

ReportSmith Creating Reports

Delphi for Windows

Introduction

Copyright Agreement

Creating Reports explains how to create reports with ReportSmith™. It familiarizes you with basic tasks such as creating a basic report, and advanced topics such as working with macros.

Who should use this guide

Creating Reports is intended for users who want to create a variety of reports from their database tables. It assumes a general knowledge of the Windows environment. Advanced topics, such as macros and SQL entry, assume a database and programming background.

Topics covered in this guide

The following table lists topics you'll find in *Creating Reports* and provides a brief description of each.

Introduction discusses basic tasks, including how to create and format a basic report.

Chapter 1, "Connecting to a data source" lists databases that ReportSmith can access, tells you how to connect to them, and how to save connection information.

Chapter 2, "Types of reports you can create" gives examples of types of reports you can create, including default report types (columnar, crosstab, form and labels), and custom report types (summary-only and master/detail).

Chapter 3, "Bringing data into a report" introduces you to the Report Query dialog boxes, describes adding tables, choosing columns, assigning column aliases, specifying selection criteria and direct SQL entry.

Chapter 4, "Manipulating data" describes how to sort and group data, create derived fields, summary fields, report variables and temporary tables, and create a master/detail report.

Chapter 5, "Formatting" explains how to select and move parts of a report, set paper size and orientation, to format a report by adding borders, using drawing tools,



formatting characters, using the Style Extractor and report styles, aligning items, and displaying graphics filenames as pictures.

Chapter 6, “Graphs” describes how to convert a report to a graph.

Chapter 7, “Crosstabs” shows you types of crosstabs you can create, describes three methods for creating a crosstab, and shows you how to modify and format a crosstab.

Chapter 8, “Macros” introduces you to macros and ReportBasic, shows you how to create and use a simple macro, and gives examples of sample macros.

Chapter 9, “Printing a report” shows you how to print a report.

Chapter 10, “Customizing ReportSmith” shows you how to customize ReportSmith’s options, its menu and toolbar, and how to run a ReportSmith report from an icon.

Chapter 11, “Runtime Viewer” shows you how to install and run this feature.

Chapter 12, “Creating and using a data dictionary” shows you how to create a view, assign aliases, and use the ReportSmith data dictionary.

Appendix A, “Using ReportSmith with other applications” discusses how to embed an OLE 2.0 object into a report, how to call ReportSmith from another application, and how to call an application from ReportSmith.

Appendix B, “Macro reference” provides a command reference for ReportSmith’s macro facility.

What you can do with ReportSmith

ReportSmith is a powerful visual database reporting and query tool. It provides a streamlined approach to creating reports using database files and tables. You can create reports from SQL and PC databases without knowing complex database commands. You can bring database tables and files into ReportSmith to produce a report that you can edit and format, combining items such as data, text, charts, pictures, and soundbites. Since ReportSmith uses “live data,” you see the result of your changes immediately, using current data.

Once you bring data into ReportSmith, you can customize a report to perform tasks such as:

- Exclude data you don’t want to display
- Combine data from other tables
- Sort and group live data
- Insert headers and footers
- Perform sums, averages, minimums, and maximums
- Create dialog boxes that prompt for information
- Insert charts, text, pictures, and OLE objects from other applications
- Create distinct master/detail sections using data from separate tables
- Instantly apply preformatted styles and add text
- Attach macros to your report

Because ReportSmith uses a visual layout approach, you can specify data you want and use live data as you edit and format reports. You decide how you want data to appear, using drag and drop techniques to place data, graphics, OLE objects, charts, and text. ReportSmith saves the layout and uses it each time you run your report.

You can add finishing touches to your report such as fonts, number formats, and borders with Windows-standard features: icons, a toolbar, a formatting ribbon, snap-to grids, adjustable rulers, zoom, and alignment capabilities.

Terms, techniques, and conventions

Creating Reports uses certain terms and conditions to describe specific actions.

The keyboard

- Letters like *Esc* designate key names on your keyboard. Here, *Esc* represents the Escape key. Keys on your keyboard may not be labeled exactly as shown in this manual, so check your computer manual if they differ.
- *Arrow* keys refer to the *Up Arrow* (↑), *Down Arrow* (↓), *Left Arrow* (←), and *Right Arrow* (→) keys.
- Combination keys appear with a plus sign (+). For example, *Alt+F1* means you should hold down the *Alt* key while pressing the *F1* key.

Text

- The text you type appears in `monospace font`. Type in the text exactly as it appears, including uppercase and lowercase letters and spaces between words, letters, and symbols.
- Filenames and database table names appear in UPPERCASE.
- New terms and emphasized items appear in *italics*.
- A light bulb in the margin indicates a useful tip or shortcut.



Procedures

In many of ReportSmith's dialog boxes, you can use three methods to place an item from a list box into another list box. You can:

- Drag and drop it into the list box using the mouse
- Double-click on an item to place it in a list box
- Select an item and press the appropriate button in the dialog box

Installed files

The Delphi installation automatically installs some ReportSmith drivers and files for you. To view a list of these files, open the FILELIST.DOC installed with ReportSmith.

Directory locations for files

These are the default directory locations for ReportSmith and its components:

- ReportSmith program files in C:\RPTSMITH
- Sample tables, reports, and files in C:\RPTSMITH\VIDEO
- Sample macros in C:\RPTSMITH\MACROS
- IDAPI files in C:\IDAPI, or if you have already installed IDAPI these files are placed into the same directory
- ODBC files in WINDOWS\SYSTEM
- OLE 2.0 drivers in WINDOWS\SYSTEM

Sample tables, reports, and macros

Sample databases and reports used in the tutorials can be found in the RPTSMITH\VIDEO directory. Sample macros are in the RPTSMITH\MACROS directory.

The sample tables contain data used in the tutorials. The sample tables were designed to work together. You can define relationships between them by linking them. The sample reports offer examples of report types you can create.

Connecting to a data source

This chapter identifies the database types you can use with ReportSmith and shows you how to connect to your database tables. It also shows you how to save connections to frequently used tables.

Databases ReportSmith can access

With ReportSmith, you can retrieve information from the following databases. Check with your system administrator if you're unsure which database types you're using.

Databases ReportSmith can access

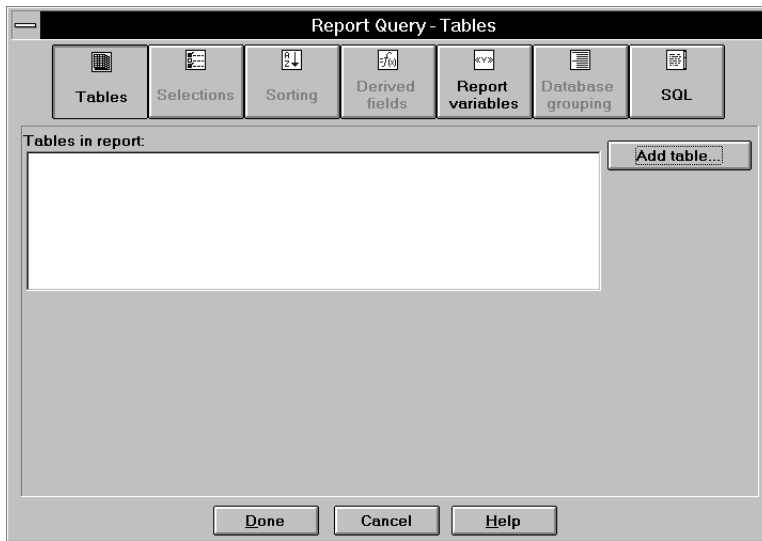
Access	Informix	SQLBase
Btrieve	InterBase	Sybase
DB2	ORACLE	Teradata
dBASE	Paradox	Text Files
Excel	Raima	Unify
FoxPro	AS/400	Watcom SQL
INGRES	SQL Server	Databases accessed through generic ODBC drivers

Connecting to tables

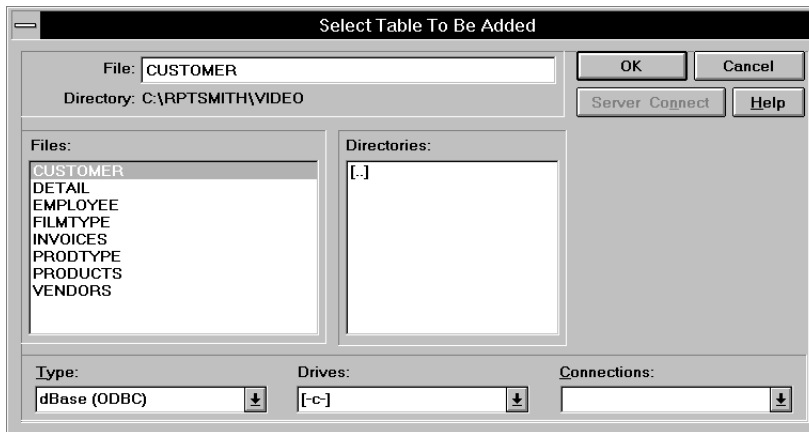
To connect to a database:

- 1 Choose File | New, or press the New Report button on the toolbar. ReportSmith displays the Create A New Report dialog box.

- 2 Press the report button that corresponds to the report type you want to create and press OK, or double-click the report button that corresponds to the report type you want to create. ReportSmith displays the Tables dialog box.



- 3 Choose Add Table. ReportSmith displays the Select Table To Be Added dialog box.

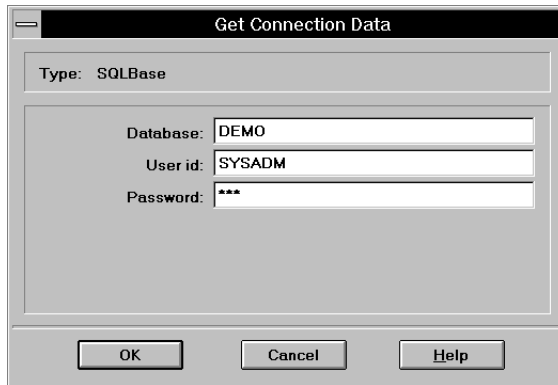


- 4 In the Type box, choose a database type. In the example above, we've chosen dBASE(ODBC). Specify the location of the database in the Drives and Directories list boxes, or if you've defined a connection, select it from the Connections list box.

A list of available tables appears in the Files box.

- 5 Select a table and press OK, or choose Server Connect to make a server-based connection.

The Get Connection Data dialog box appears with the connection type displayed at the top. In the example below, the connection type is SQL Server.



If you specify a server-based database such as SQL Server, ORACLE or INGRES you'll be prompted for server, user ID and password. Information you enter in the dialog box varies depending on the type of database to which you're connected.

If you specify a local PC-based database such as dBASE, Paradox, or Btrieve, go to the directory where their files are located and select a table.

- 6 Enter the information for the type of connection you want to make. In the example above, ReportSmith prompts you to enter the name of the server, the user ID, and password. (For specific information you'll need for each database type, see ReportSmith's on-line Help.) Press OK.

When you save a new report, ReportSmith saves all the connection information except the password. The next time you create a new report, ReportSmith displays the connection information from the last data source you connected to. You can cancel it if you intend to use a different data source.

Saving database connection information

ReportSmith can save database login information in *named connections*. With named connections, you can

- Save time by not having to retype database information that you use frequently
- Shield the end-user from remembering server names or data paths by setting up their connections for them

A *connection* points to the location of files, a specific user and database tables you want to include in a report. You can set up permanent or temporary connections by telling ReportSmith the data path or server information for your system.

Note You do not need a user-defined connection to connect to a table.

You can set up permanent connections that let you quickly access database tables that you use frequently. A *user-defined connection* lets you access a database without having

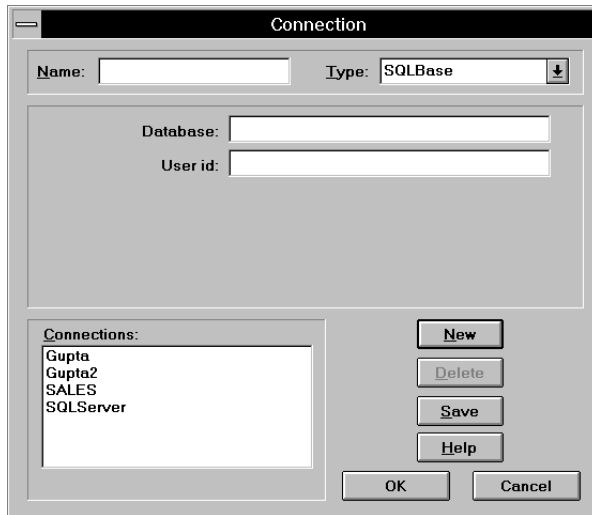
to re-enter the server name, user name, and directory path. Instead, you can select a user-defined connection and ReportSmith will login to that database directly.

If required, ReportSmith prompts you to enter your password and connects you to the database. Once you create a connection and save it, the next time you access the database you created for that connection, you simply select the name you assigned to the connection and ReportSmith makes the connection.

Note See your system administrator for server IDs and help in setting up connections you plan to use frequently.

To set up and save a connection:

- 1 Choose File | Connections. The Connection dialog box appears.



- 2 Choose New to set up a new connection.

- 3 Select the database from the Type list that you want to set up a connection for from the drop down list.

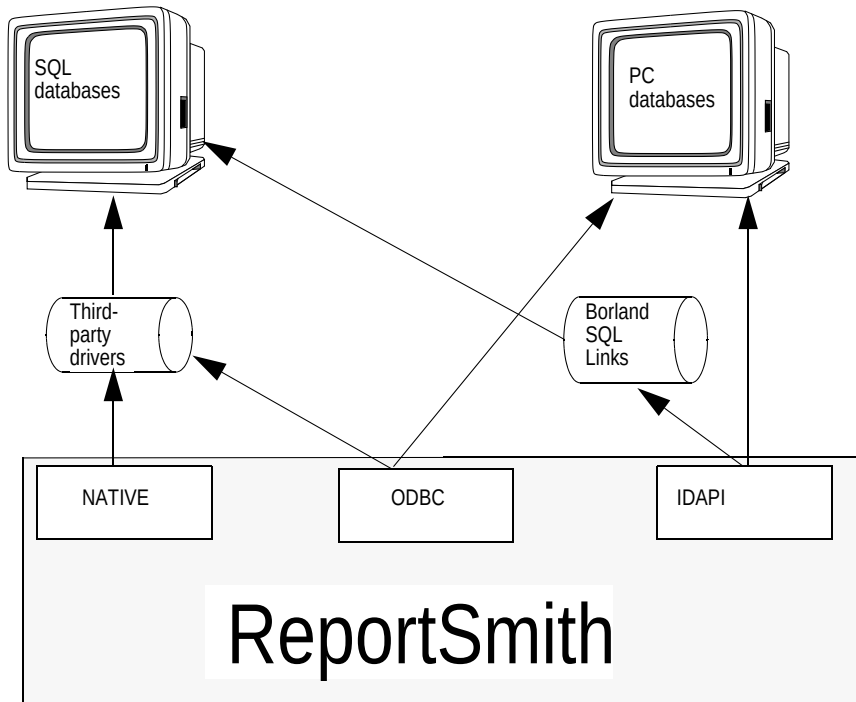
Note Displayed options change for each type of connection.

- 4 In the Name box enter a name for the connection. For example, you may have a database that holds employee sales records that you want to name *Sales*.
- 5 Enter the information for your connection type. ReportSmith needs the data path or server information for your system.
- 6 Choose Save. ReportSmith saves the connection and places it in the Connections box. To delete a connection, select it in the Connections box and choose Delete. ReportSmith removes the connection.
- 7 Press OK. Now you can select this connection in the Connections box in the Select Table To Be Added dialog box the next time you create a report.

Connecting to data sources

This chapter provides an overview of the three types of drivers ReportSmith uses to connect to data sources: IDAPI, ODBC, and native connections. It also describes how to connect to each data source and provides solutions to common questions.

The following diagram shows how ReportSmith communicates with databases. ReportSmith provides middleware (native, ODBC, and IDAPI) that allows you to connect to various databases. Third-party middleware, such as Q+E's ODBC drivers and Borland® SQL Links, can be purchased to allow ReportSmith to connect to additional data sources (see "Supported data sources" on page 10).



Connection types

ReportSmith provides three types of connections to enable you to connect to data sources: IDAPI, ODBC, and native connections. You may use one or more of these connections, depending on the data sources you are working with.

Through IDAPI

Integrated Database Application Programming Interface (IDAPI) enables you to connect to IDAPI-enabled Borland desktop products such as Paradox and dBASE, or SQL data sources supported by the Borland SQL Links, such as InterBase (for a

complete list of supported IDAPI data sources, see “Supported data sources” on page 10).

Note For more information about IDAPI connectivity to SQL databases, double-click on the IDAPI Configuration icon in the ReportSmith program group and choose Help in the IDAPI Configuration Utility after you install the product.

What are SQL Links?

Borland SQL Links enables IDAPI application users to access SQL databases. After you install SQL Links and create an SQL driver alias, you can use your IDAPI-enabled applications such as ReportSmith to access SQL databases.

Through ODBC

Open Database Connectivity (ODBC) drivers enable you to connect to PC and SQL ODBC data sources. You can install your own or use those drivers installed by ReportSmith. If you have installed your own ODBC drivers, ReportSmith overwrites existing drivers if they are older versions than those shipped with ReportSmith (for a complete list of supported ODBC data sources see “Supported data sources” on page 10).

Note For more information about ODBC connectivity double-click on the ODBC Administrator icon in the Windows Control Panel or choose File | ODBC Setup in ReportSmith after you install the product.

Through native API

A native Application Programming Interface (API) connection is integrated into ReportSmith and enables you to connect to SQL data sources. Since they are integrated within ReportSmith, native connections are able to retrieve data sets faster than other connection types. Therefore, we recommend choosing native connections before ODBC or IDAPI.

Supported data sources

You can use the samples provided by ReportSmith to learn how to use the product. However, to create reports using your own data, you will need to connect to a data source that is supported by ReportSmith. The table below provides a list of databases and files supported by ReportSmith.

Database	Version	Connection type
Access	All versions 1.0 through 2.x. ReportSmith does not supply the ODBC driver for Access 2.0; it's included when you install Access 2.0.	ODBC
Btrieve	BTRIEVE.EXE versions 5.x and earlier. Connectivity to Btrieve 6 is available through the ODBC driver from Q+E.	ODBC

Database	Version	Connection type
DB2	Supported through the MicroDecisionware Database Gateway version 2.0 running on an OS/2 server, and the Gupta DB2 Gateway provided by Gupta. (Before you can connect to DB2, a gateway must be installed.)	Native
dBASE	dBASE II, III, and IV file formats and dBASE for Windows. Through IDAPI, version 5.0 and earlier.	IDAPI, ODBC
Excel	ReportSmith accesses only the information in the database portion of an Excel spreadsheet for version 4.0 or earlier.	ODBC
FoxPro	Versions 2.0, 2.5.	ODBC
Informix	Versions 4.1.0 through SQL Links.	IDAPI
Ingres	Ingres Networking for Windows version 6.4/04.	Native
InterBase	Versions 3.3 through SQL Links. Versions 4.x through Client Server Express and SQL Links.	IDAPI
ORACLE	SQL*Net for DOS or Windows access to remote ORACLE databases, versions 6.0 and 7.x through 7.2.	Native
Paradox	Version 5.0 or earlier. For ODBC, versions 4.x and earlier.	IDAPI, ODBC
SQL Server	See the README.TXT.	Native
SQLBase	Gupta SQLBase Server products, version 4.0 and Quest 1.0 to access SQLBase databases.	Native
Sybase	See the README.TXT.	Native
Teradata	Supported through the MicroDecisionware Database Gateway, version 2.0 running Teradata DBC/1012, version 4.1.1 or greater.	Native
Text files	Text files are supported through an ODBC driver provided by ReportSmith.	ODBC
Unify	Unify 2000. The necessary Unify client software can be obtained from Borland.	Native

Note For information about installing a specific data source, see “Connecting to data sources” on page 9.

Connecting to PC data sources through IDAPI

ReportSmith for PC databases connects to dBASE and Paradox through IDAPI, provided by ReportSmith. This section describes each connection type, hardware and software requirements, and connection issues that may occur for each data source.

Note Dialog boxes include example parameters.

dBASE

ReportSmith provides an IDAPI connection to dBASE[®] versions 5.0 and earlier and dBASE V for DOS. For dBASE for Windows 5.0, IDAPI provides the only connection.

Note For detailed information about dBASE IDAPI connections, double-click on the Release Notes icon in the ReportSmith program group.

Paradox

ReportSmith provides an IDAPI connection to Paradox[®] versions 5.0 and earlier. For instructions on how to configure a Paradox connection, see your IDAPI documentation.

Connecting to PC data sources through ODBC

ReportSmith for the PC connects to PC data sources through ODBC. This section describes ODBC and describes the connection type, hardware and software requirements, and connection issues that may occur for each data source. ReportSmith installs default data sources for ODBC.

To add, modify, or delete ODBC drivers, you should have the ODBC Control Panel option (or the ODBC Administrator program if you're running Windows version 3.0a) installed on your computer. From ReportSmith you can choose File | ODBC Setup.

Note Dialog boxes include example parameters.

Access

ReportSmith provides an ODBC connection to Access version 1.x. The driver for Access 2.x is included with Microsoft Access. The ODBC Access Setup window that appears when you choose to add the Access driver in the ODBC setup program is shown here:



Hardware and software requirements

To use Access data, you must have:

- The ODBC Access driver (provided by ReportSmith)
- The ODBC Driver Manager 1.0 (ODBC.DLL)

You must edit the default data source installed by ReportSmith to specify the path to your Access database.

Make the following changes to your CONFIG.SYS and AUTOEXEC.BAT files:

- Add this line to your CONFIG.SYS file: FILES= 50
- Add this line to your AUTOEXEC.BAT: C:\DOS\SHARE.EXE /L:200. For large databases, add this line to your AUTOEXEC.BAT: C:\DOS\SHARE.EXE /L:500

Note For more information about Access databases, refer to the Access documentation.

To configure an Access ODBC data source:

- Refer to the following table and enter the required fields in the ODBC Access Setup dialog box. When you're done, press OK to create the ODBC configuration.

Setup field	Value
Data Source Name	A short name for your data source.
Description	A description of the data source (optional).
Database	Choose a database from the list that displays when you press the Select Database button. Or, enter the drive, directory path, and name of the database. For example, enter C:\RPTSMITH\VIDEO.DBF.
System Database	Choose a database from the list that displays when you press the Select Database button. Or, enter the drive, directory path, and name of the database. For example, enter C:\RPTSMITH\VIDEO.DBF. You can choose either None or Database.
Page Timeout	The number of seconds that an unused page remains in the buffer before being removed (optional).
Exclusive	Check Exclusive to open Access files in exclusive mode. Exclusive mode prevents concurrent access of files and enhances performance (optional).
Buffer Size	The size of the internal buffer in kilobytes that is used by Access to transfer data to and from the disk.

Access connection issues

Symptom	Solution
Error message: "Driver not capable," or "File locking not available."	You are not running Share. You need to add C:\DOS\SHARE.EXE to your AUTOEXEC.BAT.
You can't access your Access 2.0 tables.	You must use the data source installed with Microsoft Access 2.0: Microsoft Access 2.0 databases.

Btrieve

ReportSmith provides an ODBC connection to Btrieve versions 5x and below. Before ReportSmith can use Btrieve files, you must first incorporate them into XQL databases. Incorporating Btrieve files into XQL doesn't change the files. You may still access Btrieve files with Btrieve applications. It creates data dictionary information contained in three files: FILE.DDF, FIELD.DDF, and INDEX.DDF. These files must be in the same directory as your Btrieve files. Several vendors provide utilities to generate Btrieve data definition files, for example, Xtrieve (by Novell), and Data Junction (by Tool and

Techniques Inc.). If you have Btrieve data from another application, that vendor may have already provided it in XQL database format.

The ODBC Btrieve Setup dialog box that appears when you choose the Btrieve driver is shown below:

The screenshot shows the 'ODBC Btrieve Setup' dialog box. It has a title bar with the text 'ODBC Btrieve Setup'. The dialog is divided into several sections. The 'Data Source Name' section has a text box containing 'RS_Btrieve' and an 'OK' button. The 'Description' section has a text box containing 'This is the ODBC Btrieve driver.' and 'Cancel' and 'Help' buttons. The 'Database' section has a 'Directory:' label, a 'Select Directory...' button, and a checked checkbox labeled 'Use Current Directory' with an 'Options>>' button. The 'Driver' section has 'Collating Sequence' set to 'ASCII', 'Page Timeout' set to '600', and an unchecked 'Exclusive' checkbox.

Hardware and software requirements

To access Btrieve data, you must have:

- The ODBC driver provided by ReportSmith
- Novell Btrieve for Windows (WBTRCALL.DLL)

The ODBC Btrieve driver requires the standalone Btrieve for Windows dynamic-link library (DLL), WBTRCALL.DLL. This file must be in the directory that contains Windows system DLLs (usually the C:\WINDOWS\SYSTEM directory).

To check if you already have a copy of WBTRCALL.DLL on your hard drive:

- 1 In the Windows Main group, choose the File Manager icon.
- 2 Choose File | Search to view the Search dialog box.
- 3 In the Search For box, type: `WBTRCALL.DLL`
- 4 Make sure that the “Search All Sub-Directories” option is checked.
- 5 Type `C:\` into the Start From box and press OK.
- 6 If the WBTRCALL.DLL file exists, copy it to the directory that contains Windows system DLLs (usually the C:\WINDOWS\SYSTEM directory if it isn’t already there).
- 7 If you do not have a copy of the WBTRCALL.DLL file, contact Novell, Inc. at 1-800-453-1267 or your local software dealer.

Note For more information about Btrieve files, refer to your Btrieve documentation.

To configure a Btrieve ODBC data source, refer to the following table and enter the values in the required fields into the ODBC Btrieve Setup dialog box. When you're done, press OK to create the ODBC configuration.

Setup field	Value
Data Source Name	A short name for your data source.
Description	A description of the data source (optional).
Directory	The directory from which this data source will access Btrieve files. You can choose "Use Current Directory" as a setting. It allows the driver to access appropriate data in whatever directory you are currently in.
Collating Sequence	Choose ASCII or International collating sequence for sorting. The default is ASCII.
Page Timeout	The number of seconds that an unused page remains in the buffer before being removed (optional).
Exclusive	Check if you want to open files in exclusive mode. Exclusive mode prevents concurrent access of files and enhances performance.

To modify Btrieve functions in the WIN.INI file:

- 1 Go to the Btrieve section of the WIN.INI file.
- 2 We recommend you modify the Btrieve section of your WIN.INI file to look like the following: `options=/p:4096 /m:64 /b:16 /f:20 /l:40 /n:12 /t:c:\rptsmith\btrieve.trn`

The XQL database lists Btrieve file names and definitions for each field in a file. You must also have the following three data files in the directory:

- FILE.DDF
- FIELD.DDF
- INDEX.DDF

Note All Btrieve files must be in the same directory as these generated data definition files.

Btrieve connection issues

Symptom	Solution
You're having trouble connecting to Btrieve.	Make sure you're accessing the data on your local drive. If you're accessing data on a network, make sure you have WBTRCALL.DLL installed.
You want to access a Btrieve file consisting of more than 4,096 pages.	Make sure the options in the Btrieve section of the WIN.INI file are the same as those specified above.
You receive an ODBC dialog box showing only FILE.DDF in the Files list.	Make sure your Btrieve options in your WIN.INI are correct (see above).

dBASE/xBASE

ReportSmith provides an ODBC connection to dBASE/xBASE that supports dBASE II, III, and IV file formats. ReportSmith supports xBASE formats that are compatible with dBASE versions II, III, or IV. ODBC does not support dBASE for Windows version 5.0 or dBASE V for DOS. These are supported by IDAPI.

The ODBC dBASE Setup dialog box that appears when you select the dBASE driver is shown below:

Hardware and software requirements

To access dBASE data, you must have:

- The ODBC driver provided by ReportSmith
- dBASE II, III, or dBASE IV database files
- The ODBC Driver Manager version 1.0 (ODBC.DLL)

Make the following settings in your CONFIG.SYS and AUTOEXEC.BAT files:

- Add this line to your CONFIG.SYS file: FILES= 50
- Add this line to your AUTOEXEC.BAT: C:\DOS\SHARE.EXE /L:200. For large databases, add SHARE.EXE /L:500 to your AUTOEXEC.BAT.

Note For more information about dBASE databases, refer to your dBASE documentation.

To configure a dBASE ODBC data source:

- Refer to the following table and enter the values shown in the required fields in the ODBC dBASE Setup dialog box. When you're done, press OK to create the ODBC configuration.

Setup field	Value
Data Source Name	A short name for your data source.
Description	A description of the data source (optional).
Version	Click the radio button representing your version of dBASE.
Directory	Select a directory from the list that displays after you click the Select Directory button or enter the name of the data source directory. You can choose "Use Current Directory" as a setting. It allows the driver to access appropriate data in whatever directory you are currently in.
Collating Sequence	The sort order: ASCII or International. The default is ASCII (optional).
Page Timeout	The number of seconds that an unused page remains in the buffer before being removed (optional).

Setup field	Value
Exclusive	Check if you want to open dBASE files in exclusive mode. Exclusive mode prevents concurrent access of files and enhances performance.
Show Deleted Rows	Check if you want rows marked as deleted to display (optional).
Approximate Row Count	Uncheck if you do not want the table size statistic approximated. We recommend that you do not check this option (optional).

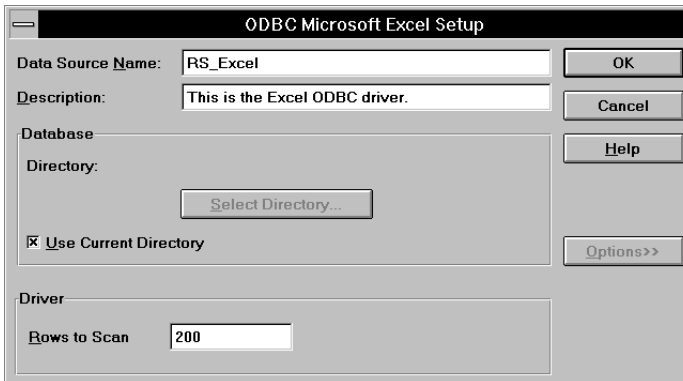
dBASE connection issues

The following information may be helpful in setting up your dBASE connection to ReportSmith.

Question	Answer
Does ReportSmith support indexing?	ReportSmith supports indexes for dBASE IV as long as the indexes have the same name as the data files. ReportSmith also supports indexing for dBASE III, however, you must select the location of the index file by pressing the Select Indexes button on the dBASE ODBC setup.
What does the error message, "File locking not available" or "Driver not capable" mean?	You are not running Share. You need to add C:\DOS\SHARE.EXE to your AUTOEXEC.BAT.

Excel

ReportSmith provides an ODBC connection to Excel versions 4.x and below. ReportSmith accesses only the information in the database portion of an Excel spreadsheet. The ODBC Excel Setup dialog box that appears when you select the Excel driver is shown below:



Hardware and software requirements

To access an Excel database, you must have

- The ODBC Excel driver (provided by ReportSmith)
- The ODBC Excel driver can access Excel version 3.0 (BIFF3) or 4.0 (BIFF4) work sheet files

- ODBC Driver Manager version 1.0 (ODBC.DLL)

To use a spreadsheet, you must save it in database format first. To do so, highlight your spreadsheet and then choose the Set Database option from the Data menu in Excel. For more information, refer to your Excel documentation.

To configure a Excel ODBC data source:

- Enter the values shown in the following table in the required fields in the ODBC Excel Setup dialog box. When you're done, press OK to create the ODBC configuration.

Setup field	Value
Data Source Name	A short name for your data source. The default is RS_Excel.
Description	A description of the data source (optional).
Database Directory	The directory from which this data source will access Excel files. You can check "Use Current Directory." You can choose "Use Current Directory" as a setting. It allows the driver to access appropriate data in whatever directory you are currently in.
Rows to Scan	The number of rows you want the driver to scan to determine the data type of each column in the spreadsheet.

Note For more information about Excel databases, refer to your Excel documentation.

Excel connection issues

The following information may be helpful in setting up your Excel connection to ReportSmith.

Symptom	Solution
You can't connect to Excel.	ReportSmith reads Excel spreadsheets that have been converted to a database using the database facility within Excel. ReportSmith won't connect to Excel if you're trying to access a spreadsheet that hasn't been converted. To learn how to convert your spreadsheet, refer to your Excel documentation for instructions.

FoxPro

ReportSmith provides an ODBC connection to FoxPro versions 2.0 and 2.5. The ODBC FoxPro Setup dialog box that appears when you select the FoxPro driver is shown below:

Hardware and software requirements

To access FoxPro data, you must have:

- The ODBC FoxPro driver provided by ReportSmith
- FoxPro version 2.0 or FoxPro version 2.5
- A computer running MS-DOS version 3.3 or greater

Make the following settings in your CONFIG.SYS and AUTOEXEC.BAT files:

- Add this line to your CONFIG.SYS: FILES= 50
- Add this line to your AUTOEXEC.BAT: C:\DOS\SHARE.EXE /L:200. If you are working with large databases, add SHARE.EXE /L:500

For more information about FoxPro databases, refer to your FoxPro documentation.

To configure a FoxPro ODBC data source:

- Enter the values shown in the table below in the required fields in the ODBC FoxPro Setup dialog box. When you're done, press OK to create the ODBC configuration.

