2
- VS.NET applications with multiple domains
- Performing verification based on object data, object properties, the application-under-test menu, and alphanumeric data
- Integration with IBM Rational PurifyPlus™ for Windows
- Managed code in unmanaged space (for example, a C# user control embedded in a VisualBasic form or an HTML page)
- Unmanaged code in a managed application (for example, a VisualBasic control in a C# application)

Note: The VS.NET Enabler that is available for Rational Testing Products version 2002.05.00 is not required for version 2003.06.00 and later. For more information, see the Rational Robot Help and the *IBM Rational Robot User's Guide*.

Cross-browser support

Cross-browser support is improved in this release. In prior releases, by default, you could test every property of the browser-under-test. This caused problems because the default set of properties was different not only between Internet Explorer and Netscape, but also between one version of Internet Explorer and another. To resolve this issue, default testing covers a smaller subset of properties that are common to all browsers. This improves cross-browser support. You can add back any properties that are no longer part of the default, but the resulting script might not be compatible across different browsers. This change has no effect on older scripts. Previously recorded scripts will play back with whatever properties they contained when they were recorded.

Object Data tests by contents can fail cross-browser testing because IE and Netscape add extra line feeds and the contents might not be compared properly. To resolve this issue, use the **Filtered Contents** option. This strips out any line feeds and extra white spaces.

The default width and height properties for any object that contains them might not be compatible across browsers. Netscape returns a value for height and width regardless of whether this value is specified in the source. Internet Explorer does not return a value for height and width unless these values are specified in the source.

The default size property for combination boxes (select tag) might not be compatible across browsers. In Netscape, if the size property is not defined, Netscape returns a value of one, which results in a drop-down combination box. If the size property is defined, Netscape returns the value that is specified in the source. In Internet Explorer, if the size property is not defined, IE returns a value of zero, which results in a drop-down combination box. If the size property is defined, IE returns the value that is specified in the source. (If the size is zero or one, the result is a drop-down combination box. If the size is greater than one, the result is a list box.)

Netscape 4.7

*Administrator's Guide*Rational Robot has a new extension for the Netscape 4.7x environment. This extension is selected by default in the Rational Robot Extension Manager. A key feature of this new extension is that it supports the testing of SSL pages over HTTPS.

The Netscape 4.7x extension has the following restrictions:

- Some browser properties cannot be tested using Netscape 4.7x. For example, you cannot test whether a form element is disabled.
- Rational Robot does not support recording a script with the Netscape 4.x extension and then playing it back in Netscape 4.7x.
- For cross-browser testing, you should record scripts using Netscape and then play them back using Internet Explorer. One reason for this is that Netscape does not support the ID attribute. Internet Explorer, however, records using the ID attribute, if it exists. Recording under the Netscape 4.7x extension ensures that the recorded script defaults to using a recognition string that can play back under both browsers.
- To record HTML image clicks, the Alt attribute must be used in the underlying HTML for successful cross-browser testing between Netscape 4.7x and Internet Explorer.
- In Internet Explorer and earlier versions of Netscape, it was possible to create an Object Data verification point on HTMLText (in addition to Contents) for almost all supported objects. For example, for an HTMLlink, a test on Contents captured the URL, and a test on HTMLText captured the entire tag. Netscape 4.7x cannot test HTMLText, so the HTMLText option was removed for all ObjectData verification points under Netscape. As a result, scripts that were recorded in Internet Explorer or in earlier Netscape versions that used HTMLText will fail on playback under Netscape 4.7x. The verification point would need to be re-recorded to test Contents.

  Note: You can use Internet Explorer to test HTMLText.
- The name attribute is valid only for forms and links.
- Netscape ignores the thead, tbody, and tfoot tags within the table tag.

JDK 1.4 support

Rational Robot now supports JDK 1.4. For more information, see the *IBM Rational Robot User's Guide* and the Rational Robot Help.

Delphi 7 support

Rational Robot supports Delphi 7, which includes enhancements to object recognition. Users of any currently supported version of Delphi should carefully review the "Guidelines and Restrictions" section in the IBM Rational Robot Release Notes.

PowerBuilder support

Rational Robot supports PowerBuilder versions 8, 9, and 10.

Performance testing

This section describes features of Rational Robot that you can use to test performance.

Session recording and script generation extensibility

Rational Robot now supports customer-written adapters for session recording and script generation. This new framework is explained in the *Session Recording and Script Generation Extensibility Reference Guide*.

NTLM feature enhancement

HTTP session recording and TestManager VU script execution now support Microsoft Windows NT Challenge/Response Authentication (NTLM). For more information, see the *VU Language Reference Guide*.

SQABasic

This section describes the following additional and enhanced commands for .NET actions:

- DataGrid
- CheckedListBox
- TreeView
- ListView
- ListBox
- TabControl
- Calendar
- Toolbar

DataGrid

This command performs an action on a DataGrid component.

Syntax

DataGrid *action%*, *recMethod$*, *parameters$*

Parameters

action%

Select one of these mouse actions:

MakeSelection

Selects the specified item in a DataGrid

Check

Selects a check box

Uncheck
Clears a check box

recMethod$
Valid values: (Separate with a semicolon)

Type=
Used to identify the object within a specific context or environment

Name=
A name that a developer assigns to a parent or child object to identify the object.

parameters$
Valid values: (Separate with a semicolon)

Row=
Row number in the DataGrid.

Col=
Column number in the DataGrid.

Location=
See "Comments."

Comments

If the DataGrid is displaying a different table as a result of user navigation, the user can click on the **ParentDetailsButton** to navigate back to the parent table. Rational Robot would record the action in the following manner:

```
DataGrid Click,Type=Datagrid;Name=dg1,Location=BackButton
```

When child table information is displayed by the DataGrid, the parent row information is displayed. Users can show/hide the parent row information by clicking on the **ShowParentDetails** button. Rational Robot would record the action in the following manner:

```
DataGrid Click,Type=Datagrid;Name=dg1, Location=ParentDetailsButton
```

Examples

```
DataGrid MakeSelection,Type=Datagrid;Name=dg1,Row=1

DataGrid MakeSelection,Type=Datagrid;Name=dg1,Col=1

DataGrid Click,Type=Datagrid;Name=dg1, Row=1;Col=1
```

TreeView

This command performs an action on a tree view control.

Syntax

TreeView *action%*, *recMethod$*, *parameters$*

Parameters

action%
Select one of these mouse actions:

MakeSelection
Selects the specified item in a TreeView

Check
Selects a check box

Uncheck
  Clears a check box

Expand
  Expands the tree

Collapse
  Collapses the tree

recMethod$
  Valid values: (Separate with a semicolon)

Type=
  Used to identify the object within a specific context or environment

Name=
  A name that a developer assigns to a parent or child object to identify the object.

  See "Comments."

parameters$
  Valid values: (Separate with a semicolon)

Text=
  Display text associated with an item.

Comments

The action string includes the complete path information of a node from the top level parent. For example, if `node1->node1.1->node1.2` is a node hierarchy, it should be included in the action string.

Examples

```
TreeView MakeSelection,Type=TreeView;Name=treeView1,Text=Node-

TreeView Check,"Type=TreeView;Name=treeView1","Text=Node-"

TreeView Uncheck,"Type=TreeView;Name=treeView1","Text=Node-"

TreeView Expand,"Type=TreeView;Name=treeView1","Text=Node-"

TreeView Collapse,"Type=TreeView;Name=treeView1","Text=Node
```

ListView

This command performs an action on a list view control.

Syntax

ListView *action%, recMethod$, parameters$*

Parameters

action%
  Select one of these mouse actions:

MakeSelection
  Selects the specified item in a ListView

Check
  Selects a check box

Uncheck
  Clears a check box

recMethod$
Valid values: (Separate with a semicolon)

Type=
Used to identify the object within a specific context or environment

Name=
A name that a developer assigns to a parent or child object to identify the object.

parameters$
Valid values: (Separate with a semicolon)

Index=
Index of an item in the ListView items collection.

Text=
Display text associated with an item.

Comments

The action string can include Index only, Text only, or both to identify an item in the ListView.

Examples

```
ListView MakeSelection,Type=ListView;Name=lv1,Index=0;Text=John

ListView Check,Type=ListView;Name=lv1,,Index=0;Text=John

ListView Uncheck,Type=ListView;Name=lv1,,Index=0;Text=John
```

ListBox

This command performs an action on a list box control.

Syntax

ListBox *action%, recMethod$, parameters$*

Parameters

action%
Select one of these mouse actions:

MakeSelection
Selects the specified item in a ListBox

Ctrl_Click
Selects one or more specified items in a list box

ShiftCtrl_Click
Selects a range of specified items in a list box.

recMethod$
Valid values: (Separate with a semicolon)

Type=
Used to identify the object within a specific context or environment

Name=
A name that a developer assigns to a parent or child object to identify the object.

parameters$
Valid values: (Separate with a semicolon)

Index=
Index of an item in the list items collection.

Text=
Display text associated with an item.

Comments

The action string can include Index only, Text only, or both to identify an item in the ListBox.

Examples

```
ListBox MakeSelection,Type=ListBox;Name=lb1,Index=0;Text=Check
```

CheckedListBox

This command performs an action on a checked list box control.