Setup field	Value
Data Source Name	A short name for your data source.
Description	A description of the data source (optional).
Version	Click the radio button representing your version of FoxPro.
Directory	Choose a directory from the list that displays after you click the Select Directory button or enter the name of the data source directory. Or choose "Use Current Directory." It allows the driver to access appropriate data in whatever directory you are currently in. You can also add in Select Indexes (optional).

Setup field	Value
Collating Sequence	The sort order: ASCII or International. The default is ASCII.
Page Timeout	The number of seconds that an unused page remains in the buffer before being removed.
Exclusive	Check if you want to open FoxPro files in exclusive mode. Exclusive mode prevents concurrent access of files and enhances performance (optional).
Show Deleted Rows	Check if you want rows marked as deleted to display (optional).
Approximate Row Count	Uncheck if you do not want the table size statistics approximated. We recommend that you not choose this option (optional).

FoxPro connection issues

The following information may be helpful in setting up your FoxPro connection to ReportSmith.

Symptom	Solution
You receive the error message, "Driver not capable" or "File locking not available."	You are not running Share. You need to add C:\DOS\SHARE.EXE to your AUTOEXEC.BAT.
Poor performance when linking, joining, or sorting FoxPro 2.0 tables.	You must run the ODBC Administrator, choose the FoxPro data source, change the source version to FoxPro 2.0, and manually assign indexes for the driver to recognize them.
Poor performance when linking, joining, or sorting Clipper tables.	The MicroSoft FoxPro ODBC driver does not recognize Clipper .NTX index files. Try building .CDX indices.

Paradox

ReportSmith provides an ODBC connection to Paradox versions 3.x and 4.x. The ODBC Paradox Setup dialog box that appears when you select the Paradox driver is shown below:

The screenshot shows the 'ODBC Paradox Driver Setup' dialog box. The 'Data Source Name' field is filled with 'RS_Paradox'. The 'Description' field contains the text 'This is the ODBC Paradox driver.'. The 'Database Directory' field is set to 'C:\rptsmith\video'. To the right of these fields are buttons for 'OK', 'Cancel', and 'Help'. Below these fields is a section titled 'Optional Settings'. It includes a 'File Open Cache' field, a 'Table Version' field with a downward arrow, and a 'Sort Order' field set to 'ASCII' with a downward arrow. At the bottom, there are three unchecked checkboxes: 'Lock While Reading Blobs', 'International Sort', and 'Read Consistency Isolation Level'.

Enter the values shown in the table below in the required fields and press OK to create the ODBC configuration.

Hardware and software requirements

To access Paradox data, you must have:

- The ODBC Paradox driver provided by ReportSmith
- The ODBC Driver Manager version 1.0 or greater (ODBC.DLL)
- SHARE.EXE loaded on your machine

To configure a Paradox ODBC data source:

- Enter the values shown in the table below in the required fields in the ODBC Paradox Setup dialog box and then click the OK button to create the ODBC configuration.

Setup field	Value
Data Source Name	A short name for your data source. This name also becomes the name of the profile that is created for you.
Description	(Optional.) A description of the data source.
Database Directory	The name of the data source directory.
FileOpenCache	(Optional.) The maximum number of unused file opens to cache.
Table Version	The table version you are using in your Paradox database.
Sort Order	The sort order: ASCII or International. The default is ASCII.
Read Consistency Isolation Level	(Optional.) Check if you want the table to be write-locked to ensure that all records can be re-read.

Paradox connection issues

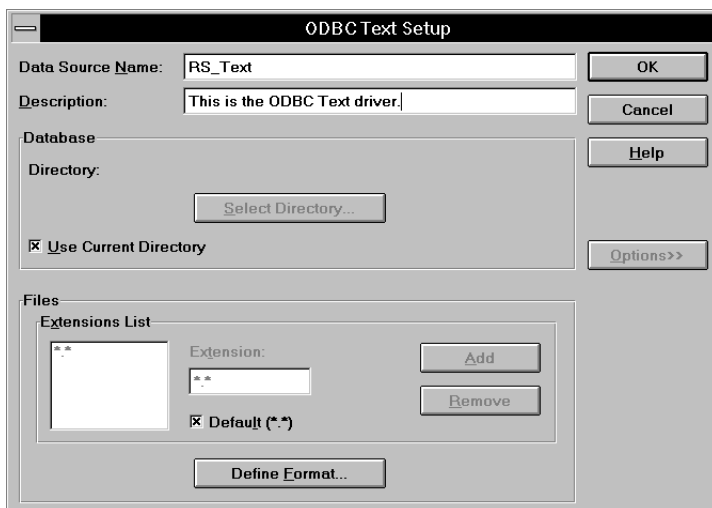
The following information may be helpful in setting up your Paradox connection to ReportSmith.

Symptom	Solution
ReportSmith displays an error message stating that SHARE hasn't been loaded.	Make sure you're running the DOS SHARE.EXE program, which is required by Paradox. Place C:\DOS\SHARE.EXE in your AUTOEXEC.BAT.
You receive the error message, "Driver not capable" or "File locking not available."	You are not running Share. You need to add C:\DOS\SHARE.EXE to your AUTOEXEC.BAT.
You receive the error, "[Q+E Software] [ODBC Paradox driver][Paradox]Field not found:yourfield" when sorting, grouping, creating derived fields.	Go into Tools Tables and remove the lowercase "x" from the assigned alias name.
You change to a directory containing your Paradox tables, but no files appear.	Verify that there is no underscore in the path to your files.
You get the error, [Q+E Software][ODBC Paradox driver][Paradox] Cannot open file: invalid directory name[table name]	Launch PXENGCFG.EXE in the ReportSmith directory and verify that your "Network Control File Path" is valid.
You get the error, [Q+E Software][ODBC Paradox driver][Paradox] Incompatible data types in expression: 'TABLE'. 'DATEFIELD'='1994-01'))	Add the following to the \WINDOWS\RS_SQLIF.INI file: DateFormat=""{d"yyyy-mm-dd"}"" TimeFormat=""{t"hh:mm:ss"}"" DateTimeFormat=""{ts"yyyy-mm-dd hh:mm:ss.sssss"}""
After you've selected your Paradox table, you are prompted for a password, but one is not assigned.	Verify that you are using the Q+E Paradox File (in ReportSmith version 2.X use "RS_Paradox") data source, and not the Microsoft data source.

Symptom	Solution
You get an error, [Q+E Software][ODBC Paradox driver][Paradox] Engine not initialized [your table]	Launch PXENGCFG.EXE in the ReportSmith directory, and verify that your "Network Control File Path" is valid. Save the changes and restart ReportSmith.
You get an error, [Q+E Software][ODBC Paradox driver][Paradox] Cannot open file: Can't handle version number encountered [your table].	This is a Paradox 5.0 table; use the IDAPI connections.

Text

ReportSmith provides an ODBC connection to Text files. The ODBC Text Setup dialog box appears when you select the Text file driver.



Hardware and software requirements

To access text files, you must have:

- The ODBC Text driver provided by ReportSmith
- The ODBC Driver Manager version 1.0 (ODBC.DLL)

To configure a text ODBC data source:

- Enter the values shown in the table below in the required fields. When you're done, press the OK button to create the ODBC configuration.

Setup field	Value
Data Source Name	A short name for your data source.
Description	A description of the data source (optional).
Directory	The directory where your text files are located. Or choose "Use Current Directory." It allows the driver to access appropriate data in whatever directory you are currently in.
Extensions List	Lists the filename extensions of the text files on the data source.

Setup field	Value
Extension	The file extension. If Default is checked, this is the default extension (optional).
Default (*.*)	Check to make the extension in Extension the default extension (optional).
Define Format	Click to define a schema for the individual tables in the directory. Schema definitions are stored in a special file called SCHEMA.INI (optional).

Press the Define Format button to view the Define Text Format dialog box:

Options within the Define Text Format dialog box are described in the following table.

Setup field	Value
Tables	This list box contains the names of tables defined in the directory. If a schema entry exists for the highlighted table, the Columns side of the window contains information about each column in the table.
Column Name Header	Check this box if the first row in the table contains column name names instead of data.
Format	Choose one of the following text file formats: Comma-Separated Values (CSV): Columns are separated by commas. Tab-Separated Values: Columns are separated by tabs. Custom Delimited: Columns are separated by the delimiter character specified. Fixed Length: Columns are not separated; rather, each column is a fixed length.
Delimiter	The character that acts as the column separators (custom delimited files only).
Rows to Scan	The number of rows you want the driver to scan to determine the data type of each column in the table (custom delimited files only).

Setup field	Value
Columns, Data Type, Name, and Width	If a schema entry exists for the highlighted table, the Columns list box contains the names of each column in the table. If a schema entry does not exist, click the Guess button to automatically generate the column data type, name, and width values for the columns in the selected table by scanning the table's contents according to the Format list box selection. Any previously defined columns in the Columns list are cleared and replaced with new entries. Guess is available only for CSV Delimited, Custom Delimited, and Tab-Separated Value formats. These fields let you specify the schema for each data source. This information is written to a SCHEMA.INI file in the data source directory. There is a separate SCHEMA.INI for each text data source directory. If the table format is fixed-length, Width is valid for all data types. If the table is any other format, Width is valid only for Char or LongChar data types.
Date Separator	The character that acts as a separator in dates (custom delimited files only).

Connecting to SQL databases through IDAPI

ReportSmith enables you to connect to SQL databases through IDAPI. To connect to these tables you must install and configure Borland SQL Links.

Note Refer to your SQL Links documentation for instructions on installing and configuring SQL Links.

InterBase

ReportSmith provides an IDAPI driver that connects to InterBase® versions 3.3 and 4.0 through SQL Links and version 4.0 through Client/Server Express.

Note For complete instructions on connecting to InterBase, refer to *Connecting to InterBase* in the Borland *SQL Links for Windows* documentation set.

Informix

ReportSmith provides an IDAPI driver that connects to Informix version 4.1.0 through SQL Links.

Note For complete instructions on connecting to Informix, refer to *Connecting to Informix* in the Borland *SQL Links for Windows* documentation set.

Connecting to SQL databases through native connections

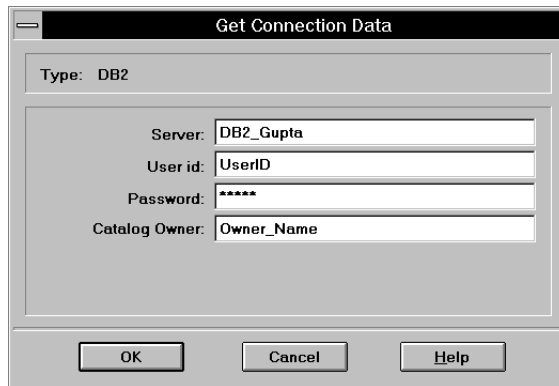
ReportSmith for SQL databases connects to all of the local PC databases listed earlier and the following server databases through IDAPI, ODBC, and native connections. This section describes each connection type, hardware and software requirements, and connection issues that may occur for each data source.

Note We've filled dialog boxes with valid parameters to illustrate examples; yours may differ.

DB2

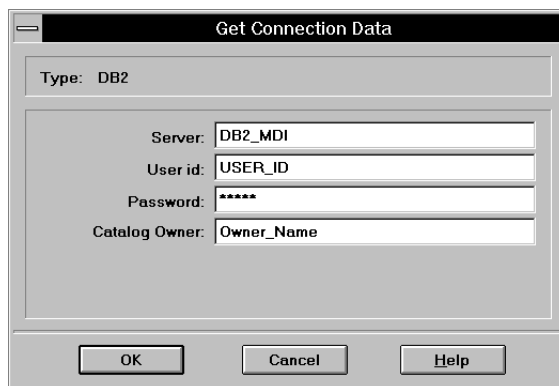
ReportSmith provides connections to DB2 via the MicroDecisionware Database (MDI) Gateway and the Gupta DB2 Gateway. Before you can connect to DB2, the gateway must be installed and working.

The Get Connection dialog box that appears when you select a DB2 Gupta ODBC connection is shown here.



The dialog box titled "Get Connection Data" has a "Type: DB2" label. It contains four input fields: "Server:" with the value "DB2_Gupta", "User id:" with the value "UserID", "Password:" with masked characters "*****", and "Catalog Owner:" with the value "Owner_Name". At the bottom are three buttons: "OK", "Cancel", and "Help".

The Get Connection dialog box that appears when you select a DB2 MDI ODBC connection is shown here.



The dialog box titled "Get Connection Data" has a "Type: DB2" label. It contains four input fields: "Server:" with the value "DB2_MDI", "User id:" with the value "USER_ID", "Password:" with masked characters "*****", and "Catalog Owner:" with the value "Owner_Name". At the bottom are three buttons: "OK", "Cancel", and "Help".

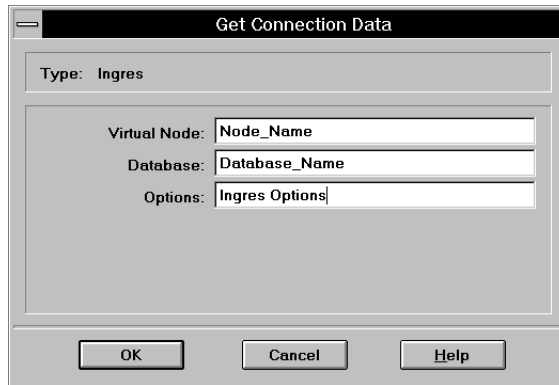
Ingres

ReportSmith provides a native API connection to Ingres Networking for Windows.

Hardware and software requirements

To connect to Ingres for ReportSmith you will need Ingres Networking for Windows version 6.4/04.

When you choose an Ingres connection, the Get Connection dialog box appears, prompting you for the following values:



The image shows a Windows-style dialog box titled "Get Connection Data". Inside the dialog, there is a section labeled "Type: Ingres". Below this, there are three input fields: "Virtual Node:" with the text "Node_Name", "Database:" with the text "Database_Name", and "Options:" with the text "Ingres Options". At the bottom of the dialog, there are three buttons: "OK", "Cancel", and "Help".

ORACLE

ReportSmith provides a native API connection to SQL*Net for DOS. It also provides Windows access to remote and ORACLE databases versions 6.0, 7.0, and 7.1. If you have an ORACLE ODBC driver, you can use it. However, this is the connection type we recommend.

Hardware and software requirements

ORACLE runs on a number of different servers in various environments. ReportSmith provides two ways to connect to ORACLE via SQL*Net:

- SQL*Net for Windows
- SQL*Net for DOS

To connect to ORACLE remotely:

- Use SQL*Net for Windows.
- Use SQL*Net for DOS.

Load the appropriate Terminate and Stay Resident (TSR) program for your network. For example, TCPIP connections use SQLTCP and SPX/IPX uses SQLSPX, provided by ORACLE. Additionally, there is an ORACLE configuration file referenced by an environment variable called *config*. Your AUTOEXEC.BAT typically contains a line such as:

```
set config=c:\oracle6\config.ora or set config=C:\oracle7\config.ora
```

To connect to ORACLE locally:

- Use SQL*Net for Windows.
- Use SQL*Net for DOS.

Load two TSRs, SQLPME, and ORACLE 6. You must also run a third program, SQLDBA. Configuration details are the same as for remote ORACLE. Local ORACLE *must* be run in standard mode rather than in enhanced mode.

When you choose an ORACLE connection, the Get Connection dialog box appears with the following options:

ORACLE connection issues

The following information may be helpful in setting up your ORACLE connection to ReportSmith.

Symptom	Solution
ReportSmith displays Error #3112.	You may have insufficient conventional memory (memory below 1 megabyte) when you try to connect to ORACLE. To check available memory, type <code>mem</code> at the DOS prompt. There should be at least 400K contiguous memory available. If there isn't, try the following suggestions to free up DOS memory: - Remove unnecessary TSRs. - Close any Windows applications that are running concurrently. - Use a video mode with smaller resolution. - Run the Windows program NETINIT.EXE.
You are unable to connect locally to ORACLE.	Select About Program Manager. You'll see the About Program Manager dialog box, where it says whether you're running in standard or enhanced mode. If you are running in enhanced mode, try switching to standard mode. To do this, exit ReportSmith and exit Windows. At the C prompt, type <code>WIN /S</code> and press the <i>Enter</i> key.
ReportSmith displays Error #1547, "Failed to allocate extent of size <i>n</i> in table space <i>table space</i> ."	You have reached an allocation limit in the ORACLE table space you are accessing. Consult your ORACLE documentation or call ORACLE Technical Support.
ReportSmith displays error #ORA-3121.	ORACLE does not recognize the server specification you have entered. First, ensure that you have spelled the server specification correctly. Check the CONFIG.ORA file or the ORACLE.INI file to see what the default specification is for the ORACLE server. You can also try leaving the server blank in the Get Connection dialog box when connecting. ReportSmith will look in the configuration and use the default server specification.

Symptom	Solution
ReportSmith displays other ORA-XXXX errors.	Look up the error message in the ORACLE manual or call ORACLE Technical Support.
ReportSmith displays error #9141.	Make sure ORAWIN\BIN is in your AUTOEXEC.BAT path. Make sure the following two statements are in \WINDOWS\ORACLE.INI. <i>If NLN:</i> LOCAL=X: YOUR SERVER NAME REMOTE=X: YOUR SERVER NAME <i>If TCP/IP:</i> LOCAL=T: YOUR SERVER NAME REMOTE=T: YOUR SERVER NAME

SQL Server/Sybase

ReportSmith provides a native API connection to SQL Server/Sybase. (For versions, see the README.TXT.) This is the connection type we recommend.

Hardware and software requirements

If you're using SQL Server/Sybase running on an OS/2 server on a LAN, you need:

- DBNMP3.DLL, shipped with SQL Server/Sybase
- W3DBLIB.DLL, shipped with SQL Server/Sybase
- NETAPI.DLL, available from your LAN vendor
- If you're running SQL Server/Sybase under UNIX, a DEC VAX or Hewlett-Packard, you need W3DBLIB.DLL using TCP/IP, DECNET or the necessary files for your network protocol.

The Get Connection dialog box that appears when you select a SQL Server connection as shown here.

The screenshot shows a standard Windows-style dialog box titled "Get Connection Data". Inside the dialog, the "Type" is set to "SQL Server". There are three input fields: "Server:" containing "Server_Name", "User id:" containing "UserID", and "Password:" containing "*****". At the bottom of the dialog, there are three buttons: "OK", "Cancel", and "Help".

SQL Server/Sybase connection issues

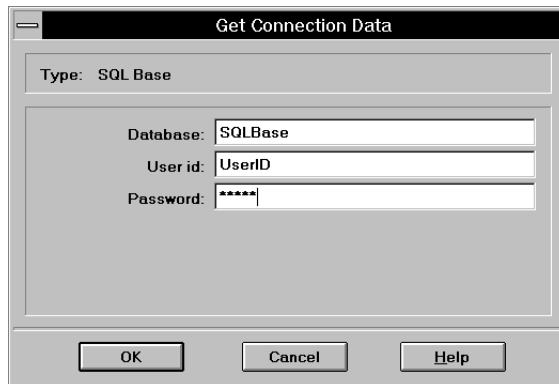
The following information may be helpful in setting up your SQL Server/Sybase connection to ReportSmith.

Symptom	Solution
You have trouble connecting to SQL Server/Sybase.	Make sure W3DBLIB.DLL is in your path. To do this, open Windows File Manager and select the Search command on the File menu. Search for W3DBLIB.DLL (and DBNMP3.DLL) in the root directory. Make sure you select the Search All Subdirectories option. When Windows finds these files, write down their size, date, and location and call ReportSmith or Sybase\Microsoft Technical Support.

SQLBase

ReportSmith provides a native API connection to SQLBase version 4.0 or Quest 1.0 to access SQLBase databases.

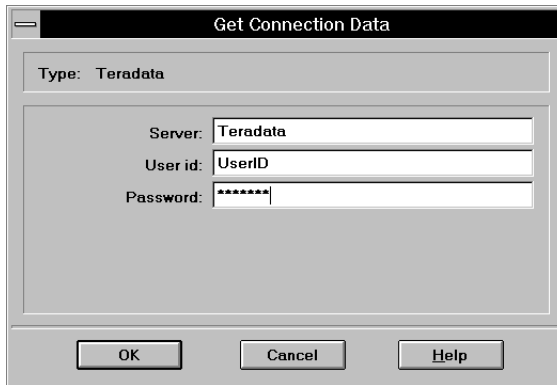
The Get Connection Data dialog box that appears when you select a SQLBase connection is shown here.

A screenshot of a Windows-style dialog box titled "Get Connection Data". The dialog has a light gray background and a black title bar. Inside, there is a section labeled "Type: SQL Base". Below this, there are three text input fields: "Database:" with the text "SQLBase", "User id:" with the text "UserID", and "Password:" with the text "*****". At the bottom of the dialog, there are three buttons: "OK", "Cancel", and "Help".

Teradata

ReportSmith provides a connection to Teradata version 2.0 running Teradata DBC/1012 version 4.1.1 or greater via the Micro Decisionware Inc. (MDI) Gateway.

The Connection dialog box that appears when you select a Teradata connection is shown here.



The image shows a Windows-style dialog box titled "Get Connection Data". At the top, there is a label "Type: Teradata". Below this, there are three input fields: "Server:" with the text "Teradata", "User id:" with the text "UserID", and "Password:" with a masked password "*****". At the bottom of the dialog box, there are three buttons: "OK", "Cancel", and "Help".

Unify

ReportSmith provides a connection to Unify 2000. To connect to Unify please contact Borland's Customer Service department to receive the Unify driver.

Types of reports you can create

ReportSmith lets you create system or custom report types and styles. Each time you create a new report, choose a default report type: columnar, crosstab, form, or label.

You can create many custom report types, such as a master/detail report. This chapter shows you default report types and explains the uses for each. It also shows you two custom columnar report types: summary-only and master/detail.

Default report types

Columnar

A *columnar* report displays data across the report page in columns.

<u>CUSTOMER REPORT</u>			
<u>CUSTOMER</u>	<u>LAST NAME</u>	<u>FIRST NAME</u>	<u>PHONE</u>
Captain Nemo Videos	Yates	Stephen	415/303-2332
Charlene's Films	Williams	Deborah	415/907-9960
Masterpeice Movies	Weinschreider	Eric	415/316-8843
HappyTime Videos	McLeod	Albert	415/310-6472
Video Palace	Krasnoff	John	415/918-5355
Mary Rents Flicks	Klein	Andrew	415/924-3010
Palace of Videos	Van Allen	Sarah	415/922-2983
Zappa Rents	Christopher	Henry	415/929-0649
The Video Company	Timmins	Mark	415/337-9522

Crosstab

A *crosstab* report displays data in a summary or spreadsheet format.

Investment Market Value						
SECTOR	TYPE	Apogee Management		Global Financial		Total
		Growth	Income	Growth	Income	
		(in thousands)				
Energy	Domestic	\$245,790	\$304,924	\$204,343	\$875,293	\$1,630,350
	Foreign	\$104,708	\$0	\$124,756	\$0	\$229,464
Finance	Domestic	\$256,093	\$124,094	\$695,935	\$254,099	\$1,330,215
	Foreign	\$0	\$0	\$64,329	\$0	\$64,329
Manufacturing	Domestic	\$245,766	\$0	\$0	\$489,092	\$734,798
	Foreign	\$354,094	\$105,394	\$876,099	\$0	\$1,335,587
Technology	Domestic	\$179,457	\$0	\$345,092	\$0	\$524,549
	Foreign	\$145,979	\$0	\$201,324	\$101,304	\$448,607
Utilities	Domestic	\$0	\$354,684	\$102,305	\$405,935	\$862,924
	Foreign	\$0	\$0	\$0	\$0	\$0
Total		\$1,531,887	\$889,096	\$2,614,184	\$2,125,657	\$7,160,823

Form

A *form* report displays data in free-form across the report page. Each record always starts on a new page, so there is just one record per page. If the record data is large, a single record can span multiple pages.

<u>CUSTOMER REPORT</u>		1
<u>CSTMN_NAME</u>	Videos R Us	
<u>ADDRESS</u>	648 Able Ave	
<u>CITY</u>	Palo Alto	
<u>STATE</u>	CA	
<u>ZIP</u>	94351	

Labels

A *label* report displays data in label format. ReportSmith can automatically generate all types of Avery label styles for you.

Sherman Zoellner 648 Able Ave Palo Alto CA 94351	Harry Devos 3555 Ball Ln Los Altos CA 94352
Terry Hoover 43360 Elligiac Alley Atherton CA 94355	Peter Wise 271 Fern Dell Menlo Park CA 94356
Pedro Allen 8 Island St Mtn View CA 94359	Eric Weinschre 832 Jennifer Ct San Carlos CA 94360

Custom report types

You can create numerous types of custom reports to suit your needs. The following examples show you two kinds of custom reports you can create with ReportSmith: a master/detail report and a summary-only report.

Master/detail

You can create a report using more than one query, called a master/detail report. Generally, the master report contains one record per key value, while the detail report contains many records per key value.



Customer Holdings & Transactions



Jennifer Davis

Account Number1023495
SOCIAL SECURITY #:405309771

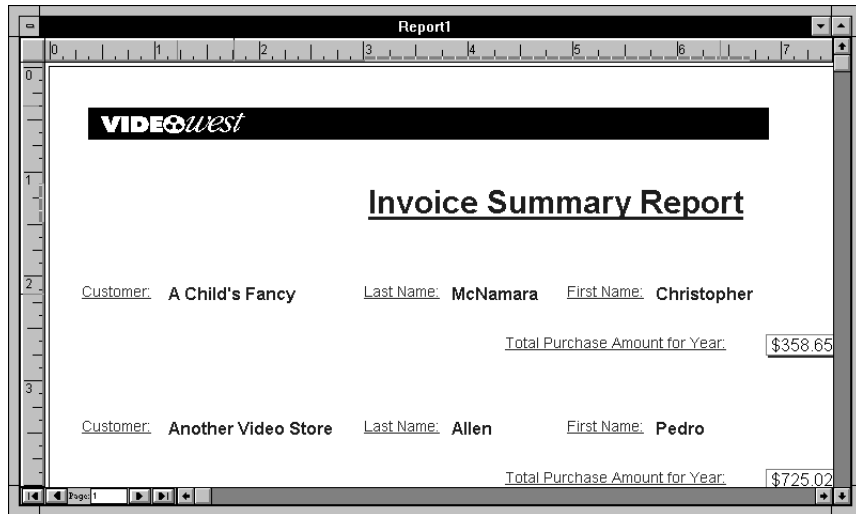
SYMBOL	SHARES	PUR PRICE	PUR DATE	COST	MARKET VALUE
VG	2,000	\$20.00	4/2/93	\$40,000.00	\$50,714.00
HGT	1,000	\$30.00	1/20/93	\$30,000.00	\$38,035.50
CIN	1,500	\$38.00	9/9/93	\$57,000.00	\$72,267.45
EMC	500	\$30.00	2/1/90	\$15,000.00	\$19,017.75
Total Holdings					\$180,034.70

SYMBOL	SHARES	MARKET PRICE	DATE	TRANSACTION TYPE	TRANSACTION TOTAL
CIN	200	(\$5.00)	5/5/90	BUY	(\$1,000.00)
VG	500	\$30.00	10/8/90	SELL	\$15,000.00
Total Transactions					\$14,000.00

A master/detail report is a multiple query report with an A to B, A to C, and so on relationship. The report breaks data into groups so that for each record in the A table, matching records in the B database print beneath it, followed by matching records in the C database. Each query in the report can come from a different data source.

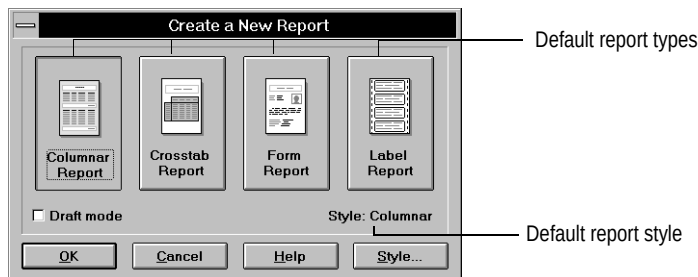
Summary-only

A *summary-only* report presents only summarized values, omitting the details used to arrive at those values. For example, in a report summarizing the number of videos sold by each video store, the total amount of sales for each video store appears, but not the individual orders comprising the total.



Choosing a report type

Each time you create a report, ReportSmith prompts you to choose a report type with the Create A New Report dialog box:



Press a report type button to create a columnar, crosstab, form, or label report.

The default formatting style assigned to the report type you select appears in the dialog box beside Style.

To choose a report type:

- 1 Choose File | New or press the New Report button on the toolbar. The Create A New Report dialog box appears.

- 2 Choose the report type you'd like to create—columnar, crosstab, form or label.
The default style attached to the report type appears beside Style in the dialog box. If desired, choose Style to select a different style and press OK.
- 3 Press OK to exit the Create A New Report dialog box.
You are ready to build your report.

Bringing data into a report

ReportSmith provides Report Query dialog boxes which you can use to add tables and build the report query. A *report query* is a request for specific data from your tables. It is a question you ask about the data, such as “Which employees sold above their quota this month?” ReportSmith uses the criteria to select just the records you specify, using current data from the tables where the data is stored.

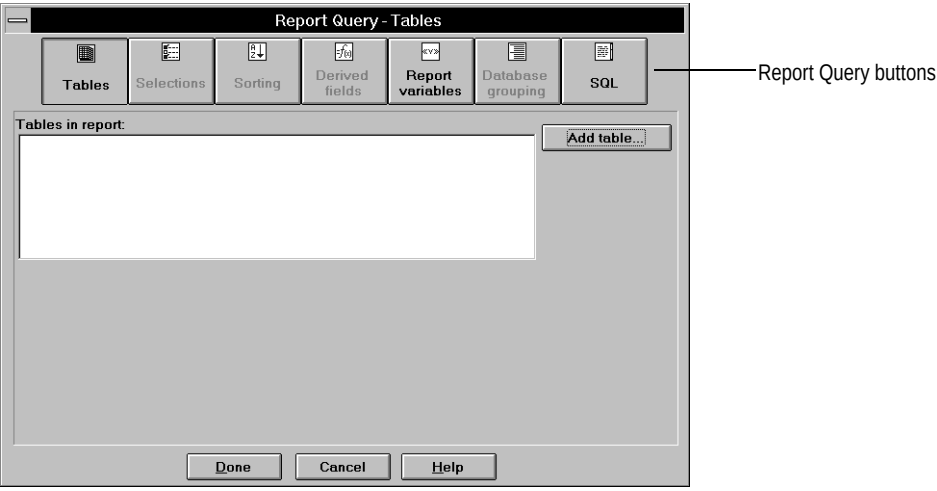
To bring data into a report, you can use either these dialog boxes, or if you’re familiar with Structured Query Language (SQL), you can use direct SQL entry. This chapter describes both methods. After reading it, you’ll know how to

- Use the Report Query dialog boxes to
 - Add tables
 - Link tables
 - Choose columns
 - Assign column aliases
 - Specify selection criteria
 - Group data
- Use Direct SQL Entry

Using Report Query dialog boxes

Report Query dialog boxes let you build an entire query *before* bringing the data into your workplace. It lets you specify selections and other parts of a query without waiting for the data to load into your workplace.

There are seven Report Query dialog boxes. Press a button at the top of the dialog boxes that corresponds to the action you want to perform: Tables, Selections, Sorting, Derived Fields, Database Grouping, Report Variables, or SQL.



Choose a button to display its corresponding dialog box. Buttons are described in the following table. You can navigate between dialog boxes to change your report query in any order; ReportSmith automatically updates it for you.

Button	Function	Button	Function
	Add, remove, and replace tables and columns of data in a report and create links between tables.		Create and place data calculated from other columns of data into your report, either through SQL or the ReportBasic macro language.
	Create, edit, and delete selection criteria.		Create, edit, and delete report variables in your report. With report variables, you can customize a report by substituting different values anywhere in your report each time you run that report.
	Group data and apply selection criteria before loading it into a report.		Add, remove, reorder the data to be sorted.
	View and edit the SQL statement that ReportSmith has built for you in the current report.		

Adding tables

After you've chosen a report type, add one or more tables to your report.

To add tables:

- 1 Press the Tables button.
- 2 In the Tables dialog box, choose Add Table.
- 3 In the Select Table To Be Added dialog box, navigate to the drive and directory where your tables are located. If you've defined a connection you'd like to use, choose it from the Connections list box. Select a table and choose OK.

After you add a table, table selection options appear in the Tables dialog box.

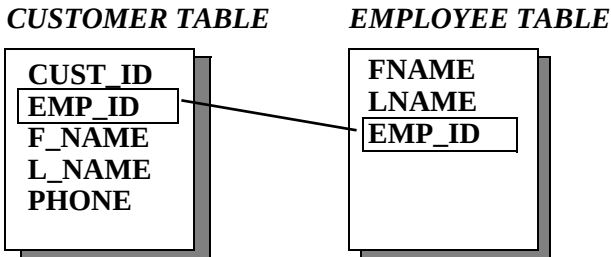
- 4 To add tables, choose Add Table and repeat the above steps. Do this for each table you want to add.

When you've chosen the tables for your report, you can link them.

Joining tables

If you have more than table in your report, you need to link them using related fields. The two fields must contain matching data in related records. Often, common fields share the same name.

In the example below, the CUSTOMER and EMPLOYEE tables share the EMP_ID field in common.



ReportSmith offers you two methods for joining tables:

- Visual joins
- The Create New Table Links dialog box

This section describes both methods.

Visual joins

Use this method to link two tables by tiling two reports, clicking on the common column, and pressing the Merge button.

- 1 Open two reports.



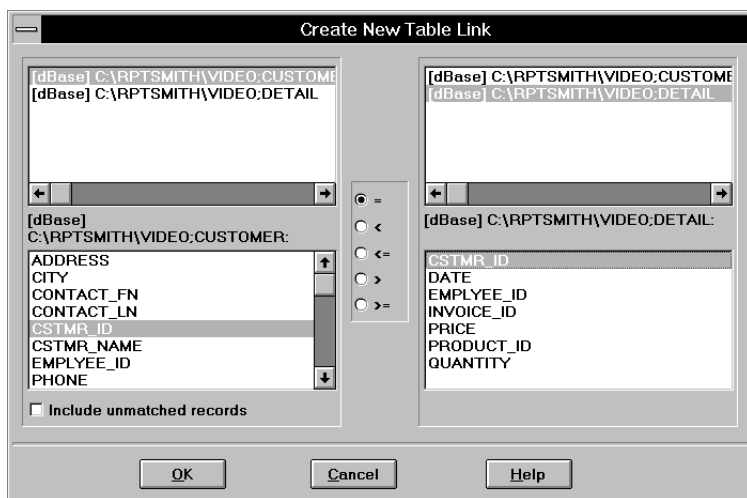
To join reports, you must have two reports open.

- 2 Choose Window | Tile to display both reports on screen at once.
- 3 Click to select a column that both tables share in common.
- 4 Press the Merge button. Data from the inactive report merges into the data on the active report.
- 5 If you no longer need the data from the inactive report, close it. When ReportSmith asks if you want to save the report, choose No.

Using the Create New Table Link dialog box

To merge tables *before* loading them into your report:

- 1 Add tables to your report using the Tables dialog box. In the dialog box, choose Add New Link.
- 2 In the Create New Table Link dialog box, choose the tables and their common field. In the example below, the common field for the CUSTOMER and DETAIL tables is CSTMR_ID.



- 3 Choose OK.

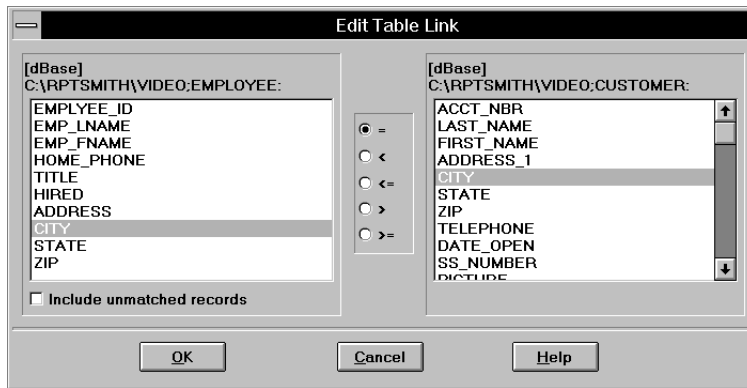


Each table in your report must be included in at least one join. A good rule of thumb is to make sure the number of links you have equals at least the number of tables in your report, minus one.

To modify an existing link:

- 1 Use the Edit Table Link dialog box: in the Tables dialog box, select the link you want to modify and choose Edit Link. The Edit Links dialog box appears.

2 Select a common field and choose OK.

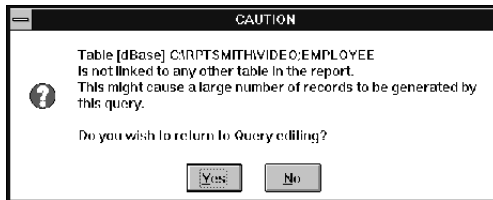


ReportSmith replaces the original link with its replacement.

To delete a join:

- In the Tables Links box, select a link and choose Remove Link. ReportSmith removes the selected link and updates your report.

Note If you try to create a report with more than one table without linking tables, ReportSmith alerts you with the following warning:



If you see this message, you have not created valid links between tables. If you press No, ReportSmith creates a Cartesian product of the tables, matching each record in the first table to each record in the second table. This results in a very large set of records.

You may not want to specify a table link if you are creating a more complicated join between tables in the selection criteria. If so, use the Report Query dialog boxes to create the table link in the selection *before* loading data into the report.

If you do not have system resources to generate such a large table, press the Yes button and create a valid link between tables in your report.

Choosing columns

You can define specific columns from tables you've chosen to appear in your report. ReportSmith offers you two ways to choose columns. You can

- Use the Table Columns dialog box to exclude columns *before* loading them into your report.

- Select them visually and delete them from the report surface.

Choose the method that best suits your needs.

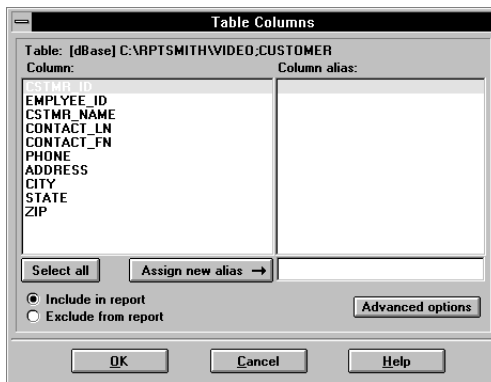
Note ReportSmith generally includes all columns in a table by default. If you want ReportSmith to *exclude* all columns in a table by default, edit the `Options` section of the `RPTSMITH.INI` file so that `ExcludeFields=1`.

Using the Table Columns dialog box

If you're working with a large data set, you can use this dialog box to save time and system resources. This method excludes columns from the report definition. If you exclude columns here, you cannot refer to them anywhere in the report.

Note Make sure to include the columns needed to create table links.

- 1 In the Tables dialog box, select a table and press the Table Columns button. The Table Columns dialog box appears listing columns in the selected table.



- 2 Select the column or columns to exclude or include in your report:
 - Select a single column and choose Exclude from Report or Include in Report. Do this for each column you want. If you don't select a column, ReportSmith loads it into the report by default.
 - To select a block of columns, click in a column and *Shift*+click another column to select those columns as well as the columns in between them. Choose Exclude from Report or Include In Report.
 - To select more than one column, click in a column, then *Ctrl*+click in each additional column you want to add. Press the Exclude from Report or Include in Report buttons.
 - Choose Select all and then include or exclude all columns from a report. This button lets you select all of the table column, so that you can then apply options to them (possibly after deselecting those you want to exclude).
- 3 Choose Advanced options to include a column for its value only, or to use it only for query. For example, you can join two tables by columns you may not want in your report. Just select your column from the list and choose Include Only for Use in Query.

The following table describes the advanced options:

Option	Function
Include in report	Include data in report and query. Choose a column or columns in the column list and click on this option to completely exclude the column(s) from any use whatsoever in the report. The column's data will not be downloaded, and the column(s) will not be available for use anywhere in the report's query.
Exclude from report	Exclude data from report and query. Choose a column or columns in the column list and click on this option to completely exclude the column(s) from any use whatsoever in the report. The column's data will not be downloaded, and the column(s) will not be available for use anywhere in the report's query.
Include column value only	Choose a column or columns in the column list and click on this option to download the column's data into your report for display purposes, but to exclude the column from use in the query. Although the column(s) cannot be used in the query, they can be used to link master/detail reports.
Include only for use in query	Data does not appear on the report, but can be used in the query. Choose a column or columns in the column list and click on this option to include the column for use in the query, but not to download the column's data into the report. If this option is chosen, the column cannot be used to link master/detail reports, nor can its value be displayed in the report.

You can change column selection at any time by opening the Table Columns dialog box and choosing columns to include or exclude from your report once again. ReportSmith re-loads data with the columns you specify, updating your report.



Double-click on a column to scroll through the value options at the bottom of the dialog box.

4 Press OK. ReportSmith loads the specified columns into your report.

Visually

- 1 On your report, choose a column and press *Delete*.
- 2 To delete more than one column, use the following methods:
 - *Ctrl*+click one or more columns to select columns that are not contiguous and press *Delete*.
 - *Shift*+click to select a block of columns and press *Delete*.

Note

Using this method only removes the columns from displaying on the report. The columns you delete still exist in the report definition and can be replaced in the report at any time.

Assigning column aliases

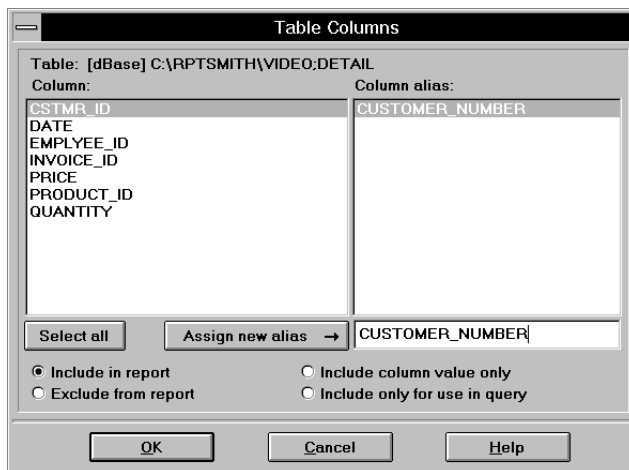
You can assign an alias to database fields for a report. Column aliases are useful when the field name for the data is a code or abbreviation and you want to assign a more

descriptive name to the data. In a dialog box, you'll see the alias name, rather than the original name as it appears in the database.

Column headings use the original field name from the table by default. If you've assigned an alias to a column name, it appears instead. To change the column heading single-click the label, then single-click within the selected heading to use the text cursor to edit its name. Save your changes and ReportSmith displays the field label as entered each time you run the report.

To assign a column alias:

- 1 In the Tables dialog box, select the table that holds the field you want to assign an alias.
- 2 Press the Table Columns button.
- 3 In the Table Columns dialog box, select a column from the Column list.
- 4 Enter the new name for the column in the text box and choose Assign New Alias. ReportSmith places the alias in the Column Alias list box.



- 5 Choose a value option at the bottom of the dialog box for each column. Press Advanced to view the Include column value only and Include only for use in query options. (See page 43 for a description of these options.)

Using direct SQL entry

If you are knowledgeable about Structured Query Language (SQL), you can build a query directly in the SQL dialog box. You can use this method to type SQL statements directly into the SQL text boxes. Alternatively, you can use the list boxes to drag and drop a SQL statement into the SQL text.

Note The SQL text box is recommended for people with comprehensive knowledge of SQL. If you are unfamiliar with SQL, use the Report Query dialog boxes and let ReportSmith build the SQL statement “behind the scenes” for you.

What is SQL?

SQL is a language used by ReportSmith to communicate with your databases to retrieve data. Usually, ReportSmith builds SQL statements behind the scenes for you based on selections you've made in dialog boxes. In general, you aren't required to possess an extensive knowledge about SQL to use ReportSmith. In some situations, ReportSmith does allow you to enter portions of the SQL text. For example, when you use one of two methods to create selection formulas (Selections command) and when you create SQL-defined derived fields (Derived Fields command). ReportSmith combines the formulas you entered with the remainder of the SQL statement that it generates for you.

After you create your own SQL statement or modify an SQL statement, you must continue to change the query using the SQL method. The Tables, Sorting and Selections portions of the Report Query dialog box are no longer available to you for this report. However, you can still use all formatting and macro dialog boxes.

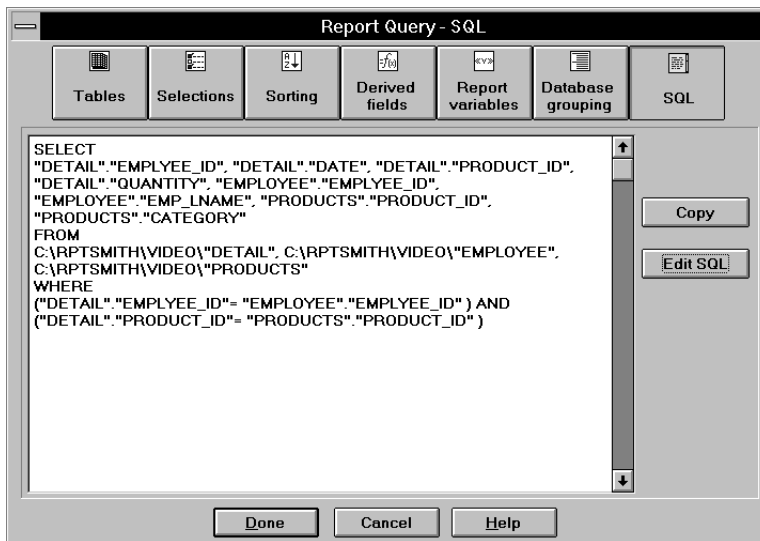
There may be situations where you want to enter your own SQL statement for a report. If you are familiar with SQL you might want to enter the statement yourself for the following reasons:

- You want to use a feature of the SQL language particular to your database that ReportSmith does not provide through its interface.
- You are a SQL expert and want to quickly enter SQL statements without utilizing the ReportSmith interface.
- You want to paste SQL text from another source, such as another program.

Viewing SQL text

- Open a report and do one of the following:
 - Press the SQL button on the toolbar
 - Choose the SQL Text command on the Tools menu
 - From a Report Query dialog box, press the SQL button

The SQL dialog box appears displaying the SQL text for the report:



Here, you can examine the SQL text for clarification, documentation, or debugging purposes. You can copy this SQL text to the Windows Clipboard for use by another Windows program using commands on the Edit menu.

Converting a report to SQL text entry

You can switch from ReportSmith mode to SQL text entry mode in an existing report. It can save you time and effort to develop as much of the report as possible through the ReportSmith dialog boxes first, then use SQL text entry. This allows ReportSmith to generate as much of the SQL text for you as possible. You can then edit the SQL to add in the special portions of the statement you need.

To convert a report to SQL text entry mode:

- 1 After creating a report using the ReportSmith dialog boxes, do one of the following:
 - Press the SQL button on the toolbar
 - Press the SQL button in a Report Query dialog box
 - Select Tools | SQL Text

The SQL dialog box appears.

- 2 Press the Edit SQL button. A warning message alerts you that after you change the report to SQL Entry mode, you cannot return to regular mode. Press Yes to continue. The SQL dialog box appears.

The SQL dialog box is similar to the SQL Selections Criteria and Derived Fields dialog boxes. You can choose elements from the fields, operators, and functions list boxes and insert them into the text box. Alternatively, you can type the text directly into the text box.

- 3 Press the Test button to test your statement. If you enter your formula incorrectly, ReportSmith informs you of the error and directs you to it. If you entered your formula correctly, ReportSmith informs you of the total number of records and displays a success message.
- 4 Press OK to remove the message.
- 5 After editing the SQL text, do one of the following:
 - Press Done to save your changes. ReportSmith uses your text as the report's SQL. The SQL dialog box appears from this point forward when you click the SQL icon or select the SQL Text command on the Tools menu.
 - Press the Cancel button to cancel your changes.

ReportSmith lets you create new reports in SQL text entry mode. If you use the SQL dialog box, you can place the appropriate list box items into a SQL formula to specify the SQL statement text, define macro derived fields, and create report variables for the new report.

Report Query - SQL (dBase (ODBC))

Tables Selections Sorting Derived fields Report variables Database grouping SQL

Table & column browser: Comparison operators: String functions:

Clear list
Add table to list...

x=y Equal
x<y Not equal
x>y Greater than
x=y Greater than or equal
x<y Less than
x<=y Less than or equal
x IS NULL

CONCAT(string, string) - Concatenation
LEFT(string, count) - Leftmost characters
LTRIM(string) - Remove leading spaces
LENGTH(string) - Number of characters
LOCATE(string1, string2) - Find position of string1 in string2
LCASE(string) - Convert to lowercase

Insert Test Clear Cut Copy Paste Undo Zoom Connection

SELECT
"DETAIL","EMPLOYEE_ID","DETAIL","DATE","DETAIL","PRODUCT_ID",
"DETAIL","QUANTITY","EMPLOYEE_ID","EMPLOYEE","EMP_LNAME",
"PRODUCTS","PRODUCT_ID","PRODUCTS","CATEGORY"
FROM
C:\RPTSMITH\VIDEO\DETAIL, C:\RPTSMITH\VIDEO\EMPLOYEE",
C:\RPTSMITH\VIDEO\PRODUCTS"
WHERE

Done Cancel Help

The Data Fields list box contains a list of the fields retrieved from the database. When you convert a report to SQL text entry, this list won't be known until the SQL statement

is executed. For a report already in SQL text entry mode, the list shows the fields selected the last time the SQL statement executed.

To create a report using direct SQL entry:

- 1 Choose File | New or press the New Report button on the toolbar.
- 2 Press the button that corresponds to the type of report you'd like to create and press OK to view the Tables dialog box.
- 3 Press the SQL button.
- 4 Press the Edit SQL button. ReportSmith displays a warning message:

Once the report is changed to direct SQL entry, it cannot be changed back to regular mode. Do you wish to proceed?

- 5 Press Yes in response to the warning message. ReportSmith displays the SQL dialog box.

You'll use this dialog box to write your own SQL statement instead of stepping through the Report Query dialog boxes while ReportSmith builds it for you. There are two ways to enter the SQL statement:

- Type the statement directly into the SQL text box.
- Place items from the list boxes into the text area box.

- 6 Press the Connection button and choose a data source in the Get Connection dialog box. Press OK.
- 7 To enter your formula, do one of the following:
 - Place the cursor in the SQL text area box and type it directly into the SQL text box.
 - Use the Table and Column Browser to view fields in one or more tables. Place fields, comparison operators, and functions into the SQL text area by dragging and dropping them or by double-clicking on them.



To clear your formula, press the Clear button. When you press it, ReportSmith displays:

Erase the current SQL text?

Press Yes or No.

Using the Table and Column Browser

If you need assistance selecting fields and tables when writing an SQL statement, use the Table and Column Browser. With it, you can select tables you're working with and view them in the Browser while creating your formula. The Browser lets you drag and drop these items into your select statement, eliminating extra keystrokes. The Browser is also helpful when you're working with several tables, eliminating the need to keep track of field and table names.

To add tables and columns to the Browser:

- 1 In the SQL dialog box, press the down arrow in the left-most list box and click to select Table and Column Browser. A list of options appears.

- 2 Double-click Add Table to List. The Select Table To Add To Browser dialog box appears.
- 3 Select a table to place in the Browser and press OK. Do this for each table you want to view. Use the Type, Drives and Directories list boxes to navigate to the location of a table or press the Server Connect button to connect to the appropriate data source.
- 4 Press OK. ReportSmith displays corresponding fields under each table you select.
- 5 Place fields from the Browser into the SQL text area.
Press the Zoom button to enlarge your view of the SQL text area.
- 6 Sometimes you have too many tables in the Browser. To clear the list, choose the Clear List option in the Browser list box and double-click on it.
- 7 You may need to press the Connection button to tell ReportSmith which table you are looking at here, if you haven't already. Use the Test button to test your formula.



You can write an SQL statement without using the Browser, then press the Connection button. Alternatively, you can connect to a table in the Browser before writing an SQL statement.

Using the SQL list boxes

You can control what is displayed in the dialog box with the SQL list boxes.

Report Variables: Table & column browser:	Comparison operators: Patterns, ranges and lists: Grouping & logical operators: Arithmetic operators: Constants: SQL statement keywords: Summary functions: Subqueries:	String functions: Numeric functions: Time and date functions: System functions:
---	---	---

- The first list box displays ReportSmith report objects and tables and columns you have selected.
- The second list box displays arithmetic functions and operators and other statements to assist in building your query.
- The third list box displays all SQL functions specific to your data source. Depending on the database to which you are connected, ReportSmith forms this list for you. It contains valid functions for that data source type.

For example, if you are connected to ORACLE, ReportSmith places all valid ORACLE SQL functions here. You don't have to remember any SQL functions.

Restrictions for SQL text entry

After you convert a report to SQL text entry, it cannot return to regular mode. From that point forward, you must modify the SQL statement rather than use the ReportSmith interface to affect sorting, selections, tables (including linking columns), and SQL-derived fields for that report.

Although ReportSmith can generate SQL text for you, it can't process or interpret SQL statements. It doesn't work backwards from your SQL statement to figure out what you've added or changed. In fact, since you're entering the SQL statement directly, you

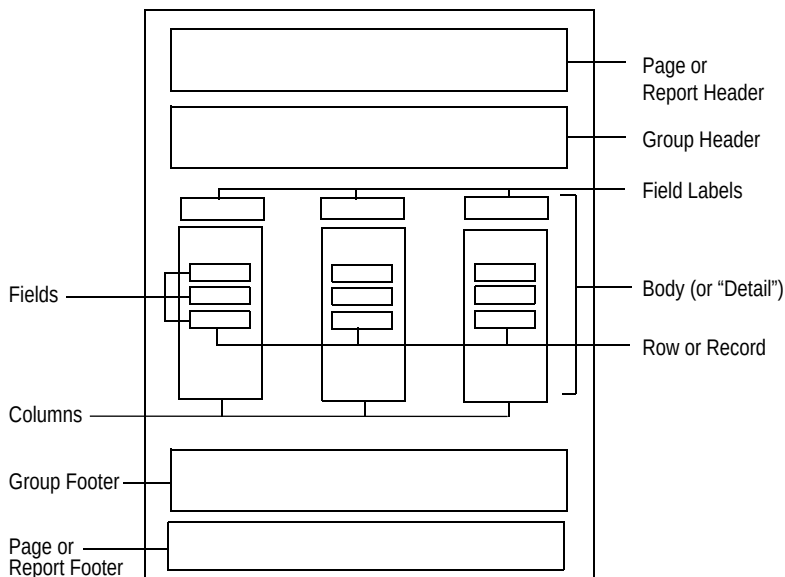
may be manipulating the database in ways that the ReportSmith interface doesn't recognize.

Note You can still use ReportSmith formatting, presentation and macro features even if you use SQL text entry to retrieve the data from the database.

Manipulating data

This chapter describes how to manipulate data in reports with sorting, grouping, creating summary fields, derived fields, report variables, and master/detail reports.

Parts of a report



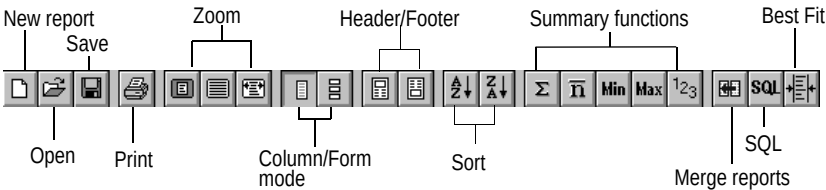
The action you perform on one selected item applies to all items in that category. If you select a group footer and add a border, borders appear in all instances of that group footer.

Note For an illustration that identifies crosstab parts, see “Parts of a crosstab” on page 132.

The toolbar

ReportSmith's toolbar puts frequently used actions at your fingertips with easy-to-use buttons. To see a description of a button, drag the mouse over it and look in the status bar. The toolbar is located at the top of the ReportSmith screen.

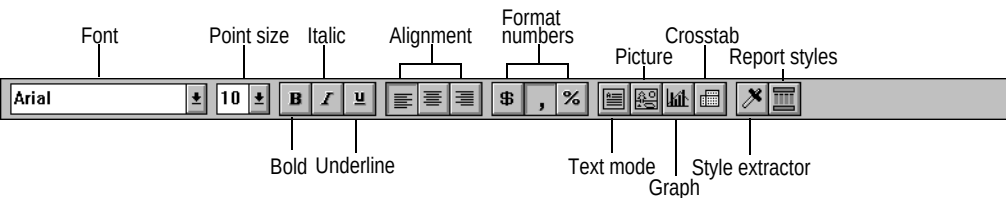
To display or hide the toolbar choose View | Toolbar.



The ribbon

Use the ribbon to quickly apply formatting. For example, you can use it to change the font style and size, to insert number formats, text or graphics, or to insert a graph.

To display or hide the ribbon choose View | Ribbon.



Sorting

You can sort data in a report in ascending or descending order with the Sort buttons on the toolbar. You can also designate sort order with Tools | Sorting. In some cases, ReportSmith sorts data for you, such as when you press a Header or Footer button.

To sort data with the toolbar:

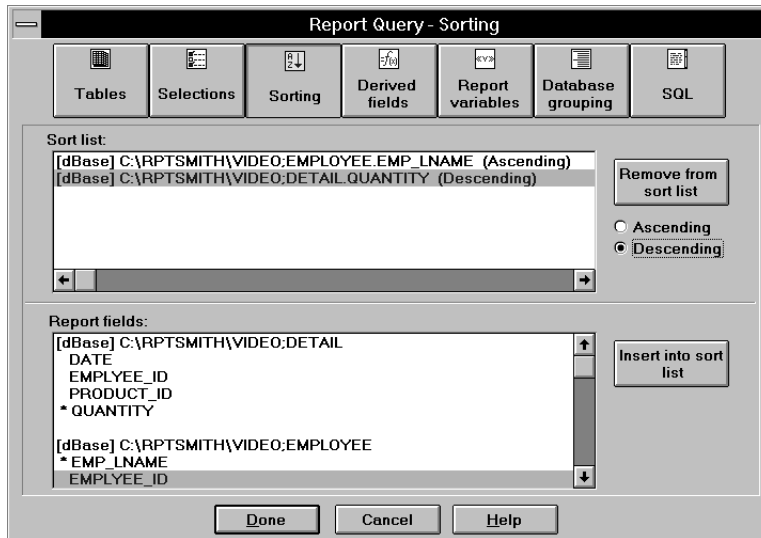
- 1 Select a column in your report you want to sort.
- 2 Choose a Sorting button.

Fields within the selected column sort in ascending or descending order.

To sort data:

- 1 Select Tools | Sorting to see the Sorting dialog box. Or, choose Sorting from a Report Query dialog box.

- 2 Place one or more fields from the Report Fields list box into the Sort list. The selected field appears in the Sort List box. An asterisk appears beside sorted fields.



- 3 Choose Ascending or Descending. Add an additional report field to the Sort list by selecting it in the Report Fields list and pressing the Insert Into Sort List button.
- 4 Choose Done. ReportSmith sorts fields in the order in which they are placed on the sort list.

Grouping

You can break data into groups based on fields you select. ReportSmith lets you group data after you load it into your report using Report Grouping; or, on the server, before loading it into your report with Database Grouping.

You can use both types of grouping in one report.

Report grouping

Use the Report Grouping command to group data after it has been loaded into your report. There are two ways to do this:

- Using the Header or Footer buttons
- Using the Tools | Report Grouping command

This section describes both methods.

To group data with the toolbar:

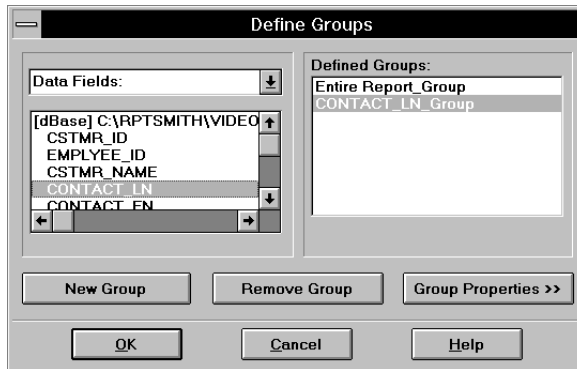
- 1 Click to select a column on your report by which you'd like to group.

- 2 Press the Header or Footer button to sort the data based on the selected field and to insert a header or footer in each group.

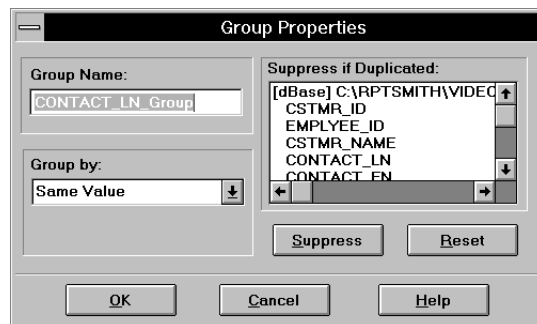
To place fields into the header or footer, see “Placing items into a header or footer” on page 107.

To group data with Tools | Report Grouping:

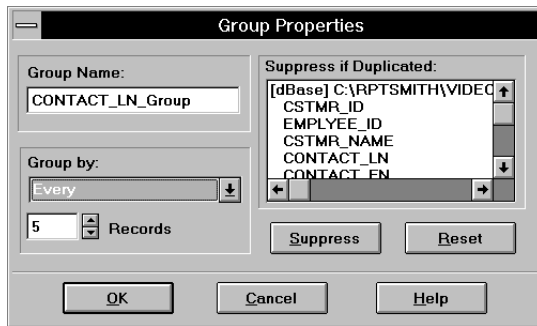
- 1 Choose Tools | Report Grouping to see the Define Groups dialog box.
- 2 Select a field type from the upper left list box. The type of field you select populates the list below it with field names associated with the data type. (The contents of this list depend on the types of fields contained in your report.)
- 3 Select a field and place it into the Defined Groups list. Do this for each field by which you want to group.



- 4 Choose Group Properties to see the Group Properties dialog box. Here, you'll define how you want to group the selected field—if you want ReportSmith to group all like values together, choose Same Value. ReportSmith compares each field with the one preceding it, and if the value is the same, groups the values together. If the values differ, ReportSmith creates a new group. In the example below, we grouped fields by each unique CONTACT_LN.



To group by a specific number of records, choose Every and select a number from the spin box that appears. In the example below we grouped by every 5 records in the CONTACT_LN group.



- 5 You can print each value in a group just once instead of displaying it repeatedly. To do this, select a field you want to suppress and choose Suppress. Do this for each field you want to suppress.

ReportSmith places an asterisk beside suppressed fields. They appear just once per group.

- 6 Press OK after you've set the properties for the defined group and choose Done.

Database grouping

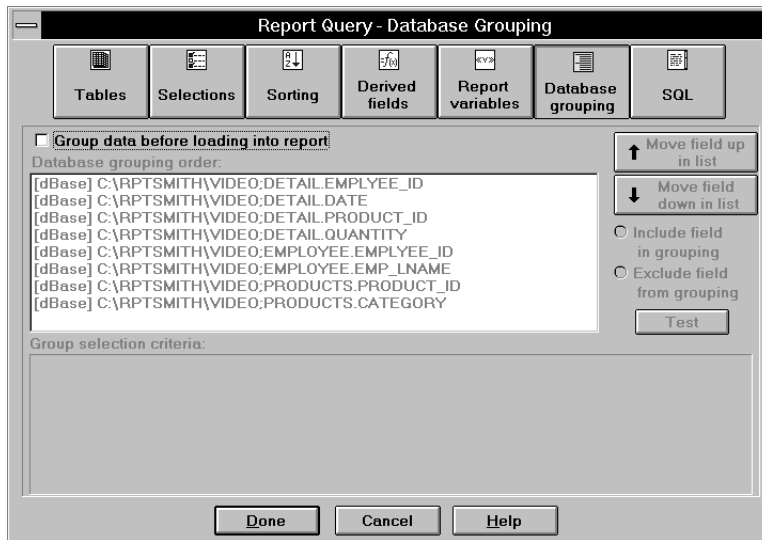
For grouping to occur on the server before loading it into your report, use the Database Grouping feature. For example, suppose you have a database containing information about 10,000 employees from ten departments and you want a report displaying a sum of salaries in each department. To increase performance, you would use Database Grouping to have ReportSmith consolidate the data on the server and then download just the ten records you requested.

If you are going to perform any database grouping, then it is a restriction of SQL that all fields which provide a unique value per row must be grouped, while those that do not should not be grouped.

For example, regular data fields must always be grouped, since there is always a potentially unique value for each record. Derived fields should not be grouped if they are computing an aggregate (summary) value. A derived field of "SUM(SALARY)" should not be grouped, because it generates a summary value for the entire group which does not change for each record in the group. However, a derived field of "SALARY * .25" must be grouped, because it computes a unique value for each record.

To group data on the server:

- 1 Choose Tools | Database Grouping or press the Database Grouping button in a Report Query dialog box. The Database Grouping dialog box appears.



Choose Group Data Before Loading Into Report. A list of fields from your report appears in the Database Grouping Order window, and options in the dialog box become enabled.

The upper portion of the dialog box defines what fields you want to group by. If you are familiar with SQL, this part of the dialog generates a Group By clause. The lower portion of the dialog box lets you select which groups to include in your report and selection criteria. In SQL, it generates a Having clause.

- 2 Select a field you want to group by and use the ordering buttons to position it in the list. The order in which you place fields in the list is hierarchical, moving from highest to lowest grouping order.
- 3 You can choose to include or exclude a field from grouping with the Include Field From Grouping and Exclude Field From Grouping options.

The Exclude Field option is disabled for tables that contain a unique value. This means you must group by every field that contains a unique value per record. Conversely, you can exclude fields that do not contain a unique value per row, such as calculated fields.

- 4 To select groups to include in your report, insert a sentence into the selection criteria in the lower portion of the dialog box and choose an operator (All, Any or None) to apply to the statement.

Note Press Test before loading data into your report. You can find and eliminate any errors in your query before ReportSmith executes the underlying SQL.

Inserting a header or footer

You can create groups based on a selected field and insert a header or footer in each occurrence of that group. ReportSmith offers you two methods for placing headers or footers into your report:

- Header and Footer buttons
- The Insert | Headers/Footers command

When you use the Header or Footer buttons, ReportSmith sorts *and* groups the data based on the field you select, and places a header or footer into each group.

To insert a header or footer without sorting your data, use the Insert | Headers/Footers command.