Syntax

CheckedListBox *action%*, *recMethod$*, *parameters$*

Parameters

action%
Select one of these mouse actions:

MakeSelection
Selects the specified item in a CheckedListBoxView

Ctrl_Click
Selects one or more specified items in a list box

ShiftCtrl_Click
Selects a range of specified items in a list box.

recMethod$
Valid values: (Separate with a semicolon)

Type=
Used to identify the object within a specific context or environment

Name=
A name that a developer assigns to a parent or child object to identify the object.

parameters$
Valid values: (Separate with a semicolon)

Index=
Index of an item in the list items collection.

Text=
Display text associated with an item.

Comments

The action string can include Index only, Text only, or both to identify an item in the CheckedListBox.

Examples

```
CheckedListBox MakeSelection,Type=ListBox;Name=lb1,Index=0;Text=Check
```

TabControl

This command performs an action on tab control.

Syntax

TabControl *action%, recMethod$, parameters$*

Parameters

action%
Select one of these mouse actions:

MakeSelection
Selects the specified tab page in the tab control.

Click
Selects the specified tab page in the tab control.

DblClick
Selects the specified tab page in the tab control.

recMethod$
Valid values: (Separate with a semicolon)

Type=
Used to identify the object within a specific context or environment

Name=
A name that a developer assigns to a parent or child object to identify the object.

parameters$
Valid values: (Separate with a semicolon)

Index=
Index of an item in the tab pages collection.

Text=
Display text associated with a tab page.

Comments

The action string can include Index only, Text only, or both to identify a page in the TabControl.

Examples

```
TabControl MakeSelection,Type=TabControl;Name=TabControl 1,Index=0;Text=tabPage1
```

Calendar

This command performs an action on a calendar control.

Syntax

Calendar *action%, recMethod$, parameters$*

Parameters

action%
Select one of these mouse actions:

Click
Selects a date in the calendar.

Shift_Click
Selects an end date in the calendar to indicate a date range.

recMethod$
Valid values: (Separate with a semicolon)

Type=
Used to identify the object within a specific context or environment

Name=
A name that a developer assigns to a parent or child object to identify the object.

parameters$
Valid values: (Separate with a semicolon)

StartDate=
Individual date or beginning of a date range.

EndDate=
End of a date range.

Comments

The action string can include StartDate only or both StartDate and EndDate.

Examples

```
Calendar Click,"Type=Calendar;Name=monthCalendar1","StartDate=8/16/2004"

Calendar Shift_Click,"Type=Calendar;Name=monthCalendar1","StartDate=8/9/2004;
EndDate=8/15/2004"
```

ToolBar

This command performs an action on a toolbar control.

Syntax

ToolBar *action%, recMethod$, parameters$*

Parameters

action%
Click

recMethod$
Valid values: (Separate with a semicolon)

Type=
Used to identify the object within a specific context or environment

Name=
A name that a developer assigns to a parent or child object to identify the object.

parameters$
Valid values: (Separate with a semicolon)

Text=
Display text associated with a toolbar item.

Index=
Unique index to identify an item in the toolbar.

Location
See "Comments."

Comments

The action string can include Text only or both Text and Index.

If the toolbar item is displaying drop down menus, it can be displayed by clicking on the drop down button arrow. This could be recorded as:

```
Toolbar Click, "Type=Toolbar;Name=toolBar1", "Text=File;Location=DropDown"
```

Examples

```
Toolbar Click, "Type=Toolbar;Name=toolBar1", "Text=File"
```

```
Toolbar Click, "Type=Toolbar;Name=toolBar1", "Text=File;Index=1"
```

```
Toolbar Click, "Type=Toolbar;Name=toolBar1", "Text="Index=1"
```

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

(c) (your company name) (year) Portions of this code are derived from IBM Corp. Sample Programs. (c) Copyright IBM Corp. _enter the year or years_. All rights reserved.

Additional legal notices are described in the legal_information.html file that is included in your Rational software documentation.

Trademarks

AIX, ClearCase®, ClearCase Attache®, ClearCase MultiSite®, ClearDDTS®, ClearGuide®, ClearQuest®, DB2®, DB2 Universal Database™, DDTS®, Domino®, IBM, Lotus® Notes®, MVS™, Notes, OS/390®, Passport Advantage®, ProjectConsole™ Purify®, Rational, Rational Rose®, Rational Suite®, Rational Unified Process®, RequisitePro®, RUP®, S/390®, SoDA®, SP1, SP2, Team Unifying Platform™, WebSphere®, XDE™, and z/OS® are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX® is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

Readers' Comments — We'd Like to Hear from You

IBM Rational Robot
Documentation Supplement
Version 7.0.0.1

Publication No. GI11-6387-00

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
GI11-6387-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Attn. Dept. CZLA
20 Maguire Road
Lexington, MA
02421-3112
USA

Fold and Tape **Please do not staple** Fold and Tape

GI11-6387-00

Cut or Fold
Along Line

Printed in USA

GI11-6387-00