To sort and group data with the Header and Footer buttons:

- 1 Select the column in your report by whose data you want to group. In the following example, we selected the ACCT_NBR column.

CUSTOMER LIST					1
<i>PageHeader</i>					
ACCT_NBR	LAST_NAME	FIRST_NAME	PHONE		
1024034	White	Cindy	415-948-6547		
2094056	Jones	Arthur	415-941-4321		
1209395	Parker	Debra	916-213-2234		
3094095	Sawyer	Dave	415-948-9998		
1023495	Davis	Jennifer	516-292-3945		

- 2 Choose the Header or Footer button on the toolbar. ReportSmith sorts and groups the data, and inserts a header or footer. In the example below, we inserted a header.

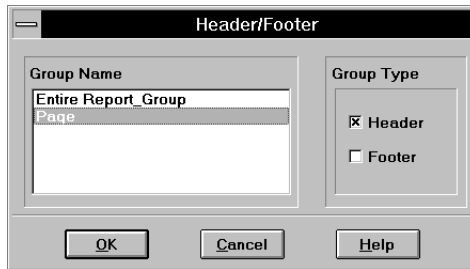
CUSTOMER LIST					1
<i>PageHeader</i>					
<i>GroupHeader ACCT_NBR_Group</i>					
ACCT_NBR	LAST_NAME	FIRST_NAME	PHONE		
1023495	Davis	Jennifer	516-292-3945		
<i>GroupHeader ACCT_NBR_Group</i>					
1024034	White	Cindy	415-948-6547		
<i>GroupHeader ACCT_NBR_Group</i>					
1209395	Parker	Debra	916-213-2234		
<i>GroupHeader ACCT_NBR_Group</i>					
2094056	Jones	Arthur	415-941-4321		
<i>GroupHeader ACCT_NBR_Group</i>					
3094095	Sawyer	Dave	415-948-9998		

Headers

ReportSmith sorts and groups your data based on the field you selected. Additionally, it labels the header. In the above example, the header is called “Group Header ACCT_NBR group.”

To group data using Insert | Headers/Footers:

- Use this method to insert a header or footer into your report without performing sorting.
- 1 Select a column in your report by whose data you want to group.
 - 2 Choose Insert | Headers/Footers to see the Header/Footer dialog box.



- 3 Choose a Group Name and check a Group Type option. Do this for each group you want to create. In the example above, we chose to insert a page header. Press OK to exit the dialog box and see the results of grouping on your report.

Specifying selection criteria

You can choose rows to display in your report by specifying selection criteria. Instead of displaying all of the data in your tables, specifying selection criteria lets you limit the data to display.

ReportSmith includes a record in your data set only if it meets your criteria. You can specify one or multiple criteria for ReportSmith to match your data. In the example below, we have specified multiple criteria: we want to see records where HIRED is greater than or equal to January 1, 1992 and TITLE is equal to “Sales.”

<u>Employee Report</u>				
<u>EMPLOYEE_ID</u>	<u>LAST_NAME</u>	<u>FIRST_NAME</u>	<u>TITLE</u>	<u>HIRED</u>
E1-23	Rimiclaud	Arthur	Selas	2/28/92
E4-62	Bleke	William	Selas	12/31/93
E2-17	Snydcr	Gary	Selcoo	8/3/83
E2-30	Pounc	Ezra	Selas	5/3/83
E3-25	Keats	John	Selas	8/3/92
E2-2	Giroborg	Allen	Selcoo	6/3/92

The selection criteria limits the data set to display specific records, reducing the size of data and making it more specific. To get the data we want, we created the following selection criteria in the Selections dialog box:

☐ Include records in the report where all of the following apply:

- 1 data field EMPLOYEE.HIRED is greater than or equal to date 1/1/92
- 2 data field EMPLOYEE.TITLE is equal to text Sales

After we apply the new selection criteria to our data, ReportSmith displays just the data we specified on our report:

<u>Employee Report</u>				
EMPLOYEE_ID	LAST NAME	FIRST NAME	TITLE	HIRED
E1-23	Rimkoud	Arthur	Sales	2/23/92
E3-25	Kests	John	Sales	8/9/92
E2-12	Ginsborg	Allcr	Sales	6/6/92

ReportSmith offers you two ways to specify selection criteria for your reports:

- The Selections dialog box
- The Field Selection Criteria dialog box

This section explains both methods.

Choosing records

You can use ReportSmith's Selections dialog box to specify which records you want to display on your report. ReportSmith lets you use English sentences (as opposed to SQL) to construct the selection criteria, building a basic outline which you can modify. You can also use SQL to create your selection criteria.

The Selections dialog box where you'll build a selection construct is shown below.

Report Query - Selections

Tables

Selections

Sorting

Derived fields

Report variables

Database grouping

SQL

☐ Include records in the report where all of the following apply:

1

☐ Exclude duplicate rows

Click on highlighted or underlined items to see context-sensitive lists

To add selection criteria, click on the number where you want to place a new sentence. (In the dialog box above, you'd click on 1.) ReportSmith displays a list of choices for adding or deleting items:

Select this option	To...
Add Selection Criteria	Place the outline for a new selection criteria sentence into this sentence.
Add SQL Selection Criteria	Display a dialog box in which you'll create selection criteria using SQL.
Add A List	Place a new list at this point.
Create A New List And Move This Item Into It	Add a new list with a basic sentence outline beneath it.

As an example, we chose Add Selection Criteria. ReportSmith places a basic sentence into the dialog box which we'll modify. To do this, click on the section you want to modify to see a drop-down list of choices.

☐ Include records in the report where all of the following apply:

1. data field CUSTOMER.STATE is equal to text CA

- Click on the Exclude Duplicate Rows option to avoid including any rows that contain the same data as another row.

You can identify items which display drop-down lists—they are underlined or highlighted. In the example above, since a basic construct exists, if you click on the "1." again you'll see a context-sensitive list:

☐ Include records in the report where all of the following apply:

1. data field CUSTOMER.STATE is equal to text CA

Insert new selection criteria after this item
 Insert new SQL selection criteria after this item
 Insert a new list after this item
 Create a new list, and move this item into it
 Delete this item
 Convert this item to SQL

As you choose items, ReportSmith alters the outline of the sentence dynamically to ensure that it makes sense. There are numerous combinations of items which can be created to form a sentence. You can specify any number of criteria in any combination, simple or complex, to your report selections.

Here, we'll modify the Include statement at the top of the dialog box, "Include records in the report where all of the following apply: "Click on all to display a list of criteria.

You can choose to include ALL, ANY, NONE or NOT ALL of the criteria you list below it in the dialog box.

Select this option	To...
ANY	Include a record in your report if it meets ANY of the criteria beneath this list. This places an implicit OR between each sentence.
ALL	Include a record in your report only if it meets ALL of the criteria beneath this list. This places an implicit AND between each sentence.

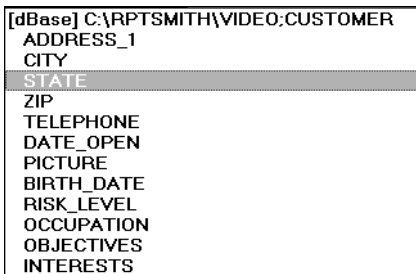
Select this option	To...
NONE	Include a record in your report if it meets NONE of the criteria beneath this list. This places a NOT before each sentence and an AND between each sentence.
NOT ALL	Include a record if it does not meet every criteria listed below this list. It can meet ANY portion of the list criteria, but NOT ALL of them.

We'll select **all** to include a record only if it meets all of the criteria we're about to specify.

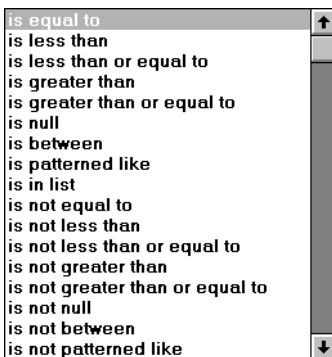
Next, click on **data field** in the basic sentence to see a list of field types to choose from:



If you choose "data field" and click on the next item in the sentence, **DETAIL.INVOICE_ID**, ReportSmith displays a list of data fields. The data field you select is the field against which you'll compare your rows:



Click **is equal to** to choose a comparison operator. The comparison operator is the verb of the sentence:



In our example, we chose **is greater than** because we want to display all sales in which we sold more than a certain amount of items.

Click on the next item in the list to choose to what you are comparing your data field:



In our example, we selected `text`. ReportSmith displays a text entry box in the sentence which you can fill with one or more values:

CA

Enter a value. In our example, we entered `CA` and pressed the *Enter* key to place it into the sentence.

We've completed our first sentence. To enter another sentence to this list, click on **1.** to display the drop-down list and select a function.

This table describes the options in this drop-down list:

Select this option	To...
Insert New Selection Criteria After This Item	Place a sentence below the current one.
Insert New SQL Selection Criteria After This Item	Display the SQL Selection Criteria dialog box in which you can create a criteria using SQL and place it below the current sentence.
Insert A New List After This Item	Place a new Include Statement beneath the current sentence.
Create A New List And Move This Item Into It	Replace the current sentence with an Include Statement and place the sentence in it.
Delete This Item	Delete the sentence. If necessary, renumber the remaining sentences.
Convert This Item To SQL	Replace this English sentence with SQL. After you select this option and click on an item, ReportSmith displays the SQL Selection Criteria dialog box.

ReportSmith adds a sentence to the list and numbers it:

☐ **Include records in the report where all of the following apply:**

1. data field `CUSTOMER.STATE` is equal to text `CA`

2. data field `INVOICES.TOTAL` is greater than number `5`

You can add numerous lists and sentences to create your selection criteria. Just follow the steps above to add, remove and modify additional lists and sentences.

After you've created your selection criteria, press the Test Selections button to see the number of records that it retrieves.

To delete an item or a list:

- 1 To delete a list, select the number corresponding to it.

To delete an item in a sentence, click to select it.

- 2 Press the *Delete* key, or select Delete This Item from the drop-down list.

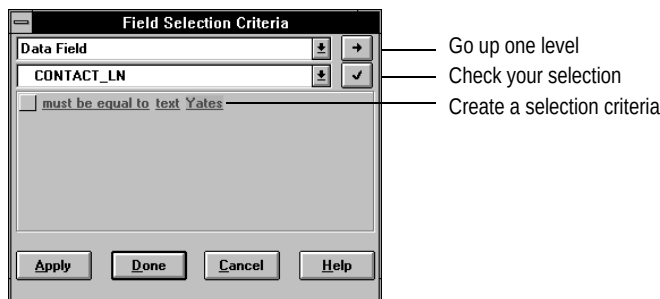
If you selected a list, ReportSmith removes the list and its subcriteria from the construct. If you selected an item, ReportSmith removes that item.

You can use some key combinations while in the Selections dialog box to perform certain functions. The following table describes them.

Key	Function
Arrow keys	Use the up, down, left and right arrow keys to navigate within your SELECT statement.
<i>Enter</i> key	Press the <i>Enter</i> key to insert a criteria or to make a choice in a drop-down list.
<i>Alt+↓</i>	Press this key combination to display a drop-down list.
<i>Esc</i>	Press the <i>Escape</i> key to close a drop-down list without making a choice.

Choosing fields

The Field Selection Criteria dialog box lets you choose fields for your selection statement. To open the dialog box, select a column, right-click the mouse and choose Selection Criteria from the pop-up menu. Alternatively you can choose Tools | Field Selection Criteria.

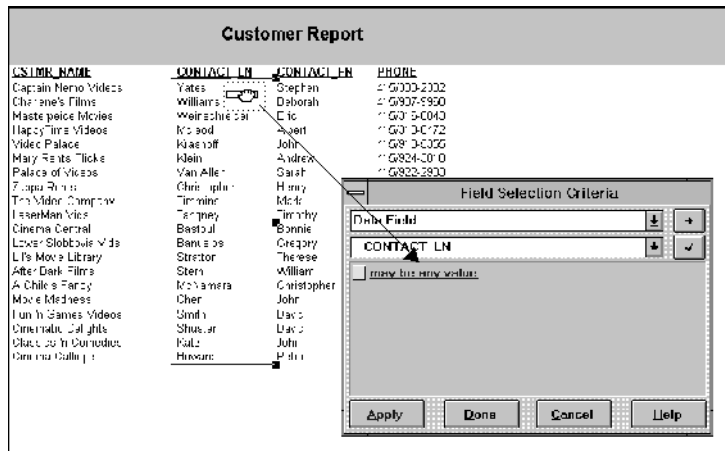


Use the dialog box to create a basic selection criteria for a report from the report surface (instead of navigating through the Report Query dialog boxes). For example, to select only customers with the last name of "Yates," select the CONTACT_LN column, right-click the mouse, choose Selection Criteria and place the selection criteria into the dialog box.



You can drag and drop fields from your report into the selection criteria. To do this, select a field on your report and drag it into the dialog box. ReportSmith creates the selection criteria for you.

In this example, we'll drag "Yates" from the CONTACT_LN column and place it into the dialog box:



ReportSmith places the field into the dialog box and modifies the selection criteria.



You'll see the grab cursor when you drag and drop items from your report and place them into the dialog box. Depending on the selection criteria you create, you'll see one of the following drop cursors:



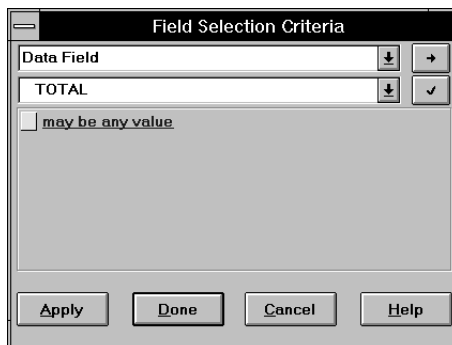
If you drop the field in the current area, you'll add selection criteria.



If you drop the field in the current area, you'll insert the text field you grabbed into the current area.

To change the selection criteria directly on the report surface:

- 1 Open a report and select a column you'd like to modify. As an example, we'll select the TOTAL column.
- 2 Right-click and choose Selection Criteria from the pop-up menu. The Field Selection Criteria dialog box appears displaying a default construct, **may be any value**.



Lets modify the selection construct. As an example we'll modify it to display all transactions greater than or equal to \$100 for certain cities.

- 3 Click on **may be any value** to see a list of options and select **must be greater than or equal to** to place it in the construct.
- 4 ReportSmith has added **number 0** to the construct so you can specify a value. Click on **0** and enter **100** into the value entry box. Choose Apply to execute the selection in your report.

Next we'll specify the cities we want to display on our report.

- 5 At the top of the dialog box, click on the drop-down list displaying TOTAL and select CITY from the list. ReportSmith displays **may be any value** in the construct.
- 6 Click on **may be any value** and select **must be in list** from the drop-down list. ReportSmith inserts the default for the in-list statement, **of values**.
- 7 Click on **<values>** to open the List Editor dialog box:



You'll use the List Editor to create a list of values.

- 8 Enter a city into the text entry box and choose Add to place it in the list. Do this for each value you want to add. After you've added the values, press Test to see how many records match the value currently selected in the List box.



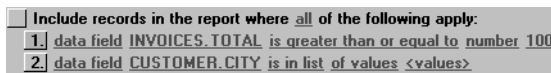
Note If you entered your selection construct incorrectly, ReportSmith returns an error message. If this happens, correct the errors and test until you receive a success message.

- 9 Press OK and press Done to exit. ReportSmith applies the modified selection criteria to your report and regenerates it for you with the specified values.

The changes you make in the Field Selection Criteria dialog box appear in the Selection dialog box. You can modify your construct there also. As an example, let's take a look at the selection criteria we just created in the Selections dialog box.

To view your selection construct in the Selections dialog box:

- 1 Open a report you modified using the Field Selection Criteria dialog box.
- 2 Choose Tools | Selections. The Selections dialog box appears with your modified construct displayed.



To see the list of values associated with this construct, click on <values>. ReportSmith displays the List Editor dialog box with the values you specified.



- 3 Choose OK to exit the dialog box.

Creating a derived field

A *derived field* is a custom field that doesn't exist in your tables. You can create a derived field by combining two or more fields or by applying a calculation or formula to a field. For example, you might create a derived field by combining the EMP_FNAME and EMP_LNAME fields from your table, creating one field (FULLNAME) which contains both first and last customer names.

You can create two types of derived fields with ReportSmith:

- SQL-derived fields
- Macro-derived fields

A typical use of a derived field is to combine two text fields into one field. You might have one field containing employee first names and another containing last names. You can create a derived field that trims unnecessary space around the field name, combining the fields.

Another way to use a derived field is to extract the number of a month from a field that holds dates. The formula for this appears as `MONTH(invoices.DATE)`.

Note This is database-specific; not all databases have this function. Its format or name might differ.

In this section, you'll learn how to create derived fields that

- Calculate a 10% sales commission
- Combine two fields into one field
- Display just the number of the month the sale occurred
- Calculate the percentage of total sales for an employee

SQL-derived fields

To create a SQL-derived field:

In this example, we'll show you how to create a derived field which calculates a 10% sales commission.

- 1 Choose Tools | Derived Fields, or choose Derived Fields in a Report Query dialog box.
- 2 Enter a name into the Derived Field Name box. This name becomes the field label for the derived field if you insert it as a column in your report.

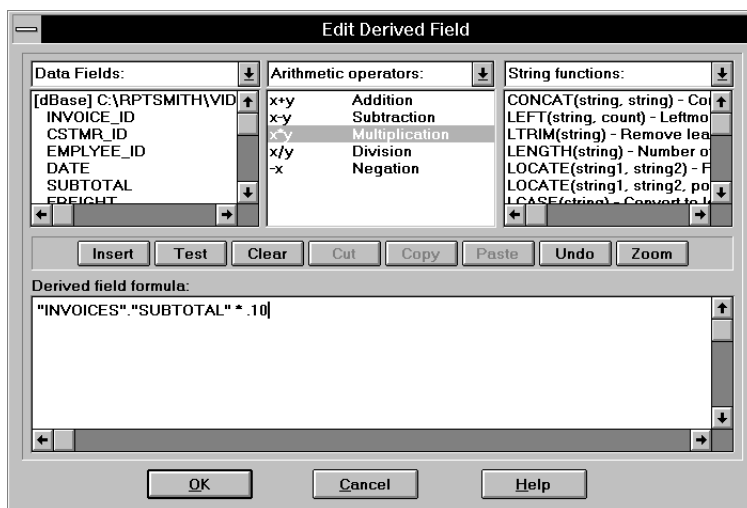
The example formula in these steps calculates the 10% sales commission for the Video West employee, William Blake.

- 3 Turn on the Defined By SQL option.

Note Typically, ReportSmith uses SQL code to get the results of the derived field. This means your database calculates the derived data.

- 4 Choose Add.

ReportSmith displays the Edit Derived Field dialog box where you define the derived field formula.



Choose elements from the field, operator, and function list boxes and place them in the Derived Field Formula window.

The order in which you select from these list boxes depends on what you're trying to achieve; you don't have to choose a field, then an operator, then a function. Use them in any order necessary to build the formula you want to apply to the data.

Note Functions and operators appearing in the list boxes differ depending on the server and database you use. Refer to your server or database documentation for explanations of available functions and operators.

- 5 Press the down arrow beside the field, operator, or function box.

A list appears displaying available types of fields, operators, or functions. For example, if you want to create a derived field based on report fields, select Data Fields from the fields drop-down list. ReportSmith displays fields from the table in your report.

- 6 Select the type of field, function, or operator and place it into your formula.
- 7 Test your formula using the Test button until you receive a success message. Choose OK.
- 8 Use the Insert | Field command to place the derived field in your report.

Macro-derived fields

Typically, ReportSmith uses SQL code to obtain the results of a derived field formula. This means your database calculates the derived data. However, there may be information you need to show in a field which cannot be created by your database. For example, to derive a percent of a total, you need to select the Defined by a ReportBasic

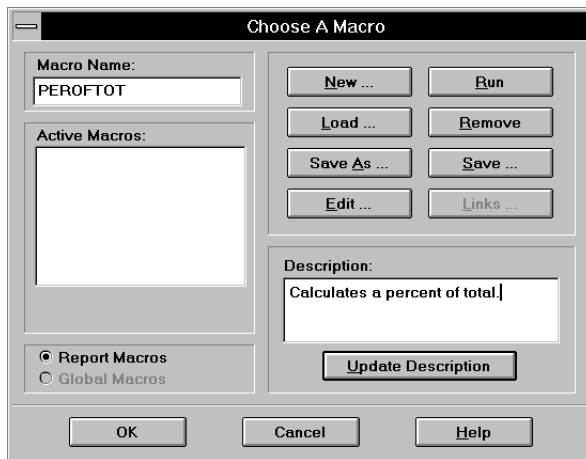
Macro option to create the derived field locally. This is because SQL code can't access summary fields, which are calculated locally.

You can create a derived field that calculates a percent of a total. For example, in our Sales By Employee report, we want to determine what percent of William Blake's sales can be attributed to each of his clients.

Sample macro-derived field

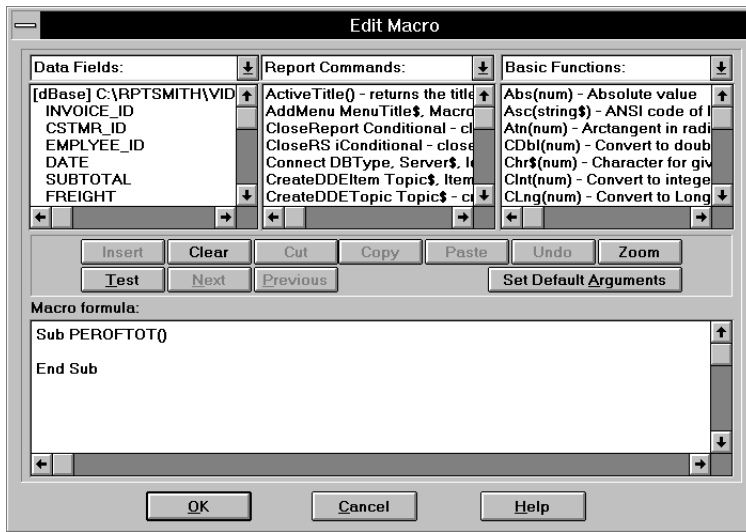
This sample shows you how to create a macro-derived field which calculates a percent of the total.

- 1 Open a report for which you want to calculate the percent of total.
- 2 Choose Tools | Derived Fields, enter a name and select the Defined by a ReportBasic Macro option. Choose Add to see the Choose a Macro dialog box.



- 3 The Report Macros option is enabled. (Derived fields must be based on report macros.) Enter an optional description for the macro function into the Description text box.

4 Choose New to see the Edit Macro dialog box.



ReportSmith has placed the first and last lines of your formula into the formula window. You do not need to re-enter them.

5 Enter the macro code into the Macro formula text box:

```
Sub PerOfTot()
' Calculate the percentage of the total quantity of this employee's sales that the
' current record's quantity of sales represents.
' The Field$ function gets the QUANTITY record as a string.
' The Val() function converts it to a number which is stored in the variable "Quantity."
Quantity = Val(SumField$("SUBTOTAL", 'c:\rptsmith\samples\invoices.dbf',2,"Sum"))
' The Next line gets the total quantity for the current customer group
' and assigns it to a variable.
TotalQuantity = Val(SumField$("SUBTOTAL", "c:\rptsmith\samples\invoices.dbf",1, "Sum"))
' A divide by zero error will halt macro execution.
if TotalQuantity > 0 then
PercentOfTotal = Quantity/TotalQuantity
' The Derived Field statement reports the derived field value and is only
' valid in macros used to define derived fields.
' It takes a string value, so the Str$ Function is used to convert Number to a string.
DerivedField Str$(PercentOfTotal)
Else
' If TotalQuantity was zero there was an error.
DerivedField "Error"
End If
End Sub
```

6 Press Test to test your formula. If ReportSmith finds errors, correct them and press the Test button until you receive a success message. Choose OK, then choose OK again to exit the dialog box.

7 Choose Done to exit the Derived Fields dialog box. ReportSmith executes the Percent of Total macro and inserts the results as the Percent of Total column.

To show a percent of total sales for William Blake for each of his customers, we would show that percentage in the customer group footer next to the sum of sales for each customer.

- 8 Press the Form Mode button. Select the Percent of Total heading, then *Ctrl*+click to select the Percent of Total fields. Drag and drop the selected items into the customer group footer.
- 9 Choose the Percent button on the ribbon to format the derived field.

Note See online Help or Appendix A for an explanation of macro commands.

Creating a summary field

You can create summary fields to perform operations on grouped data for the whole report, such as calculating a sum. After you create them, you can place summary fields in headers or footers.

To create a summary field using the toolbar:

- 1 Select the column to which you want to apply the summary operation.
- 2 Click the operation you want to perform on the column. You can choose the Sum, Average, Minimum, Maximum or Count button. ReportSmith inserts the summary field into each footer you have created in your report.



If you have not defined footers in your report, ReportSmith displays a message informing you to insert footers so that you can see the summary fields.

To create a summary field using Tools | Summary Fields:

- 1 Select Tools | Summary Fields to see the Summary Fields dialog box.
- 2 Select the report group level from the Report Group box.
- 3 Select the operation you want ReportSmith to perform in the Summary Operation list box. The following table shows the available summary functions:

Operation	Function
Average	Calculates the average value for a selected numeric field.
Count	Calculates the number of records in the selected group.
First	Selects the first value in the selected group.
Last	Selects the last value in the selected group.
Maximum	Calculates the largest value in the selected group.
Minimum	Calculates the lowest value in the selected group.
Std Deviation	Calculates the average standard deviation from the mean in a large report of data. It is the square root of the variance. It is a statistical test of how various values in a report of values deviate from the average value for that report.
Sum	Totals values for a selected group.

Operation	Function
Variance	Calculates the variance from the mean in a report of data. It is the square of the standard deviation. It is the measure of the amount by which all values in a group vary from the average value in the group.
Cumulative Sum	Summary fields placed into page headers and footers show summaries of the data on each page.

See “Summary fields” on page 105 for how to insert a page summary field.

- 4 Select the column you want to apply the summary field operation and place it into the Summary Fields window.
- 5 Choose OK to exit the dialog box.
- 6 Insert headers or footers into the report, if you haven’t already done so, with Insert | Headers/Footers. Use the Field command on the Insert menu to drop a summary field into a header or footer.

In our sample report for Video West, we applied the sum operation to the SUBTOTAL field for the customer name group. We then used the Field command on the Insert menu to drop the summary field into the customer group footer.

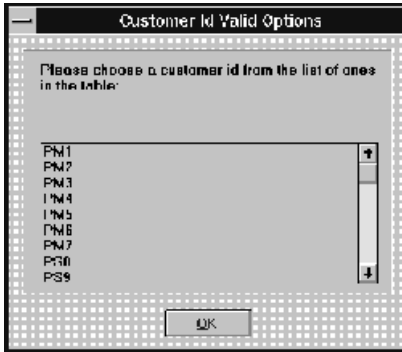
Once placed, the total sales for each customer in our sample report appears in each customer group footer:

VIDEO WEST			
77 Cassin Lane • Hollywood CA 90049 • 213-555-1010			Page 1
Sales By Employee		William Blake	
December 31, 1992			
CUSTOMER NAME	INVOICE NO.	DATE	AMOUNT
<i>Charlene's Films</i>	9487	7-Feb-92	29.61
	9E09	8-Mar-92	14.32
	9F21	11-Mar-92	18.40
	9E33	22-Apr-92	72.24
	9E45	1E-May-92	57.79
	9F69	9-Jul-92	36.76
	9E81	4-Aug-92	38.18
	9E15	1E-Sep-92	30.22
	9F41	9-Nov-92	10.57
	9E53	1E-Dec-92	57.52
			983.69

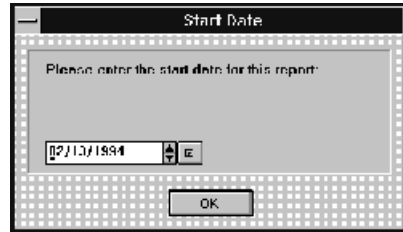
Creating a report variable

You can add dialog boxes to your reports that prompt a user for specific information. These are called *report variables*. With them, you can customize a report to obtain different results depending on data the user enters into the dialog box. Each of the

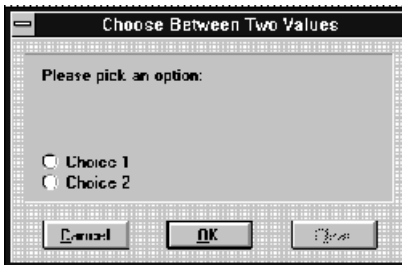
dialog boxes illustrated below is a report variable which obtains specific information from the user to create a report:



Prompt a user to enter values...



...or choose from a spin box



...or select an option

The above example shows just three of the report variables you can create. ReportSmith applies the user's value to the report.

Data types and entry types

When creating a report variable, you have five data types to choose from:

- String
- Number
- Date
- Time
- Date and Time

There are also four entry types to choose from. Any entry type can be used with any of the data types listed above:

- Type-in
- Choose from a list
- Choose from a table
- Choose between two values

The following table summarizes entry type options when paired with each data type:

Data type	Entry Options
String	Type-in Requires users to type a response to the report variable. When you select this option, you can designate the maximum characters users can enter in the Maximize Size option. The default number is 2000. Choose from a list Requires users to select a response to the report variable from a list box. When you select this option, you must also enter the allowed values in the Allowed Values list box. Use this list box to specify a list of values for the user to choose from. Type a value into the text box and press Add to place it into the list. Do this for each value you want to add. Use the Delete and Replace buttons to remove or replace values from your list. Choose from a table Requires users to choose values from tables and fields you've selected. Choose Select Table to place a table into the report variable. Choose Select Field(s) to specify fields that the user can choose. Here, you can choose a field to display from the Display value from field list and a field to use from the Use Value from field list. (For example, you might want to use the CUST_ID field but display the CUST_NAME on the report.) Choose between two values Requires users to choose a Yes or No value from a Message box or to choose from one of two values in a Dialog box. Enter prompts into the prompt boxes.
Number	Type-in Requires users to type the number response to the report variable. When you select this option, you must also select from the following check boxes: Signed determines whether users can type a + or – sign. If you leave this empty, ReportSmith uses a positive number and users can't type a + or –. Maximum determines the largest number users can type. Click this check box and enter the maximum number you'll allow users to enter. Minimum determines the lowest number users can enter. Click this check box and enter the minimum number you'll allow users to enter. Decimal determines the number of decimal places users can enter. Click this check box and enter the number of decimal places. ReportSmith uses the standard dollars and cents format. Spin indicates that a spin button (an up and down arrow) appears beside the response area to allow users to use the mouse to change the entered value. Available check boxes for this option are the same as for the Type-in option described above. For Choose from a list, Choose from a table and Choose between two values, see entry options under String, above. Options are the same.
Date	Type-in Requires users to enter the date response to the report variable. When you select this option you must enter the date format you want to appear by entering it in the text box or choosing it from the drop-down list. Choose from a list Requires users to select a response to the report variable from a list box. When you select this option, you must also enter the allowed values in the Allowed Values list box. Use this list box to specify a list of values for the user to choose from. Use the spin box or calendar button to select a date or range of dates and press Add to place it into the list. Do this for each value you want to add. Choose from a table (see entry options under String, above). Choose between two values requires users to choose a Yes or No value from a Message box and then enter the date value using the spin and calendar buttons, or to choose from one of two values in a Dialog box.
Time	Type-in Creates a variable that requires users to enter a time for a report. After creating the variable, select or enter in the Format box the time format you'll let users enter.

Data type	Entry Options
	For Choose from a list , Choose from a table and Choose between two values see entry options under Date, above.
Date/Time	For Type-in , Choose from a list , Choose from a table and Choose between two values , see entry options under Date and Time, above.

To modify a report variable:

- 1 In the Report Variables dialog box, select the report variable you want to modify from the Report Variables list. The settings for the selected report variable appear in the options.
- 2 Modify the appropriate settings in the dialog box and press Done.

To change the value of a report variable:

- 1 Do one of the following:
 - In the Report Variables dialog box, select the report variable whose value you want to change from the Report Variables list, or,
 - double-click the report variable name in the Report Variables list box.
- 2 Select or enter a new value in the dialog box and press OK. The Report Variables dialog box appears.
- 3 Choose Done. ReportSmith reruns your report based on the new value.

Sample report variables

This section shows you how to create two report variables. The first prompts the user to enter the start date for the report so that sales by employee are selected after that date. The second prompts for the name of the salesperson whose sales you want to include in your report. You can include these report variables in a selections formula, letting you filter data before running the report.

This section describes the following:

- Creating a report variable
- Deleting a report variable
- Changing values within a report variable
- Modifying an existing report variable
- Using table-driven values for report variables
- Displaying versus using different values

Sample 1: Entering or choosing a value

To create a report variable called STARTDATE:

- 1 Select Tools | Report Variables, or, press the Report Variables button in a Report Query dialog box. The Report Variables dialog box appears.

You'll use the Report Variables dialog box to create report variables by entering information into each of the text entry boxes and by choosing options. Type a name for your report variable into the Name box. The name appears in the Report Variables list for your use. Names must begin with a letter and contain only letters, numbers, and underscores.



Startdate is the name of our first report variable.

If you use a report variable in selection or derived field formulas, you must type the report variable name exactly as it was entered when created, distinguishing between uppercase and lowercase letters.

- 2 Press the down arrow in the Type list box and choose the data (or the type of response) for your report variable.

The Startdate report variable uses *Date* as the type of response.

- 3 Enter the title you want to appear in the dialog box for the report variable.

Start Date is the title for the example report variable.

- 4 In the Prompt box, type the exact wording you want to appear in the user prompt. We entered the following:

Please enter the start date for this report:

- 5 Press the down arrow under Format to see a drop-down list of date formats.
- 6 Choose a format. For our example we'll choose *Default*, which uses the format *mm/dd/yy*.
- 7 Press the down arrow under Entry to choose how you'd like the user to specify values. For our example, we'll choose **Type-In**.
- 8 Choose Add to add the report variable you defined to the Report Variables list.

You've created your first report variable. You can view the dialog box by selecting it in the Report Variables list and pressing the Value button.

Setting a date value:

To set a date value for a report variable, you can either type it in manually, or use ReportSmith's calendar.

- 1 Press the calendar button next to the text entry box to see the calendar.
- 2 To report the date value use the Year, Month and Day items:

Year	Use the right and left arrows on either side of the year shown to choose the appropriate year.
Month	Use the right and left arrows on either side of the month to choose the appropriate month.
Day	Press the button corresponding to the appropriate day.
- 3 Choose OK.

Sample 2: Choosing from a list

Here, we'll create a second report variable called *Salesperson*. Instead of using the Type-In Entry option, we'll use the List Entry option, which lets you type a list of names which users can choose from in the dialog box.

To create the Salesperson report variable:

- 1 Place the cursor into the Name box and type the following:

Salesperson

- 2 Under Type select String.

- 3 In the Title box, type the following:

Salesperson

- 4 In the Prompt box, type a prompt for the user:

Which salesperson do you want to see?

- 5 Select `Choose From A List` from the Entry drop-down list.

We're going to create a list of data from which the user can choose. You'll notice that when you select this option, settings in the dialog box change to display settings for `Choose From A List`.

Enter the following values into the text entry box and press the Add button (or the *Enter* key) after each one to add it to the Allowed Values list:

Blake
Ginsberg
Keats
Pound
Rimbaud
Snyder

- 6 Press the Add button on the left side of the dialog box to add the Salesperson report variable to the Report Variables list.
- 7 To see the dialog box you just created, highlight `Salesperson` in the Report Variables list and press the Value button. ReportSmith displays the Salesperson report variable.
- 8 Press OK to exit the report variable.

After you create the report variables, press the Selections button to see the Selections dialog box where you'll insert them into your selection criteria. By inserting the report variables into the Selection formula, ReportSmith selects the necessary data to insert in your report up-front, based on the user's responses to the prompts.

Before you delete a report variable, make sure it isn't used anywhere else in your report. If it is, you'll need to delete it from those areas.



Sample 3: Choosing table-driven values

ReportSmith lets you create a list of values for a report variable with values from a column in one of your tables. Rather than typing in a list of values, you can simply display fields from your database. This eliminates the need for manually typing and updating values if your data changes, ensuring that your reports always contain the most up-to-date data.

To use table-driven values:

- 1 Open the Report Variables dialog box. Enter a name for your report variable, choose a data type, enter a title, and a prompt.
 - 2 Under Entry, select Choose From A Table. The dialog box displays associated settings.
 - 3 Choose the Table button. The Select List Table For Report Variable dialog box appears. In the Files list, choose the table that contains the column you want to use. If you need to, navigate using the Type, Drives, Directories, or Connections list boxes.
- Note** You can use a different table or data source type than the one your report was created with. For example, if you have a dBASE report, but you have saved a list of values you want in a FoxPro table, you can connect to FoxPro here and use that table.
- 4 Press OK.
 - 5 Under Select, press the Field(s) button. The Report Variable Fields dialog box appears.
 - 6 Under Display Value from Field, choose a field to display to the user. This is the value the user sees; it is not necessarily the value used. Under Use Value from Field, choose the field you want to report the report variable equal to. This is the value used by your report variable.

In most cases, the fields you choose for Display Value from Field and Use Value from Field are the same. However, there are some cases when you'll want to *display* one field but *use* another.

For example, suppose you have two columns in your table: EMP_LNAME and EMPLOYEE_ID. The EMPLOYEE_ID field has an index which enables you to search through it quickly, so you want your selection criteria to search that field. However, you don't want your users to have to know employee ID numbers in order to run a report. In this case, you'd display an employee's name for your user to select, while using the employee's ID number as the value for your selection criteria.

- 7 Press OK and press Add. ReportSmith adds your report variable to the Report Variables list.
- 8 Press Done.

Creating a master /detail report

The Insert | Detail Reports command lets you build reports using more than one query. Queries are linked on a key to define a relationship between the master report and the

subquery, or detail. Generally, the master report has one record per key value, while the detail report contains many records per key value.

An example of a master/detail report is your bank statement—at the top, it displays your name, account number, and address from a customer table. The next section displays withdrawals for each account. You could also have another section which displays only deposits. Information in each section originates from different tables with different qualification. Furthermore, each section might have its own formatting and totals.

The diagram below shows the idea behind a master/detail relationship for a bank statement. The report created can describe three different queries: “CUSTOMER=Bob Smith,” “CUSTOMER=Bob Smith AND transaction_type= DEPOSIT,” and “CUSTOMER=Bob Smith AND transaction_type=WITHDRAWAL.” In this case, each query is linked on the CUSTOMER field.

Another example of using a master/detail report might be one in which each department shows a breakdown of information by each employee. In this case, the main report uses the DEPARTMENT table, while the detail report uses the EMPLOYEE table.

Note You can only create master/detail reports for columnar reports.

To create a master/detail report:

- 1 Create a new columnar report to serve as the master report. Choose Insert | Detail Reports. In the Master/Detail Reports dialog box, “Master Report” is the default name for the master query.
- 2 To add another query to your report, choose Add to see the Tables dialog box. In the Tables dialog box, choose Add Table, select one or more tables and choose OK. Select columns to include or exclude and press Done.
- 3 In the Link Master/Detail Reports dialog box, ReportSmith has selected a common field in each table for linking tables. You can use these fields or select another common field to link the tables. Place at least one common field into both Key Fields list boxes and choose OK.
- 4 In the Master/Detail Reports dialog box, choose OK to load records onto your report. Make sure View | Boundaries is on so you can see the distinct sections of your report: Main Report 1, and Detail Report 1.

Note Once you create a detail report you can select it by choosing it from the drop-down detail reports list box that appears in every dialog box after you create one.

Naming a detail report

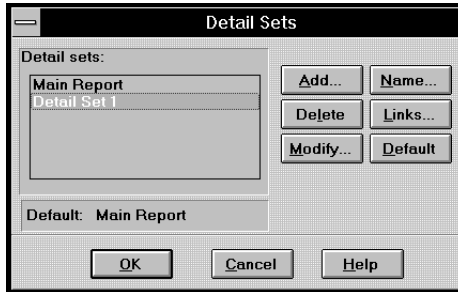
Each time you create a detail report, ReportSmith creates a default name for it. The master detail report you create is called “Master Report.” The next is called “Detail Report 1,” then “Detail Report 2” and so on. The name of the detail report appears in the report group to which it corresponds. You rename a detail report so it’s more recognizable. For example, you might want to change “Master Report” to “Customer Report” and “Detail Report 1” to “Videos Rented.”

- 1 With your report open, choose Edit | Detail Reports to see the Master/Detail Reports dialog box. Select the detail report you want to name and choose Name.

- 2 Enter a name for the detail report and choose OK. The name appears in each associated section on your report.

Using the Detail Sets dialog box

This section briefly describes each option in the Detail Sets dialog box.

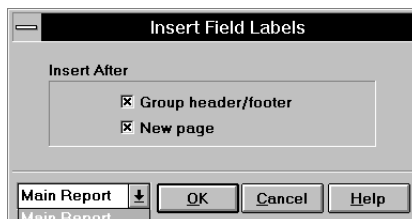


Option	Description
Add	Add a new detail report to the current report.
Delete	Remove a detail report from your report. You cannot remove the Main Report detail report, otherwise you'd have a blank page.
Modify	Change the query for a detail or the main report.
Name	Rename a detail report.
Links	Change links associated with a detail. To enable this option, you must first select a detail report. A defined link can have more than one linkage key.
Default	Select which report or detail report selected by default when you are performing other operations. For example, if you select "Master Report" as the default when you choose Tools Sorting or Tools Derived Fields, you'll see "Master Report" in the drop-down list box as the default.

Displaying field labels

ReportSmith displays field labels for each detail report the first time a master record appears on the page. You can display field labels each time a detail report displays a new set of data by using the Insert | Field labels command.

- 1 With your report open, choose Insert | Field Labels. You'll see that the dialog box displays a list box containing the name of each detail report. Select a detail report.
- 2 Turn on Group Header/Footer and New Page options and choose OK. Labels now appear for each set of data displayed for the detail report.



Inserting a field into a report section

Use the Insert Field dialog box to insert a field into the main report area or a detail. You can place a field only into a section built on the associated query. If you try to place it into the wrong detail report, an error message appears. Placement bars appear in appropriate drop-zones.

The drop-down list box at the top of the dialog box lets you choose a master or detail report whose fields you want to display in the list box below.

To insert a field into a detail report:

- 1 With your report open, choose Insert | Field.
- 2 In the top drop-down list select the report in which you want to insert a field. This populates the Field Name list with fields associated with that report.
- 3 If you want the field name to display on your report, check the Include Field Name option. Drag and drop the field into the desired section.

Formatting a report section

You can apply formatting styles to each detail report to emphasize them. You can also apply one formatting style to an entire report.

- With your report open, choose Format | Report Styles.
- Choose a detail report from the drop-down list box at the bottom of the dialog box to which you want to apply a style.
- Select a style and press Apply. Do this for each detail report you want to format.

Note Most other dialog boxes that let you modify your query will let you select which detail report your changes apply to from a drop-down list box.

Saving a report

If you're ready to begin work on another report, or you'd like to exit ReportSmith, follow these steps to save your report.

- 1 Press the Save button or choose File | Save.
- 2 In the Save As dialog box, navigate to the drive and directory where you'd like to store your report. Select a file type from the Files of Type list. For example, you can save your report with an .XLS extension if you wish to export it for use in an Excel spread sheet, .WKS for a Lotus spread sheet, .CSV for comma delimited text and so on.

For a complete list of available file types, see the online Help.

- 3 Enter a name for your report into the Filename text box and choose OK.
- 4 To exit ReportSmith, choose File | Exit.

Creating a temporary table

You can create temporary tables to perform more complex functions within ReportSmith. A temporary table contains static data that doesn't change when the originating table changes (unless you instruct it to). Temporary tables are useful for reporting on a selection of groups or for grouping and sorting on macro-derived fields. To create a temporary table, you must first create a set of data to populate the table. Types of data that are "included" columns (such as columns listed in the SELECT statement, including SQL-derived fields), summary fields, and macro-derived fields. After creating a report with the desired data, you can use a macro to export the table through an ODBC driver.

For example, you can use a temporary table to create a view of a table which contains only a subreport of its data. Also, you can use temporary tables to perform summaries on macro derived fields or to sort data by a summary field.

Note The temporary table itself must be stored in an ODBC connection, however, the original data can originate anywhere.

When you save a report as a temporary table, it saves each aspect of the report as a separate column. Data columns and derived fields are named numerically in order. Summary fields appear as columns and use the following naming convention:

SumFunction|GroupLevel_Column#

Element	Description
SumFunction	The summary function for the summary field (Min, Max, Sum and so on).
Group Level	The group level to which the summary is set.
Column#	The position or order number of the summary column in your report.

For example, the name of a summary field that counted the number of values in the 3rd column at the Entire Report Group level in your report is `COUNT1_3`, and the name of a summary field that summarized the first column in your report at your primary group (first group you have created) is `SUM2_1`.

To create and use a temporary table:

- 1 Open a report.
- 2 Write a report macro like the one below to export your report data to a temporary table:

```
Sub temptable()  
    'Create a dataset object to hold the temp table values  
    dim DS as dataset  
    DS.SetFromActive  
    ' Export the table to the database for this new report to use  
    DS.exporttable 'mytable', 55, "RS_dbase", " ", " ", " "  
End Sub
```

- 3 Link the macro you created to a report-level event, such as the closing of your report.

- 4 After making sure this macro has run, create a new report, and choose the table with the name that matches the name you indicated in the export table command in your macro. In the above example, it is “mytable.”

You now have a new report containing a separate column for each data field, derived field, and summary field from your original report. You can now manipulate these fields in the same manner as regular data fields (sort, group, and perform summaries.)

You can save your report as an *.RQF* file. This type of file can be used to update a temporary table each time you run a report. It contains a description of how to get the data you just exported from the source. Creating an *.RQF* file can be done through File | Save As, or via a macro. If you choose to save the file via a macro, add the following line to the macro above that is used to export the table:

```
DS.Save "Filename.RQF"
```

To create a ReportSmith query file (*.RQF*):

- 1 Create a new report.
- 2 Choose File | Save As. The Save Report dialog box appears.
- 3 Under File Type choose ReportSmith query (*.*RQF*) files.

Note You probably want to name an *.RQF* file the same as you named your temporary table.

Formatting

ReportSmith's versatile formatting capabilities let you transform your data into high quality presentations. This chapter discusses formatting techniques you can apply to any kind of report.

Selecting and moving objects

You can select and move columns, fields, and objects. You can also resize items in a report.

Columns

When you create a columnar report, fields appear as columns across the report page. A column heading, or *field label*, identifies each column. ReportSmith takes field names from your database and applies them to columns on your report, creating the field labels. You can change the field labels after creating them.

Use Column mode to move columns horizontally across the page, moving labels along with columns. Columns can move closer together as the selected columns are removed and other columns shift to accommodate the selected columns.

If you delete a column in Column mode, ReportSmith repositions the remaining columns moving them closer together and filling in the blank space.

Selecting a column and its field label



- 1 Press the Column Mode button on the toolbar.

2 Click on the column.

LAST_NAME	FIRST_NAME	TELEPHONE
White	Cindy	415-948-6547
Jones	Arthur	415-941-4321
Parker	Debra	916-213-2234
Sawyer	Dave	415-948-9998
Davis	Jennifer	516-292-3945

After selected, you can move the column and its label to a different location, format the column, or delete it from the report.

To select multiple columns:

- Click and drag the mouse, stretching the dotted box until it surrounds all of the columns you want to select
- *Ctrl*+click to select more than one column and release the mouse.

LAST_NAME	FIRST_NAME	TELEPHONE
White	Cindy	415-948-6547
Jones	Arthur	415-941-4321
Parker	Debra	916-213-2234
Sawyer	Dave	415-948-9998
Davis	Jennifer	516-292-3945

- *Shift*+click to select a block of columns.

LAST_NAME	FIRST_NAME	TELEPHONE
White	Cindy	415-948-6547
Jones	Arthur	415-941-4321
Parker	Debra	916-213-2234
Sawyer	Dave	415-948-9998
Davis	Jennifer	516-292-3945

To move a column:

- 1 Drag the column to a new location. You'll see a drop bar that indicates the position of the column if you release the mouse in the current location.
- 2 Release the mouse. ReportSmith repositions the column and its label, rearranging the columns around it.

To cancel the move while dragging a column, right-click the mouse. The column returns to its original position.

Fields



Fields are the data within columns. Use Form mode to move individual fields freely around the report page—horizontally and vertically. Use this mode to place fields on labels and forms and into headers and footers.

Form mode does not reorder column positions.

To select fields in a column:

- 1 Press the Form Mode button.
- 2 Click to select a column. Single boxes surround each field.

LAST_NAME	FIRST_NAME	TELEPHONE
White	Cindy	415-948-6547
Jones	Arthur	415-941-4321
Parker	Debra	916-213-2234
Sawyer	Dave	415-948-9998
Davis	Jennifer	516-292-3945

Use the sizing handles that appear on the first record to resize all fields or to add space between rows or columns.

Note You can also use the Row Height or Field Height command on the Format menu to specify a distinct size.

To select fields in multiple columns:

- Click and drag the mouse, stretching the dotted box that appears until it surrounds all the fields you want.
- *Ctrl*+click on more than one column and release the mouse when you've finished selecting.
- *Shift*+click to select a block of column fields.

Rows

You can select rows of data across columns. A *row* is the set of values from within each column that comprise a set of information. After you select a row, you can resize and format it. Changes you apply to the selected row apply to all of the rows in your report. You can resize a row to create blank space between your records or to create a form report. You can stretch a row up to the size of a full page.

Selecting a row

Click the left margin of the row between the data and the edge of the report. A box with sizing handles surrounds the selected row.

LAST_NAME	FIRST_NAME	TELEPHONE
White	Cindy	415-948-6547
Jones	Arthur	415-941-4321
Parker	Debra	916-213-2234
Sawyer	Dave	415-948-9998
Davis	Jennifer	516-292-3945

Headers and footers

You can select a page header or footer to resize it. An example of enlarging a footer might be to insert an oversized graphic, or to enlarge a subtotal within it for emphasis.

To select a header or footer:

- 1 Click in the boundary to select a header or footer.
- 2 Pull on the sizing handles to stretch or shrink the boundaries.

Note See “Placing items into a header or footer” on page 107 for how to place fields into a header or footer.

Body

You can select a body of data in a report. A body is any data positioned between a header or footer which appears as a group. After selected, you can add borders, shading, or patterns to the body of your report. You can also resize the width of the body's border by dragging the sizing handles.

To select the body of a report, double-click in the body you want to select.

Text

You can select text on your report.

To select text:

- 1 Click once on the text to select it. You can then drag and drop it into place.
- 2 Click again. The Insertion Point cursor appears to the left of the text, indicating that you are in Text Mode. You can now revise the selected text.
- 3 Once edited, click outside of the field to accept your changes.

Resizing objects

ReportSmith makes it easy for you to resize objects such as columns, fields and sections in your report quickly. Just click to select the object and pull on the sizing handles surrounding it to enlarge or decrease the size.

To resize objects in your report:

- 1 Click to select the object.
- 2 Pull on one of the sizing handles to increase or decrease the size of the object.

In a label report, headers and footers appear as the same size as your label, unless you specify otherwise. To resize a header or footer in a label report:

- 1 Select the header or footer you want to resize.
- 2 Right-click the mouse to see a pop-up menu.

- 3 Choose Detach Label. ReportSmith detaches the selected section from the size of the labels. Now, you can use the sizing handles to resize it. Only the page header and footer can be detached from the set of labels and independently resized or moved.

Note To resize a crosstab, see “Resizing rows and columns” on page 145.

Using Page Setup settings

When you create a report with the New command, ReportSmith sets up a default report page layout. However, you can change margin, paper size, and orientation settings.

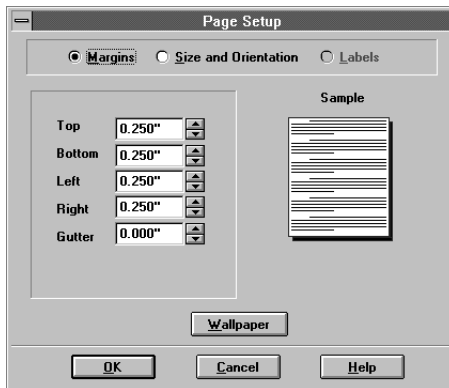
Page Setup settings apply to each new report you create. They are also applied to your active report. You can set margins and other attributes at any time for any report. You can also add a background to your report page layout.

Setting margins

Every report page has defined margins you can set with the Page Setup command on the File menu. ReportSmith has options for the top, bottom, left, right, and gutter margins.

To set margins for a report:

- 1 Open a report.
- 2 Select the Page Setup command from the File menu. The Page Setup dialog box appears.



Note Watch the Sample box as you adjust each margin setting. Changes you make are reflected instantly so you can make sure you’ve specified exactly the settings you want.

Option	Function
Margins	Click in this option and the settings in the dialog box change to reflect settings for margins including top, bottom, left, right, and gutter spacing.

Option	Function
Size and Orientation	Click in this option and settings in the dialog box change to reflect size and orientation settings including paper size, width, height, and orientation.
Labels	Click this option and settings in the dialog box reflect settings for labels. The Labels option is only available for label reports. Other report types display this option in gray to indicate that it's not available.

Selecting any of the given label types changes the label dimensions, paper size, and margins to match the label definition. The maximum number of labels that fits within the page display in the Sample box.

ReportSmith uses the non-printing areas established for your printer as the default printer margins. These margin settings appear in the Printer Margins dialog box. While you can set margins that fall inside the non-printing areas, parts of your report can be clipped off if you do. To view the print area, turn on the Print Area command on the View menu.

Default measurements

The default measurement scale is in inches. However, you can change the default to centimeters with the Options command on the Tools menu. To use a measurement scale other than the default, enter the number and one of the following abbreviations in the measurement boxes for each margin.

Measurement	Action
Inches	Enter the number of inches and <i>in</i> or <i>"</i> .
Centimeters	Enter the number of centimeters and <i>cm</i> .
Picas	Enter the number of picas and <i>pi</i> .
Lines	Enter the number of lines and <i>li</i> .
Points	Enter the number of points and <i>pt</i> .

If you use a measurement scale other than the default, ReportSmith converts it to the default measurement. For example, if you enter 3 *pi* in the top margin box and inches is the default unit, ReportSmith converts it to .50 inches

Setting paper size and orientation

ReportSmith sets page margins based on the paper size you're using to print the report.

To set the paper size:

- 1 From the File menu, choose the Page Setup command. The Page Setup dialog box appears.
You can also set paper size for many printers in the Print Setup dialog box. The settings in the Page Setup dialog box override those set in the Print Setup dialog box.
- 2 Click the Size and Orientation option button in the Page Setup dialog box. Options in the dialog box change to Size and Orientation settings; paper size, width, height, and portrait and landscape options appear.

- 3 Press the down arrow in the Paper Size box to select a default paper size you're using to print the report. Alternatively, enter a custom paper size in the Width and Height boxes.
- 4 Press OK.

To set the paper orientation:

- 1 Select File | Page Setup. The Page Setup Dialog box appears.
- 2 Click the Size and Orientation option. Settings in the dialog box change to reflect size and orientation settings.
- 3 Select either Portrait (vertical) or Landscape (horizontal) from the Orientation box. (If you have a report that's wider than 8 1/2 inches, choose Landscape as this gives you a width of 11 inches.) Press OK.

Setting up labels

If you've chosen Label as your report type in the Report Types dialog box, you'll need to set up the layout of your labels (specifications). You'll do this in the Page Setup dialog box, using the Labels option.

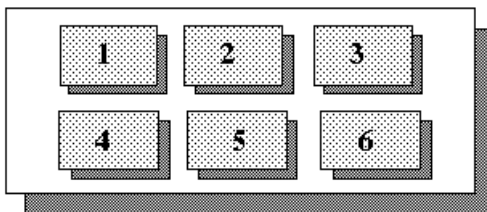
To set labels specifications:

- 1 Choose File | Page Setup. The Page setup dialog box appears.
- 2 Click in the Labels option. This option is only available for Labels reports. Settings in the dialog box change to reflect labels settings.
- 3 Under Label Type and Dimensions, press the down arrow and select a type. Selecting any of the given label types changes the label dimensions, paper size, and margins to match the label definition. The maximum number of labels that fit within the page are displayed in the Sample box.

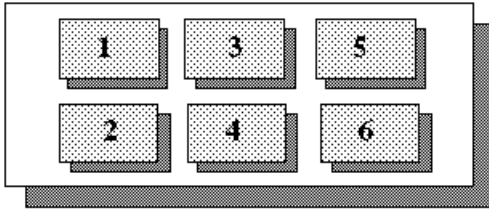
ReportSmith settings are compatible with Avery labels. ReportSmith supplies you with over 30 Avery label styles or you can create your own custom labels. A list of styles appears in the Label Type and Dimensions list on the Labels Page Setup dialog box.

- 4 Set the Width, Height, and Vertical and Horizontal Gaps. These settings indicate the amount of space between labels.
- 5 Click in one of the following options: Display Across Then Down, or Display Down Then Across.

If you choose **Across Then Down**, your labels appear like this:



If you choose **Down Then Across**, your labels appear like this:



Adding a report background

With ReportSmith, you can place objects in the background of a report to give it a unique look. You can do this in two ways: with the Wallpaper option on the Page Setup dialog box or with the Picture and Object commands on the Insert menu.

To add a background with Insert menu commands:

- 1 Open a report.
- 2 Choose either Insert | Picture or Insert | Object.
- 3 Place the object on the report page, then click and drag one of the sizing handles until the object is the appropriate size.

To reposition a background object, click and drag it to the new position.

- 4 Click and drag the body of the report until it's positioned on top of the background object. The object now resides in the background of the report.

To add a background with the Wallpaper option:

- 1 Open a report.
- 2 Choose File | Page Setup. The Page Setup dialog box appears.
- 3 Press the Wallpaper button. The Wallpaper dialog box appears.

With the Wallpaper options, you can specify the bitmap, its position on a page, and whether you can print the bitmap with a report.

- 4 Click the down arrow and a drop-down list of bitmaps appears. These bitmaps are stored in the ReportSmith directory.
- 5 Select a bitmap from the list. Alternatively, type in the path and file name of a bitmap in another directory.
- 6 Choose an option to position the bitmap in the background of the report.

The following table lists all available options for positioning a bitmap:

Type	Entry Options
Center	Positions the bitmap in the center of the page.
Horz/Vert Just.	Positions the bitmap horizontally.
Left	Positions the bitmap on the left side of a report page.

Type	Entry Options
Right	Positions the bitmap on the right side of a report page.
Vert	Positions the bitmap vertically.
Top	Positions the bitmap at the top of a report page.
Bottom	Positions the bitmap at the bottom of a report page.

7 Make sure the Do Not Print check box is deselected so the bitmap prints with the report.

8 Press OK.

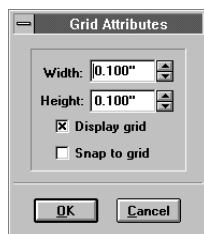
ReportSmith places the bitmap in the background of the report according to the settings you've chosen.

Using the grid

ReportSmith provides a grid which guides the placement of data and text fields. The grid is a series of row and column coordinates which you can use as a reference when placing objects. You can specify the spacing of your grid by adjusting the Width and Height parameters in the Grid Attributes dialog box. The illustration below shows a report with the grid displayed.

Title					
CSTMJR_ID	EMPLOYEE_ID	CSTMJR_NAME	CONTACT_LN	CONTACT_FN	PHONE
C1	E1-23	Videos R Us	Zoellner	Sherman	415/399-5693
C2	E4-62	Ernie's Video	Devos	Harry	415/997-4576
C3	E2-47	Movietime	Goldberg	Margaret	415/905-3459
C4	E2-30	Captain Nemo Videos	Yates	Stephen	415/303-2332
C5	E3-25	Videos by Alphonse	Hoover	Terry	415/903-2332
C6	E2-12	Videotime	Wise	Peter	415/901-1104
C7	E1-23	Us R Videos	Jones	Robert	415/309-0087
C8	E4-62	Charlene's Films	Williams	Deborah	415/907-9960
C9	E2-47	Another Video Store	Allen	Pedro	415/916-8843

Use the Snap to Grid feature to place fields and text at these coordinates. The Snap to Grid feature makes data placement and alignment easy. The grid is invisible until you select the Grid option from the View menu. When you select this option the Grid Attributes dialog box appears.



Formatting techniques

ReportSmith lets you format characters (text, numbers and symbols) in a column or field label. Within a report, you can

- Format characters in columns
- Format numbers in columns
- Format field labels
- Align values in columns
- Position columns

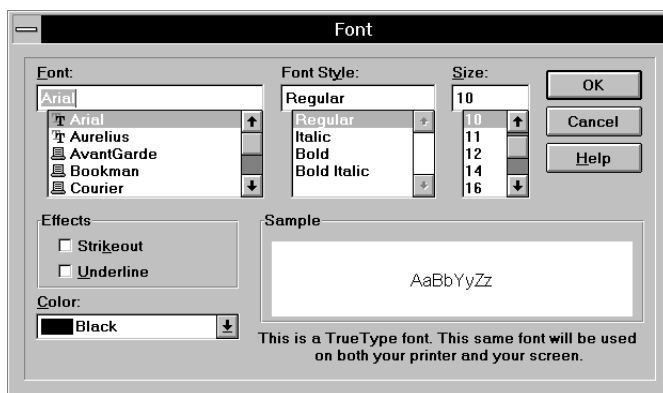
Characters

You can format characters in a column by using

- Format | Character command
- The ribbon
- Pop-up Format menu

To format a column with the Character command:

- 1 Click once on the column you want to format. Use *Shift*+click to select additional columns.
- 2 To display the Font Dialog box, do one of the following:
 - Click the right-mouse button and select the Character command from the pop-up Format menu.
 - Choose Format | Character.



Fonts and point sizes

Fonts and point sizes establish the basic look of characters in your report. A font is a set of characters with a unique design. You can format your reports to use your favorite font. The default font in ReportSmith is Arial. To change the default font setting use the Tools/Options command and change the Default Font setting in the Options dialog box.

Each font comes in a range of sizes, measured in points. In this manual, *font size* is sometimes referred to as *point size*. In ReportSmith, you can specify any size from 4 points through 127 points. To select a font size that is smaller than 8 points, type a new size into the Size box. Windows attempts to render a font that matches the one you've chosen.

Note Available font sizes depend on the fonts you have installed on your machine. Here are examples of different font sizes:

24 pt 10 pt 12 pt 8 pt

- 1 Select the formatting options you want. You can set the font, style (bold, bold italic, and italic), size, effect (strikeout or underline), and color. You can use character formats to emphasize text and to create special effects in your reports. The following character formatting options are available in ReportSmith:

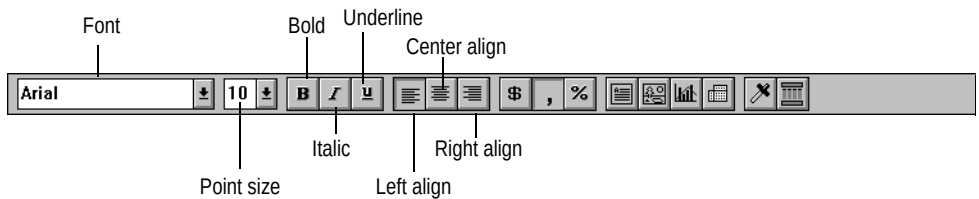
bold *italic*
~~strikeout~~ colors
underline ***bolditalic***

You can apply a format by choosing it. Remove the format by choosing it again. You can apply multiple character formats to the same text, such as adding an underline to a bold word. The Sample window displays what each selected format looks like.

- 2 Click the OK button when you've finished setting the formatting options. ReportSmith applies the settings to the selected column.

To format a column with the ribbon:

- 1 Click once on a single column or select multiple columns with the mouse.
- 2 Select the font and point size for the characters.
- 3 Use the following ribbon buttons to format the column.



Aligning characters in columns

After you select a column or group of columns, you can align characters to the left, right, or center.

To align characters within a column:

- 1 Select the column or columns with the mouse. A box with sizing handles surrounds the selected column or columns.
- 2 Click the appropriate alignment button on the ribbon.

Positioning columns

You can drag and drop columns in a report and place them in any order. You can also adjust the distance between columns. When you're positioning columns, press the Column Mode button on the toolbar. This mode lets you move columns, along with column labels, vertically across the report.

Note The database settings in the tables or files you include in a report determine the initial width of columns.

To move a column:

- 1 Press the Column Mode button on the toolbar.
- 2 Click the column you want to reposition and hold down the left-mouse button.
- 3 Drag the column to the new location in the report. A drop bar appears at the new location.
- 4 Release the mouse button and the column repositions itself. Notice that the columns are reordered and placed accordingly, filling up any blank spaces on the report page.

To adjust column width:

- 1 Click the column whose width you want to change.
- 2 Right-click and choose Column Width, or select the Column Width command on the Format menu. The Column Width dialog box appears.



- 3 Use the arrow keys or type a number in the Column Width box to set the width for a selected column, or press the Best Fit button.
- 4 Do one of the following:

- Press OK. ReportSmith applies the new width to the selected column.
- Click the Use Default Width check box and press OK.

ReportSmith uses the default column width determined by the tables or files from a database you included in a report.

- Press the Best Fit button. ReportSmith determines the best width of a selected column by scanning all records of your report for the largest field within that column.

Note For a large report, this function may take some time to calculate the best width.

Field labels

You can format characters in a field label separate from the fields in a column. For example, you can bold and increase the point size for emphasis.

To format a field label with Format | Field:

- 1 Select the first field label you want to format. A box surrounds the selected field label.
- 2 Use *Shift*+click to select all field labels in between the first selected field label and the one you *Shift*+click. If you want to select more than one individual field label, use *Ctrl*+click.
- 3 Right-click and choose Character, or choose Format | Character. The Font dialog box appears.
- 4 Select the formatting options you want. You can set the font, style (bold, bold italic, and italic), size, and effect (strikeout and underline). The Sample window shows the selected format. Press OK.

To format a field label with the ribbon:

- 1 Select the field label you want to format. A box surrounds the selected field label.
- 2 Select the font and point size for the characters.
- 3 Press only the following buttons on the ribbon to format the field. For more detailed formatting options, such as strikeout, underline and smaller fonts, use Format | Character.

To create multiple-line field labels:

- 1 Press the Form Mode button.
- 2 Select the column header area by placing your cursor to the left or right side of it. The cursor changes into an arrow and points to the column header area.
- 3 Click once.
- 4 Increase the size of the label. Drag the area down the page so it is resized to at least two characters high.
- 5 Select the label you want to appear on two lines. Click it again to enter Text mode.
- 6 Split the label onto two lines by placing your cursor in the appropriate place and pressing the *Return* key. Next, click outside of the column to deselect it.
- 7 Repeat the above step for all columns, or move the others down the column header area.

Numbers and dates

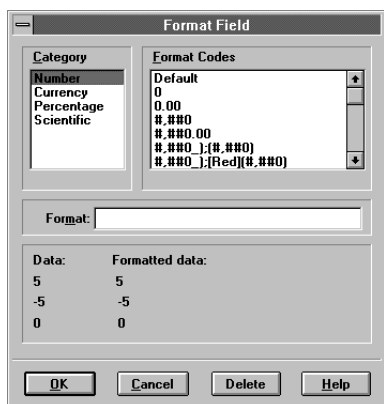
To set number formats, ReportSmith formats numbers in columns under several categories:

- Number
- Currency

- Percentage
- Scientific

To set number formats with the Field Formats command:

- 1 Select the column with the numbers you want to format.
- 2 Right-click the mouse and select the Field Formats command from the pop-up Format menu, or choose Format | Field Formats. The Format Field dialog box appears.



- 3 Choose the format category in the Category box.
- 4 Choose the format code in the Format Codes box. The selected code appears in the Format box. You can also enter a custom code by typing it directly into the Format box.
- 5 Press OK. ReportSmith applies the format to the selected column.

Choose the Index command on the Help menu and access the Field Formats command for an explanation of format codes.

To set number formats with the ribbon:

- 1 Select the column with the numbers you want to format.
- 2 Click the field format button that represents the format you want. These buttons appear on the ribbon.



- Formats selected numeric fields as standard dollars and cents (\$100,000.00).



- Inserts commas to separate digits in selected numeric fields (100,000).



- Calculates standard percentage for selected numeric figures (50%).

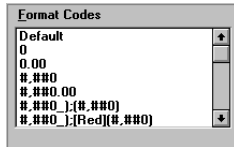
To set date and time formats with the Field Formats command:

- 1 Select the column with the dates or times you want to format. A box surrounds the column.
- 2 Right-click the mouse and select the Field Formats command from the pop-up Format menu, or choose the Field Formats command from the Format menu. The Format Field dialog box appears.

- 3 Select the format category you want from the Category box.
- 4 Select the format code you want from the Format Codes box, or type a custom code format directly into the Format box. The selected code appears in the Format box.

Customizing numeric fields

ReportSmith supplies a variety of common field format codes which you can select from for both dates and numbers. They are shown below.



Additionally, you can create your own customized formats. To create a custom format, type one into the Format box using the following structural guidelines:

```
[positive format];[negative format];[zero value format]
```

Note Separate formats with semicolons.

The first number format you type represents the format of any positive numbers; the second represents the format of negative numbers. The final format represents what you want to appear on your report if the data value is zero. You can choose to display zeros, leave the fields blank, or print a message such as “No Data Available,” among other options.

Suppose you want to display positive numbers, negative numbers in red, and instead of showing fields with zero values, you want to display “No Data Available.” To do this, you’d type the following formula into the Format box:

```
(#,##0):[Red](#,##0);"No Data Available"
```

If a field value is a positive number, it uses:

```
#,##0_
```

If a field value is negative, it uses:

```
[Red]#,##0
```

If a field value is 0, it uses:

No Data Available

You can modify ReportSmith’s system codes to create your own formats. Save time by placing one of ReportSmith’s format codes into the Format box, then type your modifications.

Leaving null values blank

If you want to leave zero values blank in your report, type quotes in the zero value format position. For example, if we modify the formula shown above so it leaves zero values blank, the formula looks like this:

###.##; (Red) ###.##; " "

To create custom date formats:

You can create your own custom date formats too. For example, to create a date in the format **November 8, 1994**, type the following format into the Format box: *Month D, YYYY*.

Entering and formatting text

You can enter or modify text in a report with the Text button on the Drawing toolbox or the ribbon.

To enter text in a report:



- 1 Choose Tools | Drawing and click the Text button on the Drawing toolbox, or press the Text button on the ribbon.
- 2 Click where you want to place the text in your report.

Note

When placing text within a section—a row or page header or footer, for example—it appears in each section of the same type. For example, if you place text in a row, it repeats in every row. A blinking text cursor appears.

- 3 Type the text. Click the cursor anywhere on your screen when you've finished typing.

To edit text:

- 1 Click once on the text you want to edit with the mouse to select it. A box with sizing handles surrounds the text.
- 2 Click again within the text box and the text cursor appears for you to enter your text.
- 3 Modify the text.
- 4 Click outside the text box when you've finished editing.

To format text with the Character command:

- 1 Select the text you want to format with the mouse. A box with sizing handles surrounds the text.
- 2 Click the right-mouse button and select the Character command from the pop-up Format menu, or select the Character command from the Format menu.

The Font dialog box appears.

- 3 Select the formatting options you want. You can set the font, style (regular, bold, bold italic, and italic), size, and effect (strikeout and underline). Press OK.

To wrap text:

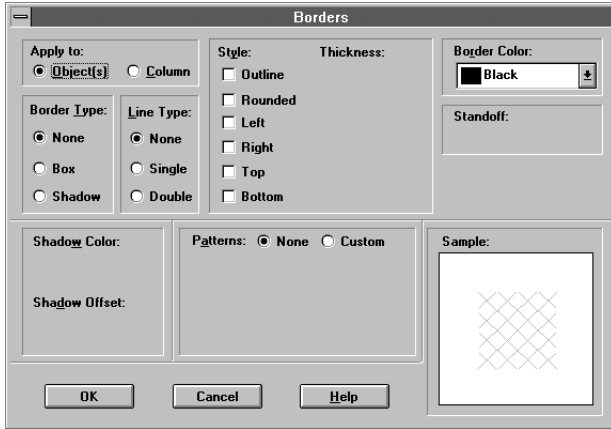
- 1 Select the text item.
- 2 Pull on the sizing handles to stretch or reduce its boundaries so that it wraps as long or as short as you want it to. It only wraps at spaces. If you have no spaces in your text, press the *Return* key where you want it to wrap.

Adding borders

When you're formatting a report you can add a border for style or emphasis. This section describes how you can add borders to reports and items within them.

To place a border around an item:

- 1 Make sure you haven't selected anything else in the report and select the item you want to place a border around.
- 2 Choose Format | Border, or select the item and right-click the mouse, then choose Border from the pop-up menu. The Borders dialog box appears.



Note The Sample window shows how a border changes as you select options. The gray crosshatch represents the object surrounded by the border.

- 3 Click the Box button under the Border Type options.
- 4 Select the type of line to use for the border from the Line Type options. A box surrounds the selected item with the chosen line border.
- 5 Choose the color for the border with the Border Color option. ReportSmith displays the border with the selected color.
- 6 Click the Outline button under Style options to place a square border around the item or the Rounded button to place a border with rounded corners.

Note If you don't want to add a border around all sides, use a combination of the Left, Right, Top, or Bottom styles to achieve the desired effect.

- 7 Set the distance between the border and the item with the Standoff option. Click repeatedly on the bottom arrow to move the border closer to the body of the item. Click repeatedly on the top arrow to move it further away from the item.
- 8 Set the line thickness for the border with the Thickness option.
- 9 Press the OK button when you're satisfied with the border settings. ReportSmith places a border around the item.
- 10 Press the Whole Page button on the ribbon to see how the border looks.

To undo a border:

- 1 Click the bordered object.
- 2 Right-click and choose Border from the pop-up Format menu, or select Format | Border. The Borders dialog box appears.
- 3 Select the Column button under the Apply To options if you've selected a column.
- 4 Select None under the Border Type options and press OK.

To add shadow to a border:

- 1 Select the object or objects you want to shadow.
- 2 Click the right-mouse button and choose the Border command from the pop-up Format menu, or choose the Border command on the Format menu. The Borders dialog box appears.
- 3 Select the Column button under the Apply To options if you've selected a column.
- 4 Set the Border Type, Line Type, Style, Thickness, Standoff, and Pattern Options.
- 5 Click the Shadow button under the Border Type options. The controls for setting Shadow Color and Shadow Offset appear.
- 6 Choose a color. The default is dark gray.
- 7 Set the Shadow Offset measurement for the distance between the shadow and the object or column you're shadowing.

- Note** Examine the Sample to make sure you have the formatting look you want before applying it.
- 8 Press OK. ReportSmith darkens the bottom and right side of the selected object based on your settings.

To place a pattern and color within a bordered item:

- 1 Select the object you want to fill with a pattern.
- 2 Right-click the mouse and choose Border from the pop-up menu, or choose the Border command on the Format menu. The Borders dialog box appears.
- 3 Select the Column button under the Apply To options if you've selected a column.
- 4 Set the Border Type, Line Type, Style, Thickness, Standoff, and Shadow Options.
- 5 Click the Custom button in the Patterns option. The controls for selecting a pattern, and a color to use with the pattern, appear.
- 6 Select the pattern you want from the drop-down list box.
- 7 Select a color from the Foreground list box. Use this feature to determine how dark or light the shadow will be.
- 8 Press OK. ReportSmith fills the object you selected based on your settings in the dialog box.

Inserting fields

You can place report fields into your report using the Insert | Insert Field command. Report fields include fields that you create such as *derived fields*, *report variables* and *summary fields*; fields included with ReportSmith for your use called *system fields*; and each of the *data fields* in the table(s) to which your report is connected.

You can choose from a list of Fields in the Insert Database Field dialog box. The list is shown below:

Data Fields:
Derived Fields:
Report Variables:
System Fields:
Summary Fields:

A list of fields in each category appear in the dialog after you select a category. For example, if you select Data Fields, a list of the available fields from each database table you are connected to appears in the dialog box. If you choose Derived Fields, a list of each derived field you created for this report appears in the dialog box.

Note If you haven't created any fields such as derived fields, report variables, and summary fields the list appears blank.

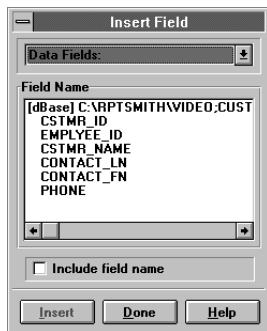
There are three ways to insert a field using the Insert Field dialog box:

- Select the field and press the Insert button, and click on the report to place the field.
- Select the field and while holding the mouse button, drag the field off the dialog box and onto the report.

Note You can use this method for all types of reports except crosstab reports.

Data fields

Data Fields are the columns of information contained in each of your tables. You can insert a data field into your report using the Insert Field command on the Insert menu.



Derived fields

You can insert derived fields that you create using the Insert Field command on the Insert menu. A *derived field* is a field you have created using SQL code or the ReportBasic macro language.



Report variables

You can insert report variables that you create using the Insert Field command on the Insert menu. A *report variable* appears in your report as a dialog box and can prompt a user for information.



System fields

A *system field* is one that has been created by ReportSmith for your use, such as the date, time or page number. ReportSmith offers you a selection of system fields.

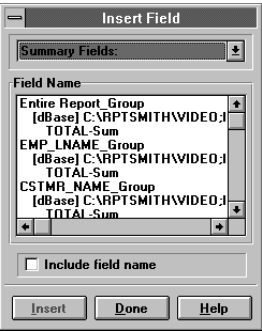


The types of system fields are described in the following table:

System Field	Description
Today's Date	Prints today's date.
Current Time	Prints the current time as it's set in your computer.
Today's Date and Current Time	Prints both today's date and the current time as they're set in your computer.
Page Number	ReportSmith automatically inserts page numbers for you; however, if you want to insert page numbers yourself, choose this command.
Text Fields	Select this field name to insert text into your report. It is similar to using Text Mode on the toolbar.
Report Name	If you select Report Name, ReportSmith inserts the report name as it appears in the title bar of your report. The title is entered in the following format, including the drive, directory path and name of your report: C:\RPTSMITH\VIDEO\CUSTOMER.RPT

Summary fields

You can insert summary fields that you create by using the Insert | Insert Field command. A *summary field* is a calculated field such as a subtotal or count.



If you place a summary field into a group header or footer, ReportSmith calculates the field based on that group. If you drop a summary field into a page header or footer, ReportSmith calculates the summary field based on the records per page and displays a summary on each page.

To insert a summary field into a group:

- 1 Choose Insert | Insert Field. The Insert Database Field dialog box appears.
- 2 Press the down arrow in the list box and select the appropriate field category. A list of data fields from each added database table appears for you to choose from.
- 3 Select a field name to insert from the Field Name list and press Insert.
- 4 Position the Object cursor in a group header or footer and click the mouse. The data field appears on your report, calculating a summary for each group.

Note If you decide to change the field to base it on a page, right-click the mouse and choose Calculate Over Page from the pop-up menu.

To insert a summary field for each page:

- 1 Follow the above steps for inserting a field into your report.
- 2 Position the Object cursor in a page header or footer and click the mouse.

Note If you decide to change the field to base it on a group, right-click the mouse and choose Calculate Over Group from the pop-up menu.

Inserting field labels

ReportSmith does not automatically add labels after grouped data unless you tell it to. You can control the recurrence of field labels in a report. ReportSmith automatically places field labels above columns at the top of each report page. You can turn field labels on or off with the Field Labels command on the Insert menu. To insert field labels at the top of each report page:

- 1 Choose Insert | Field Labels. The Insert Field Labels dialog box appears.



- 2 Click the New Page check box and an X appears. The field labels appear at the top of each report page and print with the report. (Click the check box again and the X disappears. This turns off field labels so they don't appear at the top of each page.)

To insert field labels before or after grouped data:

- 1 Choose Insert | Field Labels. The Insert Field Labels dialog box appears.

- 2 Click the Group Header/Footer check box to place an X in the check box. The field labels appear after any set of group headers and footers that appear before a detail section.

Note Click again to delete the X in the check box and the field labels turn off.

Placing items into a header or footer

After you create headers and footers, you can use the Insert | Field command to place items into them. Alternatively, you can use Form mode to move data values into them. You can insert all types of fields, dates, crosstabs and OLE objects including graphs or pictures.

You can place several types of headers or footers in a report. The following table describes the header or footer types ReportSmith uses:

Header/Footer type	Placement
Page	Places the header or footer at the top or bottom of each report page.
Entire Report Group	Places the header or footer at the top of the first report page or the bottom of the last report page. Actions performed with the Entire Report Group affect all fields in your report.
Group	Places the header or footer above or below a selected group of data in your report.

- Choose Insert | Field to place an item into a header or footer.
- To align items in a header or footer, select them using Form Mode and use the Alignment toolbox.

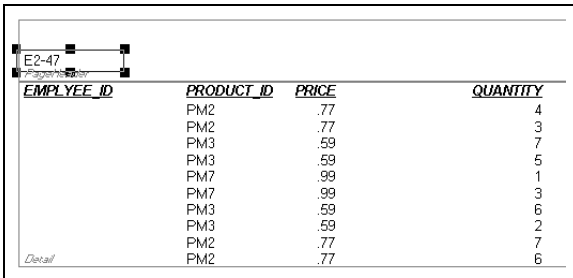
To place a field into a header or footer:

To move data values into a header or footer section, use Form mode. When you move a column into a header or footer, only the first value of that column is displayed in that section; all others are suppressed.

- 1 Select a column. Press the Form Mode button.
- 2 Select the first field (it has sizing handles on it), hold the mouse and drag it up into the page header or down into the page footer.

<i>PageHeader</i>			
<u>EMPLOYEE_ID</u>	<u>PRODUCT_ID</u>	<u>PRICE</u>	<u>QUANTITY</u>
E2-47	PM2	.77	4
E2-47	PM2	.77	3
E2-47	PM3	.59	7
E2-47	PM3	.59	5
E4-62	PM7	.99	1
E4-62	PM7	.99	3
E2-30	PM3	.59	6
E2-30	PM3	.59	2
E2-30	PM2	.77	7
E2-30	PM2	.77	6

ReportSmith places the column into the header or footer.



<i>EMPLOYEE_ID</i>	<i>PRODUCT_ID</i>	<i>PRICE</i>	<i>QUANTITY</i>
PM2		.77	4
PM2		.77	3
PM3		.59	7
PM3		.59	5
PM7		.99	1
PM7		.99	3
PM3		.59	6
PM3		.59	2
PM2		.77	7
PM2		.77	6

Resizing field and record height

ReportSmith lets you include fields of different heights in your report. You can “shrink and grow” each record to include all of a field’s data upon request. For example, suppose your table contains a field that lists a customer’s comment about a video rental. Each customer has different comments of different lengths. You can tell ReportSmith to shrink and grow each comment field in your table so that it fits the data contained within it, instead of reserving a fixed length for each record.

Using Shrink and Grow

When you use ReportSmith’s Can Grow feature in a Columnar report, longer lines of text automatically wrap two or more lines. Fields wrap to the next line at a space so that a single word doesn’t get cut off. If you use Can Shrink with a very long line of text, the text is truncated. A field can be fixed in height, can grow, can shrink, or can shrink and grow.

To shrink and grow:

- 1 Open a report and select a column which you would like to be of variable height.
- 2 Choose Format | Field Height, or right-click and choose Field Height from the pop-up menu. The Field Height dialog box appears.
- 3 Choose an option in the dialog box. To allow the field to increase in height from the default height, choose Can Grow. To allow the field to decrease in height from the default height, choose Can Shrink. To allow the field to fit the data contained within the field, turn on both options.



When you use the Field Height or Row Height command, the Repaginate button appears on the toolbar. You can press the button to automatically repaginate your report after you’ve resized fields or rows. (This button only appears when you’re using these commands.)

Specifying row height

You can also control the height of a row. A row can be a fixed height, can grow, can shrink, or it can shrink and grow according to the default field height and the data contained within the row. If you choose the Can Grow option for a field, by default ReportSmith flags the row to grow as well. However, you can fix the height of a row and

still allow a field to grow, allowing you to control the size of row. For example, you can ensure that the row does not exceed a specified size.

To specify the height of a row:

- 1 Choose Format | Row Height or select the row you want to be of variable height, right-click and choose Row Height from the pop-up menu. The Row Height dialog box appears.
- 2 Choose an option in the dialog box. To allow the record to increase in height from the default height, choose Can Grow. To allow the record to decrease in height from the default height, choose Can Shrink. To allow the record to increase or decrease in height from the default height, turn on both options.

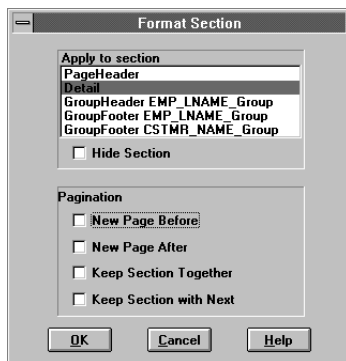
Choosing items to display and print

You can determine data, headers, and footers display and print in a report by setting options with the Section command on the Format menu. Use this command to hide sections of a report and to paginate a report. A section is defined as either a detail or as a header or footer. Refer to “Paginating a report,” later in this chapter, for instructions on inserting page breaks into a report.

For example, you can create an employee report that groups data by department. You drop a summary field into the department group footers that calculates the sum of salaries for each department. With the Section command on the Format menu, you can hide all data in the report, except for the group footers. ReportSmith then displays only the department group footers with the total salaries for each department.

Hiding a section of a report

- 1 Choose Format | Section. The Format Section dialog box appears.



- 2 From the Apply to Section box, select the section with data you want to hide.

All the headers or footers in your report are listed. The Detail section also appears in the list box.

- To move it, point in the title bar and drag it into place.

The Align toolbox

Each button on the Align toolbox is described below.

Button	Function
	Align to the top-most field.
	Align to the center of the selected group.
	Align to the bottom-most field.
	Align to the left-most field.
	Vertically align to the centerpoint of the selected fields.
	Align to the right-most field.
	Reduce vertical space equally between selected fields. Each time you press the button, ReportSmith moves fields in a pixel at a time.
	Space fields equally horizontally within the selected area.
	Add equal horizontal space between selected fields. Each time you press the button ReportSmith adds one pixel between each field.
	Space selected items closer together vertically.
	Distribute fields equally in a given vertical space.
	Add space between stacked fields or columns vertically.
	Align fields to a common baseline. The baseline is the line the letters rest on as opposed to the text or column boundary.

Button	Function
--------	----------



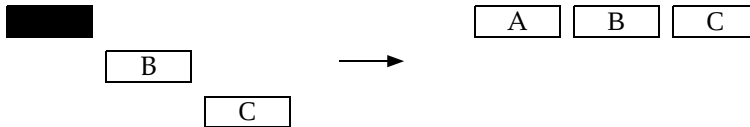
Stack fields horizontally in a row aligned in relation to the baseline of the lowest column you selected. This function is order-dependent.



Stack fields vertically in a column in relation to the boundary of the right-most column you selected. This function is order-dependent.

To align objects to the top:

- Click the left-mouse button and drag the rubber band around all objects in a section. If you select objects in a global section (for example, a page header or footer or a row), you also select all objects in like sections. For example, if you select all objects in one page header, you also select objects in all page headers. The alignment function you choose applies to them as well.
- Click to select an object you want to align to the top. *Ctrl*+click to select additional objects you want to align. If you only select one object, it aligns to the top of the section.
- Select an entire section and you select all objects in the section.
- Choose the Align to Top button on the Align toolbox. ReportSmith moves all selected objects (B, C) so they align with the uppermost selected object (A).



To center objects:

1 Do one of the following:

- Click the left-mouse button and drag the rubber band around all objects in a section you want to align.

Note

If you select objects in a global section—a page header or footer, a row, and so on—you also select all objects in sections of the same type. For example, if you select all objects in one page header, you also select objects in all page headers and the alignment function you choose applies to them as well,

- Click to select an object that you want to center vertically. Then press *Ctrl*+click to select additional objects,

Note

If you only select one object, it aligns to the vertical center of the section it's placed in.

- Select an entire section and you select all objects in the section.



2 Press the Align to Vertical Center button.

ReportSmith moves all selected objects (A, B, C) so they're placed on the same horizontal plane, based on the center object in a group of selected objects.



To align objects to the bottom:

1 Do one of the following:

- Click the left-mouse button and drag the rubber band around all objects in a section you want to align.
- Click to select an object you want to align to the bottom. Then press *Ctrl*+click to select additional objects,

Note If you only select one object, it aligns to the bottom of the section it's placed in.

- Select an entire section and you select all objects in the section.



2 Press the Align to Bottom button. ReportSmith moves all selected objects (A, B) so they align with the lower-most selected object (C).



To left-align objects:

1 Do one of the following:

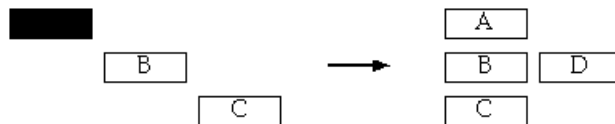
- Click the left-mouse button and drag the rubber band around all objects in a section you want to align,
- Click to select an object you want to align to the left. Then press *Ctrl*+click to select additional objects,

Note If you only select one object, it aligns to the left of the section it's placed in. If you select a column or a row and another object, the column or the row is the anchor and doesn't move. The other object aligns with the left edge of the column or row.

- Select an entire section and you select all objects in that section.



2 Press the Align to Left button. ReportSmith moves all selected objects (B, C) so they align with the left-most selected object (A). The left-most selected object serves as the anchor and doesn't move.



To center objects horizontally:

1 Do one of the following:

- Click the left-mouse button and drag the rubber band around all objects in a section you want to align,
- Click to select an object that you want to center horizontally. Then press *Ctrl*+click to select additional objects,

Note

If you only select one object, it aligns to the horizontal center of the section it's placed in. If you select a column or row and another object, the column or row is the anchor and doesn't move. The other object aligns to the horizontal center of the column or the row.

- Select an entire section and you select all objects in that section.



2 Press the Align to Horizontal Center button. ReportSmith moves all selected objects (A, B, C) so they're centered horizontally.



To right-align objects:

1 Do one of the following:

- Click the left-mouse button and drag the rubber band around all objects in a section you want to align,
- Click to select an object you want to align to the right. Then press *Ctrl*+click to select additional objects,

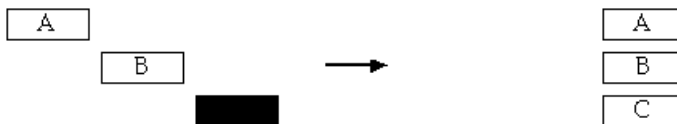
Note

If you only select one object, it aligns to the right of the section it's placed in. If you select a column or a row and another object, the column or row becomes the anchor and doesn't move. The other object aligns with the right edge of the column or row.

- Select an entire section and you select all objects in that section.



2 Choose the Align to Right button on the Align toolbox. ReportSmith moves all selected objects (A, B) so they're aligned with the right-most selected object (C). The right-most object serves as the anchor and doesn't move.



To decrease horizontal space among objects:

1 Click the left-mouse button and drag the rubber band around all objects in a section you want to align, or click to select the left-most object. Press *Ctrl*+click to select additional objects you want to move closer to the left-most selected object.



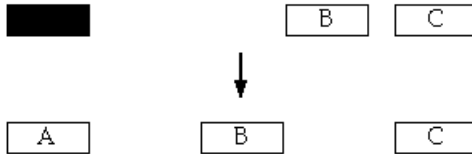
- 2 Press the Decrease Horizontal Space button. ReportSmith moves all selected objects toward the left-most selected object one pixel for each click of the button. The left-most object serves as the anchor and doesn't move.

To equalize horizontal space among objects:

- 1 Click the left-mouse button and drag the rubber band around all objects in a section you want to align, or click to select the left-most object that you want to space. Press *Ctrl*+click to select additional objects.



- 2 Press the Equalize Horizontal Space button (shown below). ReportSmith adjusts all selected objects (A, B, C) so they're equally spaced. The left-most object (A) serves as the anchor and doesn't move.



To increase horizontal space among objects:

- 1 Click the left-mouse button and drag the rubber band around all objects in a section you want to align, or click to select an object from which you want to move other objects away. Press *Ctrl*+click to select additional objects.



- 2 Choose the Increase Horizontal Space button on the Align toolbox. ReportSmith moves all selected objects away from the left-most object, one pixel for each click. The left-most object serves as the anchor and doesn't move.

To decrease vertical space among objects:

- 1 Click the left-mouse button and drag the rubber band around all objects in a section you want to align, or click to select an object to which you want to move other objects closer. Press *Ctrl*+click to select additional objects.



- 2 Choose the Decrease Vertical Space button on the Align toolbox. ReportSmith moves all selected objects up, toward the upper-most selected object, one pixel for each click. The upper-most object serves as the anchor and doesn't move.

To equalize vertical space among objects:

- 1 Click the left-mouse button and drag the rubber band around all objects in a section you want to align, or click to select an object that you want to space. Press *Ctrl*+click to select additional objects.



- 2 Press the Equalize Vertical Space button on the Align toolbox. ReportSmith adjusts all selected objects (A, B, C) so they're equally spaced. The upper-most object (A) serves as the anchor and doesn't move.



To increase vertical space among objects:

- 1 Click the left-mouse button and drag the rubber band around all objects in a section you want to align, or click to select an object from which you want to move other objects away. Press *Ctrl*+click to select additional objects you want to move away from the upper-most selected object.



- 2 Choose the Increase Vertical Space button on the Align toolbox. ReportSmith moves all selected objects down from the upper-most object, one pixel for each click. The upper-most object serves as the anchor and doesn't move.

To align text to a common baseline:

- 1 Click to select the first field you want to align to a common baseline. This field serves as the anchor.
- 2 Press *Ctrl*+click to select additional fields you want to align to a common baseline.



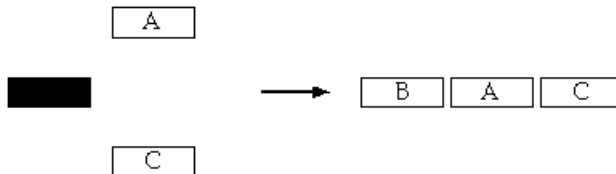
- 3 Press the Baseline button. ReportSmith aligns all selected fields to the font baseline of the anchor field.

To align objects in a row:

- 1 Click to select the object you want to align all other selected objects with, to create a row. This object serves as the anchor and doesn't move.
- 2 Press *Ctrl*+click to select any other fields you want to align in the row.



- 3 Choose the Row button on the Align toolbox. ReportSmith aligns the selected objects (A, C) to the right of the object serving as anchor (B). It places the objects in the row in the order you select them, and each object touches the one next to it.



To stack objects:

- 1 Click to select the object under which you want to stack other objects. This object serves as the anchor and doesn't move.

2 Press *Ctrl*+click to select additional objects you want to stack.



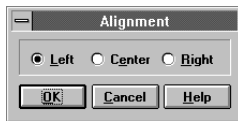
3 Choose the Stack button.

Before stacking objects enlarge the row or section so it is large enough to contain the set of stacked objects. ReportSmith stacks all selected objects (B, C) underneath the anchor object (A), in the order they're selected, so that each object touches the one above it.



Aligning text

Alignment refers to the placement of the field value within the space allotted for the field on the report. ReportSmith lets you align text with the Text Alignment toolbox. To access this toolbox, select the text you want to align, right-click the mouse and choose Text Alignment from the pop-up format menu. Choose Text Alignment and the toolbox appears.



To align text:

- 1 Select the text you want to align.
- 2 Right-click the mouse. A pop-up format menu appears.
- 3 Choose Text Alignment. The Text Alignment toolbox appears.

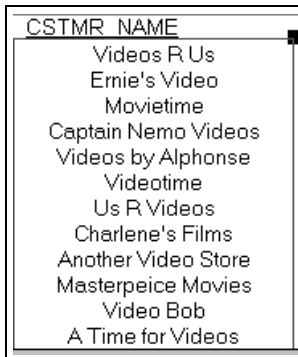
Aligning left

Click Left to place all field values flush left in the space allotted. The first character in the value is flush against the left margin of the field border. When you select Left, ReportSmith aligns the first character in each field:

<u>CSTMTR_NAME</u>
Videos R Us
Ernie's Video
Movietime
Captain Nemo Videos
Videos by Alphonse
Videotime
Us R Videos
Charlene's Films
Another Video Store
Masterpeice Movies
Video Bob
A Time for Videos

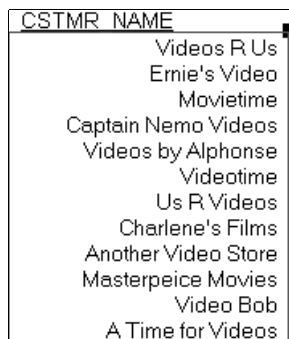
Aligning center

Click Center to center the values within the space allotted by the border:



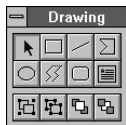
Aligning right

Click right to place all values flush right in the space allotted. The last character in the value is flush against the right margin of the field. When you choose Right, ReportSmith aligns the last character in each value:



Drawing

ReportSmith's Drawing toolbox helps you accurately draw commonly used shapes such as lines, circles, boxes and polygons; you can position items in back or in front of each other. You can type text into your drawings.



To open the Drawing toolbox:

- Choose Tools | Drawing. The Drawing toolbox appears.
- To move the toolbox, point the cursor in the blue title bar of the toolbox and drag it to the new location.

The Drawing Toolbox

Each tool on the Drawing toolbox is described below.

Name	Icon	Function
Arrow		Select an object.
Box		Draw a box. When the cross-cursor appears, select a starting point and hold the left mouse button. Drag the cursor until the box is the size you require and release the mouse button.
Line		Draw a line. When the cross-cursor appears, select a starting point and hold the left mouse button. Drag the cursor until the line is the size you want, then release the mouse button.
Polygon		Draw a multi-sided figure. When the cross-cursor appears, select a starting point and hold the left-mouse button. Drag the cursor until the side is the desired length and release the mouse. Left-click and drag the cursor to the next endpoint or move the cursor to the desired endpoint and left-click. To complete the figure, click the starting point or select another tool, or double-click on the final point to automatically connect it to the starting point.
Round Rectangle		Draw a box with rounded corners. When the cross-cursor appears, select a starting point and hold down the left mouse button. Drag the cursor until the box is the size you require and release the mouse.
Circle		Draw a circle. When the cross-cursor appears, select a starting point and hold the left mouse button. Drag the cursor until the circle is the size you require and release the mouse.
Polyline		Draw a polyline figure. When the cross-cursor appears, select a starting point and hold down the left-mouse button. Drag the cursor until the side is the desired length, then release the mouse. Left-click and drag the cursor to the next endpoint or move the cursor to the next desired endpoint and left-click the mouse. To complete the figure, click the starting point or select another tool, or double-click on the final point.
Text		Insert text into a report. When the Text cursor appears, position it where you want to place text and left-click the mouse. Type in the text. When you've finished entering text, click the Text cursor outside the text field or select another tool.
Group		Group selected objects together. Select the objects you want to group (click the first object; Ctrl+click to select additional objects.) Click the Group tool and the objects group automatically. To select one object within the group, click any object to select the group, then click on the object you want to select.

Name	Icon	Function
Ungroup		Ungroup objects and select them individually. Click any object within the group to select it, then click the Ungroup tool. Objects can now be selected individually.
Move to Front		Move a selected object to the front of other objects. Click the object you want to move to the front, then click the Move to Front tool to place the selected object on top of all other objects placed in the same position. If you have several layers of object you may need to press this button more than once.
Move to Back		Move a selected object to the back of other objects. Click the object you want to send to the back, then click the Move to Back tool to place the selected object behind all other objects that are placed in the same position. If you have several layers of object, you may need to press this button more than once.

Using report styles

Every ReportSmith report has a style attached to it. A *style* is a collection of character and section formats that you've given a name. Suppose you've formatted a report to look the way you want. You've changed the font and point size, italicized the title, and placed colored borders around selected fields to emphasize them. Now you want to apply this same style to another report. You could create a new report, then reapply each formatting change to it, but that's extra work. Furthermore, if you decide to change the formatting, you'd have to repeat the formatting steps for each report.

Instead, you can save the formatting as a style by selecting the report you've formatted and assigning a name to it in Report Styles dialog box. After you save a style, it appears in the Custom Styles list. To apply the style to a new report just select it from the list. You'll choose a style in the Report Style dialog box each time you create a new report.

You always choose a style when you first create a report, however you can change the style using the Report Styles dialog box at any time during report creation.

Use styles to

- Ensure consistent formatting throughout your reports.
- Save time by quickly applying instant formatting changes.
- Make design changes easy to incorporate.

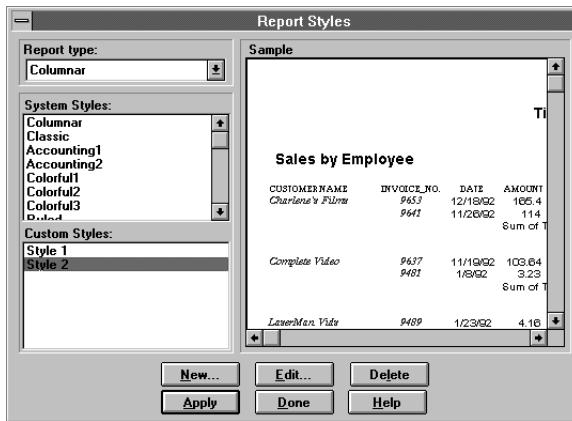
To distribute a report to another user, you must save your style as a system style, then distribute the system file to that user.

Applying a style

Use the Report Styles dialog box to choose a report style, save or edit a style, assign a default style to a report, and remove styles you no longer use.

To apply a style :

- 1 Open a report and press the Style button. The Report Styles dialog box appears.



- 2 Choose a style from the System Styles or Custom Styles list boxes and choose Apply.
- 3 Select a style and preview it in the Sample box. Choose Apply.
- 4 To save a style, choose New and enter a name for it.

Assigning a default style

You can assign a style to a report type you use often.

When you create a new report, ReportSmith displays the Create A New Report dialog box. The default report style appears beside Style.



- To use the default style already assigned, select a report type and press OK and create your report.
- To change the style, choose Style and select a style in the New Report Style dialog box, and choose Use As Default. The default style appears in the Create A New Report dialog box when you select the report type to which it has been assigned.

Creating a style

You can save styles you use frequently and apply them instantly. Styles that you create appear in the Custom Styles list. If you've already formatted a report, most of the work

to create a style is behind you. Now all you need to do is assign names to report formats you like. Alternatively, you can customize a system style.

To save a style:

You can save the current formatting of a selected report as a new style.

- 1 After applying formatting elements to your report, press the Styles button.
- 2 Choose New and type a name for your style into the Save As dialog box. If you want, you can use the default style name ReportSmith assigns, such as "Style 1", "Style 2", and so on. Style names can contain up to 24 characters and can include any combination of characters and spaces. Style names are not case-sensitive; ReportSmith treats "Summary" and "summary" the same. The name you assign a style must be unique for a particular report. For example, a crosstab can only have one style named "Financial."
- 3 Press OK to add your style to the Custom Styles list, and press Apply to apply it to your report.

Modifying a style

After you've defined a style, you can modify it and save it to another style. Redefining a style updates that style only in the active document, leaving those reports which you've previously formatted with that same style untouched. To modify other reports that use the same style, open each report and apply the redefined style.

ReportSmith offers you two methods for modifying a report style:

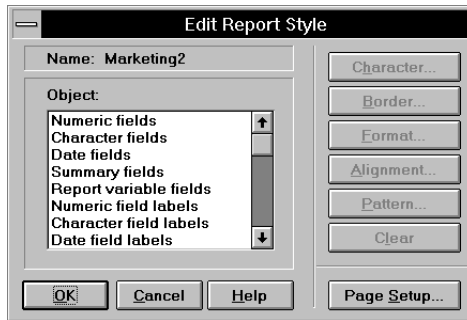
- Modify formatting on the report and save the style
- Use the Edit Report Style dialog box

To modify a style on the report surface:

- 1 Open a report with a style you want to modify.
- 2 Make formatting changes.
- 3 Press the Report Styles button, choose New and type a name for your style into the Save As box. Press OK to exit.

To modify a style with the Edit Report Style dialog box:

- 1 In the Report Styles dialog box, select the style you want to modify and press Edit. The Edit Report Style dialog box appears.



- 2 Scroll through the Object list, select the type of object you want to edit, and choose a formatting button to select options in the associated dialog box. Set options and press OK, then press OK again to exit the Edit Report Style dialog box. Choose Apply.

ReportSmith applies the style to the report.

Deleting a style

You can delete a custom style, but this affects reports which have been assigned that style. ReportSmith alerts you with a message reminding you that removing the style affects any reports with that style. When other reports are affected, you replace their styles with the basic, non-formatted style. You cannot delete a system style.

- 1 In the Report Styles dialog box, choose the style you want to delete.
- 2 Press Delete. ReportSmith displays a message asking you to confirm that you want to delete the style.
- 3 Press Yes to delete the style, No to cancel the delete.
- 4 Choose Close to exit the Report Styles dialog box.

Using the Style Extractor



If you've formatted an item in your report, you can apply that style to other areas within the report with the Style Extractor. Use it to extract styles you select and instantly apply them to the item that you click on.

- 1 Select a field or column whose formatting you want to copy.
- 2 Press the Style Extractor button. The Style Extractor cursor appears.
- 3 Move the cursor over a value or column and click the mouse. ReportSmith applies the style to the selected item.

This example shows how you can apply three formatting attributes (bold, no underline, and larger font size) to other fields in your report with just a click.

Extract this style 

INVOICE_ID	DATE	TOTAL
9478	1/2/92	\$11
9479	1/3/92	\$3
9481	1/8/92	\$3
9482	1/9/92	\$13
9483	1/10/92	\$5
9484	1/14/92	\$12
9487	1/21/92	\$5
9489	1/23/92	\$4
9490	1/27/92	\$43
9491	1/28/92	\$22
9492	1/29/92	\$42
9496	2/5/92	\$98
9499	2/12/92	\$21

...and apply it to these fields  

INVOICE_ID	DATE	TOTAL
9478	1/2/92	\$11
9479	1/3/92	\$3
9481	1/8/92	\$3
9482	1/9/92	\$13
9483	1/10/92	\$5
9484	1/14/92	\$12
9487	1/21/92	\$5
9489	1/23/92	\$4
9490	1/27/92	\$43
9491	1/28/92	\$22
9492	1/29/92	\$42
9496	2/5/92	\$98
9499	2/12/92	\$21

Displaying data fields as pictures

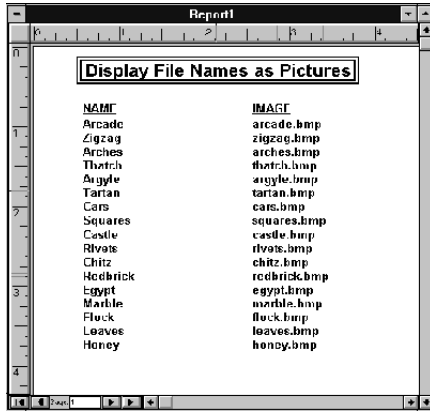
Suppose you have an employee table that contains pictures of your employees. File names are stored in a column in the table point to the directory paths and names of the graphics files. When you view the table in ReportSmith, file names appear.

To display data fields as pictures:

- 1 Choose Tools | Options to see the Options dialog box.
- 2 In the Picture File Search Path, enter the directory path where the graphics files are located. This is not needed if the full path name is included for each graphic filename in the table.
- 3 Create a new report and open the Tables dialog box.
- 4 Press the Add Table button. The Select Table To Be Added dialog box appears.
- 5 Choose the table containing the fields with graphic file names and press OK to add it to your report. The graphics file names appear on your report.
- 6 Select the column containing graphics file names and choose the Format | Display As Picture command from the Format menu. The Display as Picture dialog box appears.
- 7 Select the options you want and press OK. File names in your report now appear as graphics.

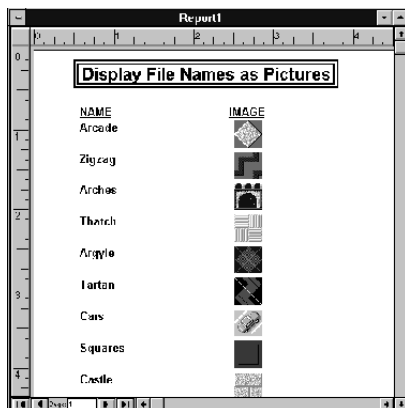
Note ReportSmith takes the first row encountered and shrinks or expands the remaining rows based on the height of the first row. You can adjust all of the rows by resizing the row, then the picture.

The following example report contains two columns. The first, NAME, contains the name of a picture (or .BMP) file. The second, IMAGE, contains the file name of each picture. Here, we'll show you how to display the file names in the IMAGE column as pictures.



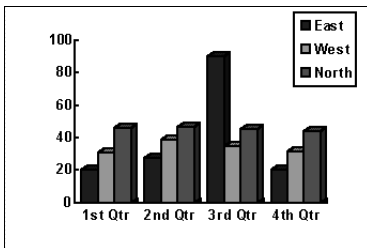
To transform file names into pictures:

- 1 Select the column containing the picture files. In this case, we'll select the IMAGE column.
- 2 Right-click and choose Display as Picture. The Display As Picture dialog box appears.
- 3 From the Display Field As list, select the type of image file. Our files are *bitmaps*, or *.BMP files*, so we have selected Windows Bitmap.
- 4 Adjust the Initial Scale option to scale the size of the picture that display on your report and press OK. ReportSmith locates each file name on the system and displays the picture stored inside it.



Graphs

If you have Microsoft Graph installed on your system you can instantly transform report data into graphs. You can create any type of graphs including pie, line, area, column and combination graphs which you can format to suit your needs.



Instantly transform data
into a graph

There are three ways to create a graph:



- Select up to three columns and press the Graph button.
- Press the Graph button to open Microsoft Graph and define the graph from scratch.
- For a crosstab press the Graph button or right-click and select Chart.

Converting a columnar report to a graph

To create a graph using the point and click method:

- 1 Select one, two, or three columns of data in a report.
- 2 Press the Graph button.

You can cancel creating a graph by clicking on the left-mouse button when the No-drop cursor appears or by pressing the ESC key.

- 3 Move your cursor within the boundaries of a header or footer. The Graph cursor appears.

- 4 Use the mouse to position the Graph cursor in the header or footer where you want to place the graph. Click the left mouse button to place the graph.

To create a graph using Microsoft Graph:

- 1 Make sure nothing is selected in your report and press the Graph button.
- 2 Click in the header or footer to place the graph. The graph application opens.
- 3 Create a graph in Microsoft Graph. (Refer to your Microsoft Graph documentation.)
- 4 When you've finished, choose the Update command on the application's File menu. This places the graph below the point where you clicked in your report. For OLE 2.0, click outside the graph.
- 5 To return to ReportSmith choose File | Exit and File | Return to Report, or click outside the window.

To reposition a graph within a header or footer:

- 1 Single-click the graph.
- 2 Drag it to a new location within a header or footer and release the mouse. ReportSmith repositions the graph in the new location.

To modify a graph:

- 1 Double-click the graph. The Graph tool opens. For OLE 2.0, the menus and toolbars change to show Graph options.
- 2 Modify the graph.
- 3 Do one of the following:
 - Select the Update command on Graph tool's File menu when you've finished editing the graph and then choose the Exit command on the File menu to return to ReportSmith.
 - Select the Exit and Return to Report command on the graph application File menu to cancel or save your changes when prompted and return to ReportSmith.

Note You cannot close your ReportSmith document until you exit Microsoft Graph.

To delete a graph:

- 1 Select the graph you want to delete.
- 2 Press the *Delete* key. ReportSmith deletes the graph from your report.

To create a graph from a crosstab:

- 1 Select a crosstab.
- 2 Press the Graph button on the toolbar or right-click and select Chart from the pop-up menu.
- 3 Single-click in a header or footer to place the graph in your report.

Converting a crosstab to a graph

To create a graph based on a crosstab report:

- 1 Select the crosstab and Tools | Crosstab to display the Crosstab toolbox.
- 2 Press the Graph button.

When you move your cursor within the boundaries of a header or footer, the Graph cursor appears.

- 3 Drag the mouse to position the Graph cursor in the header or footer where you want to place the graph and left-click the mouse. ReportSmith places the graph below the point where you press.
- 4 Click to select the header or footer if the graph exceeds its boundaries. A box with sizing handles surrounds the header or footer.
- 5 Drag a sizing handle until the entire graph falls within the boundaries, then release the mouse button.

Placing a graph

Where you place the graph controls what data is included and how often it appears in your report. If you drop the graph into a global region like a page header or report footer, this creates the graph using all the data in the report and ReportSmith places it in every page or in the report header or footer. If you drop it into a group header or footer, the Graph tool creates a graph based on the data in just that group and ReportSmith places it into that group header or footer. ReportSmith positions the graph's top left corner where you click the mouse.

Crosstabs

With crosstabs, you can instantly summarize your data in a concise spreadsheet-like layout. You can create a crosstab directly or generate one from an existing columnar report. Each time you open your report, ReportSmith automatically updates the crosstab if data changes.

You can instantly apply formatting styles, resize individual cells, drag and drop fields and labels, automatically change the value hierarchy, drop fields directly into your crosstab, use in-cell editing to change total and group headings, and more.

ReportSmith lets you create a crosstab with three methods:

- Create a new crosstab and choosing Crosstab from the Create A New Report dialog box
- Create a crosstab based on an existing columnar report and specify rows, columns and values in the Crosstab Report dialog box
- Click to select columns in your columnar report and press the Crosstab button to instantly transform them into a crosstab

Parts of a crosstab

Crosstab reports can consist of any number of rows, columns, and values.

Row labels		Column labels				Row
		1st Qtr	2nd Qtr	3rd Qtr	4th Qtr	Total
Blake	Los Altos	\$152.29	\$448.60	\$480.72	\$559.28	\$1,640.89
	San Mateo	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00
	Woodside	\$73.05	\$229.52	\$111.08	\$424.41	\$838.06
	Total	\$225.34	\$678.12	\$591.80	\$983.69	\$2,478.95
Keats	Los Altos	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00
	San Mateo	\$162.19	\$154.85	\$319.32	\$207.27	\$843.63
	Woodside	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00
	Total	\$162.19	\$154.85	\$319.32	\$207.27	\$843.63
Total		\$387.53	\$832.97	\$911.12	\$1,190.96	\$3,322.58

Types of crosstabs

You can create numerous types of crosstabs, combining multiple rows, multiple columns, and multiple values. Some common variations are

- Value-only
- Single-row
- Single-row/single-column
- Single-column
- Double-column
- Double-row

Value-only

The Value-only crosstab consists of only two cells: the value field label and the grand total value.

TOTAL
\$3,489

This report creates a summary total for one data field. The data set can be an entire report if you place the crosstab in a global header or footer (that is, a page or report) or it can be a section of the data if you place it in a group header or footer. You can make this type of crosstab by only specifying one data field as a value in the crosstab.

Single-column

The Single-column crosstab consists of a column and values.

EMP_LNAME		
Blake	Keats	Total
TOTAL	TOTAL	TOTAL
\$2,603	\$886	\$3,489

It provides summary totals for each field within the column, as well as a grand total of all value fields. The only difference between this type of report and the Single-row crosstab is the print direction of the data.

Single-row

The Single-row crosstab consists of a row and values.

EMP_LNAME	TOTAL
Blake	\$2,603
Keats	\$886
Total	\$3,489

It provides summary totals of each unique value in a row, as well as a grand total of all value fields. The only difference between this type of report and the Single-column crosstab is the layout of the data.

Single-row and single-column

The Single-row and single-column crosstab provides another level of detail. It includes a row, a column, and a value. Each summary value is a total based on the row as a subset of the column.

		Column				
		CSTMN_NAME				
		Charlene's Films	Complete Video	LaserMan Vids	Movie Madness	Total
Row	EMP_LNAME	TOTAL	TOTAL	TOTAL	TOTAL	TOTAL
	Blake	\$1,033	\$690	\$880	\$0	\$2,603
	Keats	\$0	\$0	\$0	\$886	\$886
	Total	\$1,033	\$690	\$680	\$886	\$3,489
		Column grand totals				
						Row grand totals
						Report grand total

In the example above, CSTMR_NAME is the column and EMP_LNAME is the row. The values within the crosstab are Blake's total sales for each customer, and Keat's total sales for each customer. There are also column grand totals and row grand totals and a report grand total that combines the column and row grand totals.

For example, the row totals are the total values for each employee across all customers. The column totals are the total values for each customer for all salespeople (EMP_LNAME.) The report grand total adds all the row grand totals and all the column grand totals.

Double-column

The Double-column crosstab provides an extra level of information within a column. It consists of two columns and values.

DATE,EMP_LNAME					
1st Ctr	2nd Ctr		3rd Ctr		
Blake	Keate	Blake	Keate	Blake	Keate
TOTAL	TOTAL	TOTAL	TOTAL	TOTAL	TOTAL
\$237	\$170	\$112	\$163	\$621	\$335

The secondary column you select is a subset of the primary, or first, column you select. Therefore, the value of each field within a secondary column is calculated based on the value of each field within the primary column. These can be thought of as grouping levels in a columnar report.

In this example, DATE is the *primary* column and EMP_LNAME is the *secondary* column. The grand total adds the column totals to come up with total sales for both salespeople.

Double-row

The Double-row crosstab provides an extra level of information within a row. It consists of two rows and values.

DATE	EMP_LNAME	TOTAL
1st Qtr	Blake	\$237
	Keats	\$170
2nd Qtr	Blake	\$712
	Keats	\$163
3rd Qtr	Blake	\$621
	Keats	\$335
4th Qtr	Blake	\$1,033
	Keats	\$218
Total		\$3,489

The second row you select is a subset of the primary, or first, row you select. This means that the value of each field within a secondary row is calculated based on the value of each field within the primary row.

In the example above, DATE is the primary row and EMP_LNAME is the secondary row. The report grand total adds the row totals to come up with total sales for both salesmen.

The Crosstab menu

After you select a crosstab, right-click the mouse to see the pop-up Crosstab menu.

Command	Description
Modify	Select this option to view the Crosstab Report dialog box and to modify your crosstab.
Styles	Choose Styles to create, edit, and apply formatting styles to your crosstab.
Border	Select Border to view the Borders dialog box, where you can customize crosstab borders.
Select All	Choose this option to select everything in your crosstab, including all of the cells, totals, and headings.
Chart	Choose this option to convert the crosstab to a chart using Microsoft Graph.
Invert	Choose Invert to swap rows and columns in your crosstab.
Delete	Select this option to delete your crosstab.
Options	Choose this option to change general crosstab options.

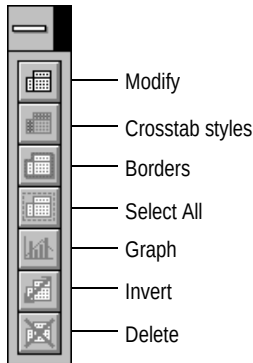
The Crosstab toolbox

The Crosstab toolbox makes it easy to create and modify crosstabs and crosstab styles. To view the Crosstab toolbox, select Crosstab from the Tools menu. Each button on the

toolbox corresponds to actions you can also perform using the Crosstab menu. This section describes the functions of each button on the Crosstab toolbox.

To display the Crosstab toolbox:

- 1 Choose Tools | Crosstab to view the Crosstab toolbox.



Note When you press the Delete button, ReportSmith deletes your crosstab.

Creating a crosstab

There are three methods you can use to create a crosstab:

- Choose Crosstab Report in the Create A New Report dialog box to create a crosstab from scratch.
- Specify rows, columns, and values in the Crosstab Report dialog box based on an existing columnar report.
- Click to select columns within a columnar report and press the Crosstab button on the toolbar. This is called the visual method.

In this section, we'll show you how to create your crosstab using each of the three methods.

Method 1: From scratch

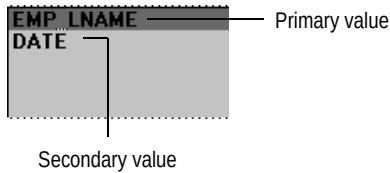
To create a crosstab:

- 1 In the Create A New Report dialog box, choose Crosstab Report.
- 2 Press OK.

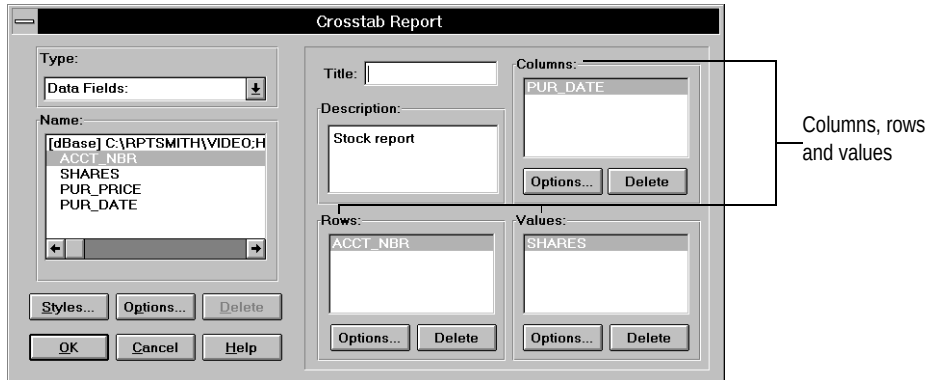
Choose one or more tables for your report and specify any other information in the dialog boxes such as table columns and links, and press Done. The Crosstab Report dialog box appears.

You can place fields that you want to appear in your crosstab by dragging fields from the Name list box and dropping them into the Rows, Values and Columns boxes. If you decide to use multiple values in a section, the order in which you place these

fields is significant; the first field you place in a list box becomes the *primary* value, the second field becomes the *secondary* value, and so on.



- 3 Drag and drop data fields into the Rows, Values and Columns list boxes.



- 4 If desired, specify row, value, or column options by choosing the associated Options button beneath each and press OK.

Depending on the location of your mouse, the cursor becomes either a crosstab or a No-drop cursor.

- 5 Move the cursor over the report header and click the mouse to place the crosstab.

Method 2: Based on a columnar report

To create a crosstab from an existing columnar report:

Use this method if you already have a columnar report on which you'd like to base the crosstab and to predefine specifications such as sorting, value hierarchy, and field labels in the Crosstab dialog box.

- 1 Open a columnar report with a header or footer. Turn on View | Boundaries.
- 2 With nothing in your report selected, choose Tools | Crosstab. ReportSmith displays the Crosstab toolbox.
- 3 Press the Crosstab button. The Crosstab dialog box appears. The data fields in the columnar report appear in the Name list box.
- 4 Drag and drop data fields into the Rows, Values, and Columns list boxes.
- 5 Specify row, value, or column options and press OK.

- 6 Move the crosstab cursor over a header or footer and click the mouse to place the crosstab.
- 7 To hide the columnar report, choose Format | Section. The Format Section dialog box appears.
- 8 Select the Detail section, check the Hide Section box and press OK. ReportSmith suppresses the columnar report.

Method 3: Selecting three columns

To create a crosstab using the visual method:

When using this method, you can select up to *three* columns to create a basic crosstab. The first column you select is the Row, the second becomes the Column, and the third becomes the Value. Selecting two columns creates a single row crosstab. Selecting one column creates a Value-only crosstab.

To choose more than three values for a crosstab, use the Crosstab Report dialog box discussed in the last section.

- 1 Open a columnar report that has a header or footer and display the boundaries.
- 2 Choose Tools | Crosstab to view the Crosstab toolbox.
- 3 Click to select a column in your report you'd like to appear as a crosstab row.
Ctrl+click to select two more columns for the column and values in your crosstab.
- 4 Press the Crosstab button on the toolbox.
- 5 Move the cursor into a header or footer and click to place your crosstab. The crosstab appears on the report.
- 6 To hide the columnar report, use the Format Section dialog box. The columnar report disappears and your report displays just the crosstab.

Placing a crosstab

If you've generated it directly, you can place the crosstab anywhere on the report page. If you've generated the crosstab from a columnar report, when you move over one of these drop zones, you'll see the Crosstab cursor, indicating that you can click to place the crosstab. If you're over a no-drop zone, such as the report detail, the No-drop cursor appears, alerting you that you cannot place the crosstab in the current report section. After you place a crosstab you can move it: double-click to select it, then drag to place it.

To place a crosstab:

- 1 Create a crosstab using either the dialog or visual methods previously described. Depending on the location of your mouse, the No-drop cursor or the Crosstab cursor appears.
- 2 Use the mouse to position the Crosstab cursor in the header or footer where you want to place the crosstab and click the mouse.

- 3 If the header or footer isn't big enough to hold the crosstab, the area resizes to fit it. ReportSmith places the crosstab below the point where you press.

You can expand the header or footer to a full page or more to force a crosstab across multiple pages vertically.

The type of header or footer within the columnar report that you place a crosstab controls what values are used in the calculations and how often the crosstab appears in your report. If you drop the crosstab into a global region such as a report or page header or footer, it calculates values based on *all* data in the columnar report and it appears in *every* report header or footer. If you place it in a group header or footer, it calculates values based only on the data in each group and places it *only* into that group header or footer.

To reposition a crosstab:



- 1 Double-click to select crosstab or position the cursor on an inside edge of the crosstab until the selection finger cursor appears.
- 2 Drag the crosstab to a new location within a header or footer, then release the left-mouse button.

When you press and hold the mouse after selecting a crosstab, the movement cursor appears. When you see this cursor, you can move the crosstab horizontally or vertically. ReportSmith places the crosstab in the new location.

Expanding a boundary

If the crosstab doesn't fit within the header or footer of a columnar report, ReportSmith suppresses the cells that fall outside the top or bottom boundaries. To expand the boundary, click on it. Press and drag one of the sizing handles until the entire crosstab fits within the boundaries. Also, you can stretch a header or footer across pages. This is helpful if a crosstab is too large to fit on one page. If you stretch the header or footer to the next page, the crosstab splits across the pages, in the vertical direction only.

To cancel creating a crosstab

You can cancel creating a crosstab before you place it in your report. Left-click mouse button when the no-drop cursor appears, or press the *Escape* key.

Selecting crosstab items

After you select an item in a crosstab, you can format or align it like any selected item in a columnar report.

To select the entire crosstab, double-click in the crosstab.

To select specific fields in a crosstab, you can select rows, columns or individual totals in a crosstab:

- 1 Click in a field. A box surrounds the field and all other fields of the same type.
- 2 *Ctrl*+click to select additional fields types.

To select all fields in a crosstab:

- 1 Double click the crosstab.
- 2 Right-click the mouse and choose the Select All command from the pop-up Crosstab menu.

Modifying a crosstab

You can make changes to your report at any time. ReportSmith offers you many ways to modify your crosstabs. You can use the Crosstab dialog box, drag and drop rows and columns into place, directly edit crosstab cell values, and use the formatting ribbon and toolbar.

To display the Crosstab toolbox, choose Tools | Crosstab.

You can modify your crosstab at any time. After you apply them, ReportSmith regenerates the crosstab with your new changes. To modify a crosstab with the Crosstab Report dialog box:

- 1 Double-click a crosstab to select it.
- 2 Press the Modify button on the toolbox, or right-click the mouse and select Modify from the pop-up menu.
- 3 Make the changes you want in the Crosstab Report dialog box and press OK. ReportSmith applies your changes to the crosstab.

To modify a crosstab directly:

- Select some crosstab cells and right-click to see formatting and editing options.
- Reselect a cell to enter in-cell editing mode. Click outside the cell to complete editing.

Select the toolbar icons to:

- Sort row or column values in ascending or descending order
- Create or modify subtotals for rows or columns
- Format fonts, alignment or field formats

You can visually modify a crosstab without using the dialog box. Select a row or column and drag it to where you want it on the report. ReportSmith shows you where you can and cannot place the items and after you drop items, automatically regenerates the report.

To drag and drop rows and columns:

- 1 Select an item in your crosstab that you'd like to move.

	CATEGORY					Total
	Action	Cartoons	Classics	Munchies	Supplies	
Blake	27.62	4.44	0.00	0.28	0.00	32.34
Ginsberg	43.88	16.78	0.00	0.00	2.78	63.44
Keats	31.49	35.78	0.00	0.78	1.47	69.51
Pound	5.54	30.08	5.38	0.53	5.63	48.26
Rimbaud	24.00	7.00	0.00	0.00	2.00	35.47
Snyder	44.01	39.02	0.00	0.50	0.00	83.52
Total	173.41	133.90	5.38	2.19	12.67	332.55

- 2 Holding the mouse, drag and drop the selected item to another location.

	CATEGORY					Total
	Action	Cartoons	Classics	Munchies	Supplies	
Blake	27.62	4.44	0.00	0.28	0.00	32.34
Ginsberg	43.88	16.78	0.00	0.00	2.78	63.44
Keats	31.49	35.78	0.00	0.78	1.47	69.51
Pound	6.54	30.08	5.38	0.63	5.63	48.26
Rimbaud	24.88	7.80	0.00	0.00	2.80	35.47
Snyder	44.01	39.02	0.00	0.50	0.00	83.52
Total	178.41	133.90	5.38	2.19	12.67	332.55

ReportSmith replaces the row as a column.

You can further define a crosstab by choosing row, value, and column options in their respective Options dialog boxes. Highlight a value and press the Options button to view the corresponding Options dialog box, where you can specify options such as labels, headings, calculations, sort order, select an alias and choose custom values.

Rows

To specify row options, press the Row Options button. The Crosstab Row Options dialog box appears.

Crosstab Row Options

Name: ACCT_NBR
Type: Numeric
Label: Account Number

Subtotal Heading:
Calculation: <none> ↓

Sort Order:
☒ Ascending
☐ Descending
☐ None
☐ Custom

Custom Values:

OK Cancel Alias... Help

Options	Description
Name	Field name you are setting options for.
Type	Type of field, such as Numeric, Character or Date.
Label	Enter an alternate name for the field into the text box. This is optional. If you leave this field blank, by default ReportSmith uses the field name as the label.
Subtotal Heading	Enter a heading for the subtotal here. This is optional. If you leave this field blank, ReportSmith leaves the subtotal heading blank. This appears if you've decided to show subtotals in your crosstab.
Calculation	Press the down arrow to view a drop-down list of subtotal calculations for you to apply to a field.
Sort Order	Click to turn on one of the following options and ReportSmith automatically regenerates and sorts the crosstab for you: Ascending (A to Z, 1 to 10), Descending (Z to A, 10 to 1), None, or Custom. If you choose Custom, type the sort order you want into the Custom Values box.
Custom Values	Enter custom values. Separate values with commas and use single quotes around each value.

Values

Specify value options in the Crosstab Value Options dialog box. To view the dialog box, highlight the value you want to customize and press the Value Options button.

Crosstab Value Options

Name: SHARES
Type: Numeric
Label: SHARES

Summary Heading: Total
Calculation: Sum

Shown Value

☒ Values
☐ Percent of Row
☐ Percent of Column
☐ Percent of Total

OK Cancel Help Edit

Option	Description
Name	The name of the value field for which you're setting options.
Type	The field data type, such as numeric, character or string.
Label	Enter an alternate label into this box for your value label. This is optional. If you leave this field blank, ReportSmith uses the field name from within the table as the default label.
Summary Heading	Enter a label for the summary heading. This is optional.
Calculation	Press the down arrow to view a drop-down list of calculations which you can perform on this field.

Shown Value Option	Description
Value	The normal calculation.
Percent of Row	Percent of total for rows.
Percent of Column	Percent of total for columns.
Percent of Total	Percent of all values in the crosstab.

Columns

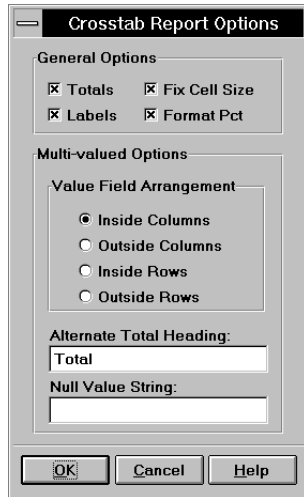
To specify column options, press the Column Options button. The Crosstab Column Options dialog box appears.

These options appear for date fields only.

Option	Description
Name	The name of the field for which you're setting options.
Type	The field type, such as numeric, character, or string.
Label	Enter an alternate label into this box for your column label. This is optional. If you leave this field blank, ReportSmith uses the field name from within the table as the default column label.
Subtotal Heading	Enter a label for the subtotal heading. This is optional.
Calculation	Press the down arrow to view a list of subtotal calculations you can perform on this column.
Sort Order	Choose Ascending, Descending, None, or Custom for the order of values in the column. If you choose Custom, enter values by which you'd like to sort into the Custom Values window.
Date Options	Select the dates of the report you'd like to see from the top list box. Choose how you want the date grouped from the Group By list box: Quarterly, Same Value, Daily, Monthly, Weekly, or Annually. Choose a format for your date field from the Format list, such as "QTR 1" or "FIRST QUARTER." Choose the first month you want to display data for on your report from the First Month of Year List. Choose Display Range to display the full range of data for this crosstab.

Crosstab report options

Press the left-most Options button to set general crosstab options, value field arrangements, and to create an alternate total heading. The Crosstab Report Options dialog box appears.



Option	Description
Totals	Display totals in the crosstabs.
Labels	Display labels for fields, columns and values.
Fix Cell Size	Use the same cell size for every field value.
Format Pct	Use a % format for percentage values. Overrides any formatting settings in the associated crosstab style.
Inside Columns	Place multiple-column values within crosstab body.
Outside Columns	Place multiple-column values all together outside of crosstab body.
Inside Rows	Place multiple-row values within crosstab body.
Outside Rows	Place multiple-row values all together outside of crosstab body
Alternate Total Heading	Lets you specify a composite total heading for multi-valued crosstabs since the totals can show different calculations.

Resizing rows and columns

When you select a crosstab, you can use the sizing handles that appear to resize individual columns or rows in your crosstab. Alternatively, choose a sizing handle to resize all rows or columns uniformly.

You can easily hide a column in your crosstab by dragging a column sizing handle left, reducing the size of the column until it disappears from view. You can enlarge one or

more columns or rows with these sizing handles also. The crosstab sizing handles are described below.

Resize columns

		Quarter 1	Quarter 2	Quarter 3	Quarter 4	Total
Action**	Blake	No Sale	No Sale	No Sale	\$51.00	\$51.00
	Ginsberg	\$29.00	\$1.00	\$25.00	\$24.00	\$79.00
	Keats	\$38.00	\$21.00	No Sale	\$3.00	\$62.00
	Pound	No Sale	No Sale	No Sale	\$11.00	\$11.00
	Rimbaud	\$9.00	No Sale	No Sale	\$33.00	\$45.00
Cartoons	Snyder	No Sale	No Sale	\$0.00	\$74.00	\$74.00
	Blake	No Sale	No Sale	No Sale	\$9.00	\$9.00
	Ginsberg	\$4.00	\$7.00	No Sale	\$74.00	\$84.00
	Keats	\$7.00	\$7.00	No Sale	\$44.00	\$68.00
	Pound	No Sale	No Sale	\$8.00	\$43.00	\$56.00
	Rimbaud	No Sale	No Sale	No Sale	\$74.00	\$74.00
	Snyder	No Sale	No Sale	\$7.00	\$71.00	\$75.00
Total		\$87.00	\$55.00	\$40.00	\$199.00	\$581.00

Resize rows

Uniform column sizing

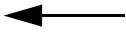
Uniform row sizing

Individual rows

Use the row sizing handles on the left side of the crosstab to resize an individual row. For example, to enlarge the size of two rows containing “Rimbaud,” shown above, grab the row-sizing handle just above the cell and pull upward. Do the same for the second row.

Individual columns

Use the column-sizing handles at the top of the crosstab to resize an individual column. For example, to remove the column “Quarter 4,” pull the sizing handle above the column left until “Quarter 4” disappears from view.



		Quarter 1	Quarter 2	Quarter 3	Quarter 4	Total
Action™	Blake	No Sale	No Sale	No Sale	\$51.00	\$51.00
	Ginsberg	\$25.00	\$1.00	\$25.00	\$24.00	\$79.00
	Keats	\$36.00	\$21.00	No Sale	\$3.00	\$60.00
	Pound	No Sale	No Sale	No Sale	\$11.00	\$11.00
	Rimbaud	\$5.00	No Sale	No Sale	\$33.00	\$45.00
	Snyder	No Sale	No Sale	\$3.00	\$74.00	\$77.00
Cartoons	Blake	No Sale	No Sale	No Sale	\$9.00	\$9.00
	Ginsberg	\$4.00	\$16.00	No Sale	\$14.00	\$34.00
	Keats	\$7.00	\$17.00	No Sale	\$44.00	\$68.00
	Pound	No Sale	No Sale	\$0.00	\$43.00	\$56.00
	Rimbaud	No Sale	No Sale	No Sale	\$14.00	\$14.00
	Snyder	No Sale	No Sale	\$4.00	\$71.00	\$75.00
Total		\$87.00	\$55.00	\$40.00	\$390.00	\$581.00

Below, the column containing “Quarter 4” no longer appears. However, it still exists in your crosstab. To make it reappear, select Edit | Undo, or pull the sizing handle right to increase the column size.

		Quarter 1	Quarter 2	Quarter 3	Total
Action™	Blake	No Sale	No Sale	No Sale	\$51.00
	Ginsberg	\$29.00	\$1.00	\$25.00	\$79.00
	Keats	\$36.00	\$21.00	No Sale	\$62.00
	Pound	No Sale	No Sale	No Sale	\$11.00
	Rimbaud	\$5.00	No Sale	No Sale	\$45.00
	Snyder	No Sale	No Sale	\$3.00	\$77.00
Cartoons	Blake	No Sale	No Sale	No Sale	\$9.00
	Ginsberg	\$4.00	\$16.00	No Sale	\$34.00
	Keats	\$7.00	\$17.00	No Sale	\$68.00
	Pound	No Sale	No Sale	\$3.00	\$56.00
	Rimbaud	No Sale	No Sale	No Sale	\$14.00
	Snyder	No Sale	No Sale	\$4.00	\$75.00
Total		\$87.00	\$55.00	\$40.00	\$581.00

Uniform row and column sizing

Use the uniform column- or row-sizing handles located on the right and bottom of the crosstab. This ensures consist sizing in your report.

- To resize rows uniformly, drag the row-sizing handle up or down. All of the rows in your report resize along with it.
- To resize columns uniformly, drag the column-sizing handle left or right. All of the columns resize along with it.



As you drag the sizing handle, you can use the size of the last column or row as a gauge for sizing the other columns or rows in your report.

Manipulating data

ReportSmith offers you numerous ways to expand your crosstabs. This section describes

- Grouping
- Sorting
- Inverting rows and columns
- Calculations (including macro-derived fields)
- Aliases

Grouping

ReportSmith groups all values by same value except for dates. Dates are automatically grouped by quarter. You can use the options dialog box to customize grouping for dates. For example, you can specify a date range, choose to display dates by month, days or minutes, or you can define the first month of your fiscal year. When grouping dates, ReportSmith offers you numerous options.

To specify dates in a crosstab:

- In the Crosstab Report dialog box, select the DATE field you want to modify and press the Options button that corresponds to it. An options box appears with settings for the DATE field.

Option	Description
Name	The name of the field for which you're setting options. DATE should appear here.
Type	The field type, DATE, Date/Time, should appear here.
Label	Enter an alternate label into this box for your DATE label. This is optional. If you leave this field blank, ReportSmith uses the field name from within the table as the default column label.
Subtotal Heading	Enter a label you'd like to appear as the subtotal heading.
Calculation	Press the down arrow to view a drop-down list of subtotal calculations which you can perform on the DATE field.

Option	Description
Sort Order	Click to turn on one of the following options and ReportSmith automatically regenerates and sorts by DATE for you: Ascending, Descending, None, or Custom. If you choose Custom, type the sort order you want into the Custom Values box.
Custom Values	Enter custom values here. Separate values with commas and use single quotes around each value.
Date Options	This lets you select a range of dates that you would like to include in your crosstab. By default, it includes all dates, but you can limit the data to a specific date range here.
Group By	Press the down arrow to view a drop-down list of options by which you can group the DATE field.

- Group formats translate into field labels in your crosstab. You have the following choices:

Group	Group Formats
Daily	Sunday or Sun
Monthly	January or Jan
Weekly	Week #
Annually	Year #
Quarterly	Quarter 1, Qtr 1, First Quarter, First Qtr, 1st Quarter, 1st Qtr
Hourly	Hour #
Minute	Minute #
Second	Second #

First Month of Year

If you choose Annually or Quarterly as the Group By option, you can select the first month of your fiscal year. Press the down arrow to see a drop-down list and choose a month.

Format

If you select Daily, Monthly, or Quarterly in the Group By option, you can select a label format and other options. Press the down arrow in the Format box to display a list of choices. After you choose an option in the Group By box, the grouping options appear in the dialog box.

The default format for dates is Quarterly with a “Qtr 1” format.

Deleting rows, columns or values

You can delete a row, column, or value from your report by selecting the corresponding Delete button for the list box or by dragging the row, column, or value back toward the Name box. You can also change the hierarchical order within a list box. Press to select a row, column, or value and drag it to another position in the list box.

Sorting

You can reorganize the order of the values within a selected row or column in a crosstab. You can do this with the Sort options in the Crosstab Row Options or Crosstab Column Options dialog boxes.

You can also use the Custom Sort option to exclude values you don't want to include in your crosstab.

To sort fields in a crosstab:

1 Do one of the following:

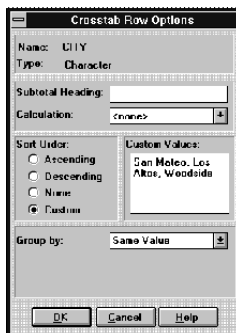
- Select some cells in a crosstab and press a sort button on the toolbar.
- Select the row or column in the Crosstab Report dialog box and press the corresponding Options button.

If you select a row, the Crosstab Row Options dialog box appears. If you select a column, the Crosstab Column Options dialog box appears. In our example, we'll sort by the CITY row.

2 Press an option in the Sort Order box. The table below defines your sort choices:

Ascending	Sorts fields in ascending order. A to Z, 1 to 10. This is the default.
Descending	Sorts fields in descending order. Z to A, 10 to 1.
None	No sort order is specified. This is the default
Custom	Allows you to type a custom sort in the Custom Values box. This lets you arrange the order of values within a selected row or column. You can also use this option to exclude fields you don't want in your crosstab.

If you select the Custom option, type a sort list for the fields in the selected row or column in the Custom Values box. Each value should be separated by a comma. For instance, if you want to reorder the CITY fields in the example, you could type San Mateo, Los Altos, Woodside in the Custom Values box.



If you want to exclude a field in your crosstab, don't include it in your sort. For example, if you don't want to include Woodside in the example crosstab, you would type **Los Altos, San Mateo** in the Custom Values box.

- 1 Press OK. Press OK again to exit the Crosstab Report dialog box. ReportSmith generates the crosstab with the sorted fields.

		DATE				
		1st Qtr	2nd Qtr	3rd Qtr	4th Qtr	Total
EMP_LNAME	CITY	TOTAL	TOTAL	TOTAL	TOTAL	TOTAL
Blake	Los Altos	\$160	\$471	\$505	\$587	\$1,723
	San Mateo	\$0	\$0	\$0	\$0	\$1,723
Keats	Los Altos	\$0	\$0	\$0	\$0	\$1,723
	San Mateo	\$170	\$163	\$335	\$218	\$886
Total		\$330	\$634	\$840	\$805	\$2,609

Inverting rows and columns

Double-click a crosstab and press the Invert button on the Crosstab toolbox, or right-click the mouse and choose Invert from the pop-up Crosstab menu.

ReportSmith swaps rows with columns.

	EMP_LNAME,CITY		
	Blake	Total	
	Los Altos	San Mateo	
DATE	TOTAL	TOTAL	TOTAL
1st Qtr	\$160	\$0	\$160
2nd Qtr	\$471	\$0	\$471
3rd Qtr	\$505	\$0	\$505
4th Qtr	\$587	\$0	\$587
Total	\$1,723	\$1,723	\$1,723

To undo the Invert command, choose Edit | Undo Command, or select the Invert command once again.

Calculating fields

ReportSmith enables you to calculate values in a crosstab based on other crosstab values using ReportSmith's macro language. You can create SQL-derived fields in the Derived Fields dialog box, and they appear in the crosstab just like regular data fields. However, to include a macro-derived field in your crosstab, you must define it within the Crosstab Report dialog box, using the Calculated Column option.

You can only calculate values in a crosstab, not rows or columns.

To create a calculated field:

- 1 Select the crosstab, right-click the mouse and choose Modify, or select the entire report and press the Modify button on the Crosstab toolbox.

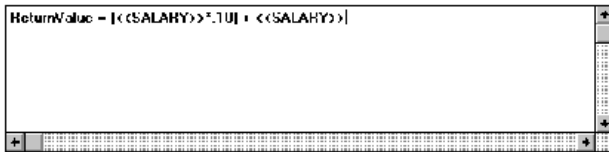
The Crosstab Report dialog box appears displaying the rows, values and columns you have selected to include in it.

- 2 Under Type, select Calculated Fields. A symbol appears in the Name box, <New>.
- 3 Drag the <New> symbol from the Name box and drop it into the Values list box.

The Calculated Field dialog box appears, where you will tell ReportSmith what calculation to perform on the field. The current value fields appear in the left-most list box.

- 4 In the Name text box, enter a name for your calculated field. In the example, we've named our calculated field "NewSalary."
- 5 You can create the calculated field formula by typing it directly into the formula box, or you can drag and drop items from the list boxes into the formula.

In the example below, we've created a formula which uses the SALARY field and calculates a 10% raise for each employee.



- 6 After you've created the formula for your calculated field, press the Test button. ReportSmith tests your formula and, if you entered it correctly, returns a success message. If you entered it incorrectly, an error message appears. You need to correct the formula before continuing.
- 7 Press OK to exit the Test message box. Press OK to exit the Calculated Field dialog box. Your calculated field appears in the Values box. We named our sample calculated field NewSalary. It appears in the Values list box beneath SALARY.
- 8 Press OK.

Creating a field alias

You can use aliases for fields in a crosstab. For example, suppose you've created a report using the EMPLOYEE table, which identifies your employees by number. Instead of using the EMP_ID number you can use aliases to display employee names on your report quickly without linking to another table.

ReportSmith offers you two methods for creating and using aliases:

- Click on a selected cell to enter Text mode and type in the alias
- Press the Alias button in an Options dialog box

As an example, we'll change the first EMPLOYEE_ID field, "E1-23," to "J. Smith."

EMPLOYEE_ID	DATE				Total
	Quarter 1	Quarter 2	Quarter 3	Quarter 4	
QUANTITY	QUANTITY	QUANTITY	QUANTITY	QUANTITY	QUANTITY
E1-23	9.00	0.00	0.00	55.00	64.00
E2-12	45.00	17.00	25.00	30.00	125.00
E2-30	39.00	0.00	20.00	62.00	121.00
E2-47	19.00	0.00	7.00	145.00	171.00
E3-25	35.00	38.00	0.00	47.00	130.00
E4-62	8.00	0.00	0.00	60.00	68.00
Total	215.00	55.00	52.00	407.00	729.00

To create aliases in Text mode:

- 1 Open a crosstab. Click on the field you want to edit, then click on it once again to activate Text mode.
- 2 Type the alias over the field.

EMPLOYEE_ID	DATE				Total
	Quarter 1	Quarter 2	Quarter 3	Quarter 4	
QUANTITY	QUANTITY	QUANTITY	QUANTITY	QUANTITY	QUANTITY
J. Smith	9.00	0.00	0.00	55.00	64.00
E2-12	45.00	17.00	25.00	30.00	125.00
E2-30	39.00	0.00	20.00	62.00	121.00
E2-47	19.00	0.00	7.00	145.00	171.00
E3-25	35.00	38.00	0.00	47.00	130.00
E4-62	8.00	0.00	0.00	60.00	68.00
Total	215.00	55.00	52.00	407.00	729.00

ReportSmith replaces each occurrence of that field in your report with its alias. Your database remains unaffected; only the report changes. Now every time ReportSmith finds E1-23 in this crosstab, it uses J. Smith instead.

Note You can also create a single alias that includes multiple employees. Employees assigned the same alias collapse into the same group.

To create aliases using the Options dialog box:

- 1 Open a crosstab. Select it and choose Modify from the pop-up menu, or double-click to select the entire crosstab and press the Modify button on the Crosstab toolbox.

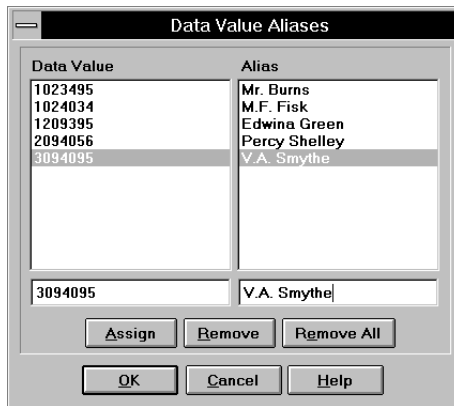
The Crosstab Report dialog box appears.

- 2 Select the field in the Rows, Columns or Values list boxes that you want to create aliases for and press the Options button beneath it.

In our example we'll create an alias for the Account Number field. We'll highlight the field and press the rows Options button. The Crosstab Row Options dialog box appears with the selected field in the Name box.

- 3 Press the Alias button. The Data Value Aliases dialog box appears. If you have created any aliases, they appear beside the data value which they represent.
- 4 Select a data value in the left list box. Click in the text box under the Alias list and type an alias for it. Press the Assign button. ReportSmith places the alias in the Alias list box beside its respective value. Repeat these steps for each alias you want to create.

In this example, we created aliases for each Account Number field:



- 5 When you've finished creating aliases, press OK. ReportSmith replaces each data value you assigned aliases to with its alias.

To correct a mistake in an alias:

- 1 Edit an alias by highlighting it in the list.
- 2 Retype a new alias for it in the Text box and press Assign.
- 3 To remove one or all aliases from the list box, press one of the Remove buttons.

Formatting techniques

You can format selected fields in a crosstab. You can change the font and point size, add borders, patterns, and shadows, align characters, and so on. ReportSmith lets you save your formatting changes in styles, or you can apply system styles that come with ReportSmith.

This section describes how to use crosstab styles and how to resize items in a crosstab.

Working with crosstab styles

A *style* is a set of cell formatting information that you've given a name. Suppose you've formatted a crosstab to look the way you want. You've changed the font and point size, italicized the title, and placed colored borders around selected fields to emphasize them. Now you want to apply this same style to another crosstab. You could create a new crosstab and then reapply each formatting change to it, but that's extra work. Furthermore, if you decide to change the formatting you'd have to repeat the formatting steps for each crosstab.

Instead, you can save the formatting as a style by selecting the crosstab you've formatted and assigning a name to it. After you save a style, it appears in the Custom Styles list. To apply it to a crosstab, select it. You can apply numerous formats in one easy step.

Use styles to

- Ensure consistent formatting throughout your crosstabs.
- Save time by applying formatting changes quickly.
- Make design changes easy to incorporate.

With a crosstab selected, press the Styles button on the Crosstab toolbox or select Styles from the pop-up Crosstab. Pressing this button displays the Crosstab Styles dialog box, where you can create new styles or apply existing styles.

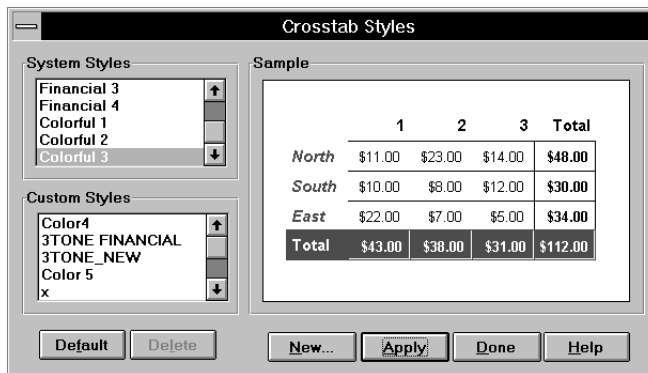
A list of System Styles that ReportSmith has created for you appears in the System Styles list. Styles you create appear in the Custom Styles list.

Applying a style

You can apply a style using the ribbon or the Crosstab Styles dialog box.

To apply a style with the Crosstab Styles dialog box:

- 1 Select a crosstab, right-click the mouse, and choose Styles from the menu. The Crosstab Styles dialog box appears.



- 2 Select a style in the System Styles list box or in the Custom Styles list box.

3 Press Apply.

To apply a style with the ribbon:

- 1 Select a crosstab. Choose Tools | Crosstab.
- 2 Press the Styles button, or right-click the mouse to display the pop-up Crosstab menu and choose Styles. The Crosstab Styles dialog box appears.
- 3 Choose a style from either the System Styles or Custom Styles list box and press Apply.

ReportSmith displays the style you select in the Sample box so you can see the formatting before applying it. ReportSmith applies the style to the selected crosstab.

- 4 Press Done.

Creating a style

You can create a style based on a formatted crosstab. Styles that you create appear in the Custom Styles list box. When you select a style, ReportSmith displays it in the Sample box, so you can preview it. If you've already formatted a crosstab, most of the work to create a style is behind you. Now all you need to do is assign a name to the style. A style is created and is available for use with other crosstabs.

Create custom styles using, for example, your company colors. When you delete a custom style it affects crosstabs which have been assigned that style. ReportSmith alerts you with a message reminding you that removing the style affects any crosstabs with that style.

- 1 Select a crosstab whose formatting you want to save.
- 2 Choose Tools | Crosstab to see the Crosstab toolbox appears.

You can drag the toolbox out of your way by dragging the title bar and moving it to where you want it.

- 3 Press the Style button, or right-click to display the pop-up Crosstab menu and choose Styles. The Crosstab Styles dialog box appears.
- 4 Press the New button. Enter a name for your style into the Name text box. Style names can contain up to 255 characters and can include any combination of characters and spaces. They are case sensitive and must be unique.
- 5 Press OK and press Apply to exit the Crosstab Styles dialog box. Now, each time you open the Crosstab Styles dialog box your new style appears in the Custom Styles list.

After you have defined a style, it appears in the Report Styles dialog box and you can apply it to any number of crosstabs. Applying a style to a crosstab gives it the same formatting as the crosstab you used as the model for that style.

Extracting a style

Suppose you've formatted a column, row, or value within your crosstab and you'd like to apply that style to other areas within your report. You can do this with the Style

Extractor button on the toolbar. The Style Extractor extracts styles you select and applies them to columns, rows or values that you click on.

To apply formatting using the Style Extractor:

- 1 Select a row, column, or value whose formatting you want to copy.
- 2 Press the Style Extractor button on the toolbar. The Style Extractor cursor appears.
- 3 Move the cursor over a value you want to apply that style to and click the mouse. ReportSmith applies the style to all similar rows, columns, or values.

The following illustration shows an example of using the Style Extractor to apply three formatting changes (bold, italic, and dollar sign) to each Total row value.

We want to apply the formatting of the row Grand Total to all of the total values in the row. To do this, we selected the Grand Total field.

CATEGORY						
	Action	Cartoons	Classics	Munchies	Supplies	Total
Blake	27.62	4.44	0.00	0.28	0.00	32.34
Ginsberg	43.88	16.78	0.00	0.00	2.78	63.44
Keats	31.49	35.78	0.00	0.78	1.47	69.51
Pound	6.54	30.08	5.38	0.63	5.63	48.26
Rimbaud	24.88	7.80	0.00	0.00	2.80	35.47
Snyder	44.01	39.02	0.00	0.50	0.00	83.52
Total	178.41	133.90	5.38	2.19	12.67	\$332.55

ReportSmith applies the formatting changes from the Grand Total field to each Total row value:

CATEGORY						
	Action	Cartoons	Classics	Munchies	Supplies	Total
Blake	27.62	4.44	0.00	0.28	0.00	32.34
Ginsberg	43.88	16.78	0.00	0.00	2.78	63.44
Keats	31.49	35.78	0.00	0.78	1.47	69.51
Pound	6.54	30.08	5.38	0.63	5.63	48.26
Rimbaud	24.88	7.80	0.00	0.00	2.80	35.47
Snyder	44.01	39.02	0.00	0.50	0.00	83.52
Total	\$178.41	\$133.90	\$5.38	\$2.19	\$12.67	\$332.55

When you no longer need a style, you can delete it. When this happens, you replace any crosstab formatted with this style to the basic, non-formatted style. You cannot delete a system style.

To delete a custom style:

- 1 Select the crosstab and choose Tools | Crosstab.

- 2 Press the Styles button on the Crosstab toolbox.
- 3 Select the style you want to delete from the System Styles or Custom Styles list and press the Delete button. ReportSmith displays a message asking you to confirm that you want to delete the style and reminding you that removing the style affects all the reports with this style.
- 4 Press Yes to delete the style, or No to cancel the deletion. Press the Close button.

Macros

Macros let you enhance your productivity and add complex features to your reports. A *macro* is a list of actions you want ReportSmith to carry out for you. You can create a macro or use one of the macros ReportSmith provides for you to do any of the following tasks (and many more):

- Automatically load and print reports
- Customize ReportSmith by creating a custom dialog box, hiding or disabling menu items, and creating new menu items and DDE topics
- Create derived fields
- Create customized prompts such as prompting a user for a password
- Schedule report timing
- Perform complex calculations

This chapter defines macros and shows you how to create and use them to simplify many ReportSmith tasks. Finally, it describes some advanced macro concepts.

See Appendix B, “Macro reference,” for more information about macros.

Introduction to macros

In this introductory section, you’ll learn what macros are and how to use them. You’ll also learn the difference between global and local macros, and how they are used. This section also provides basic information about using macros with other Windows applications and about the ReportBasic programming language.

What is ReportBasic?

ReportSmith has licensed the Softbridge Basic Language (SBL) from Mystic River Software, Inc. in order to provide ReportSmith users with a complete, high-level

programming language. ReportSmith has also added commands to SBL that were designed with reporting in mind. This combined command set is called ReportBasic, a complete programming language designed specifically for report creation.

Because it is a complete programming language, ReportBasic can be used for a variety of tasks. The following table provides a list of its more common uses.

Use	Description
Calculating derived fields	With ReportBasic, you can create derived fields that are not stored in the database, but are a result of a calculation using the values stored in your database fields.
Conditional formatting and conditional text values	ReportBasic lets you highlight a value if it meets a certain criteria in a report. For example, you might want to display all sales transactions in a report that are greater than a stated quota to print in a bolder, larger font than those that fall below the quota. ReportBasic can also be used to warn a report reader that invalid data exists in a database. Suppose a mailing list database contains a number of null values in the state field. ReportBasic can print the value "NO STATE CODE" in place of the null state code value.
Customized user interface	You can use ReportBasic to create custom user interfaces such as dialog boxes that contain user-entry fields, command buttons, and check boxes. A report user can build a complex SQL query with a few mouse clicks, with the help of the report designer who is familiar with ReportBasic. ReportBasic can also be used to add, remove, or disable ReportSmith menus, menu items and toolbar icons. A report designer may want to do this to prevent a user from modifying a report or to add specific menu items that run specific macros. You can create your own DDE topics and items. Macros can be assigned to run in response to a change in one of the items or a request for data from another application.
Automatic report processing	ReportBasic can be used to load a number of reports, send them to a printer and close those reports when they have printed. A user can link such a macro to a specific event. A user could cause such a macro to execute in response to a user keystroke, or other ReportSmith event. Or, a macro might be programmed to run reports at midnight, when network traffic is low.
Creating user-defined functions	Advanced users can create ReportBasic functions that can be called by other ReportBasic scripts or a ReportSmith event. Users can also call *.DLL functions, created in other development environments, that can be used within ReportBasic.
Dynamic Data Exchange (DDE)	ReportSmith, along with ReportBasic, can serve as either a DDE server or client. A developer may want to build a number of reports with ReportSmith and build DDE commands within the application that calls ReportSmith or ReportSmith Runtime to print those reports when the user clicks a button or selects a menu option. This can easily be done from within applications developed with Delphi. You can perform DDE executes, pokes and requests and create your own DDE topics and items.

Uses for macros

Macros are programs that you build using the ReportBasic programming language. (See "What is ReportBasic?" on page 159.) You do this by creating a macro, then linking it to an event on which the macro can run. For example, you can display a dialog box that prompts for options whenever a user opens a report. You can link a macro to a specific event, such as a keystroke or the opening of a report.

Scoping macros

You can create two kinds of macros: a global macro or a report macro. A *global macro* belongs to the ReportSmith application and appears in the active list each time you run ReportSmith. A *report macro* belongs to a given report and becomes a part of that report.

When you save the report, ReportSmith saves any associated report macros in the *.RPT* file.

Global macros

You can create global macros to customize ReportSmith or to automate global tasks. For example, suppose you want to combine loading, printing, and closing a report into one step. This same macro can also display a dialog box that prompts the user for the report names. When you run the macro, it opens, prints, and closes each report the user specifies in the dialog box.

You can create a global macro that only lets certain people access a given report by prompting for a password, or one that hides the toolbar or executes a specific menu command, such as File | Open.

Report macros

A *report macro* lets you automate tasks that are *specific to a certain report*. For example, create a report macro that only lets a certain user access a confidential report by prompting for a password, or by verifying the value of a DOS variable. For example, you might have this line in your AUTOEXEC.BAT:

```
SET USER=John Doe
```

In this case, the macro opens the report only if the environment variable is set to John Doe.

```
' This code will abort the open event if the user is not John Doe
' Get the user name
USER$=ENVIRON$("USER")
  IF USER$ <> "John Doe" THEN
    ' Abort the report's loading
    RESUMEEVENT 0
  END IF
```

You can create a report macro that behaves dynamically according to changes in the information it receives. For example, suppose you have a report that is unusually large, and you don't want users opening it when network traffic is high (between 4:00 and 5:00 every week day). You can prevent a user from opening the report during peak time, and display a message explaining why. In this case, the macro behaves differently according to the time of day.

You can create a report macro that steps through the fields in a report column and produces a summary field. For example, suppose your Sales Department has a commission structure with different multipliers for different quotas. You could write a macro that calculates a sales person's commission based on the commission structure, and place that value into your report.

Using macros with other applications

Report macros let you extend to other Windows applications. For example, if you have a report with a column listing part numbers and an Excel spreadsheet with a table listing these part numbers and their corresponding part names, you can write a macro that changes the part number to the part name in your report. You'd simply link the macro

to the Display event and use the FieldText command to change the text of each record. You can then do a DDE request to your Excel table to locate the corresponding part name for each part number.

You can also add C functions from a Windows Dynamic Link Library (DLL) into a report macro. If you have a Windows DLL containing functions that perform complicated financial computations, you can create a report macro that uses those functions to calculate the data in your report.

Creating and working with macros

This section shows you how to create a macro. It takes you step-by-step through the creation of a simple macro and shows you how to link a macro to objects and events. First, you'll create a macro that displays a message box. Then you'll link the macro to an object and an event.

Creating a simple macro

To create a macro:

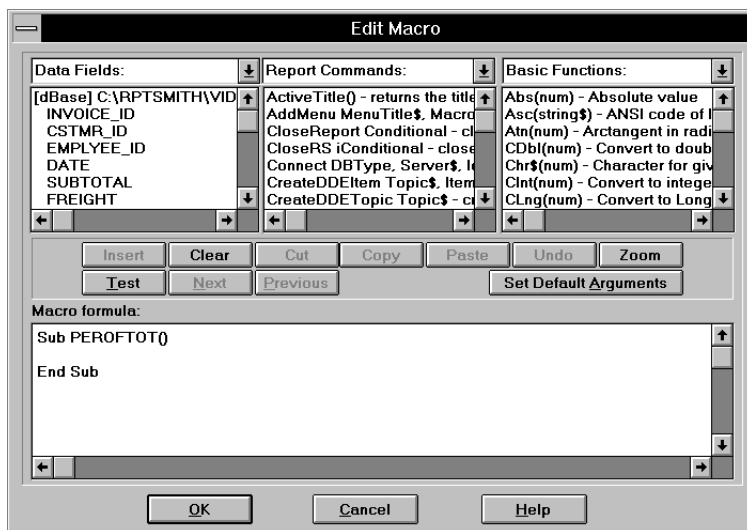
- 1 Select Tools | Macro to see the Macro Commands dialog box.

Notice the Global Macros radio button at the bottom left-hand corner is active. This indicates that you're creating a global macro. (Assuming you don't have any reports open, the Report Macros button is disabled.)

- 2 In the Macro Name text box, enter a name for your macro into the Macro Name text box. For our sample, we entered `TEST`. This enables the New button.

Note Your macro name cannot be the same as a Basic reserved word, or errors result.

- 1 Choose New. The Edit Macro dialog box appears.



- Note** ReportSmith commands and the Basic commands are listed and described in “The Macro Command Reference” in online Help.
- You create the macro by inserting the commands you want in the Macro Formula text box. Since you’re creating a macro that displays a message box, you’ll use the MsgBox command.
- 2 In the Basic Functions list box, scroll through the list to display the MsgBox function, and then drag it to the beginning of the blank line in the formula (between `Sub TEST()` and `End Sub`), then release the mouse button to insert the function into the routine. (You can also type the command directly in the formula.)
 - 3 In this exercise, you can also use the MsgBox command as a statement, so you must edit the command by removing the parentheses. Delete the parentheses and commas after MsgBox.
 - 4 Enter the following into the Macro Code text box: `'This is a sample message.'`
 - 5 Choose Test. If you entered the routine correctly, ReportSmith displays a success message. If your formula is incorrect, ReportSmith informs you of errors and lets you fix the routine.
 - 6 When you receive a success message, press OK to return to the Macro Commands dialog box. Now you can run the macro and see the effect.
 - 7 In the Macro Commands dialog box, choose Run. The message you created appears.
 - 8 Press OK in the message box. Press OK again to close the Macro Commands dialog box. Your macro is complete.

Linking a macro to an event

This section shows you how you can link a macro to an event.

To link a macro to an event:

- 1 Open the Macro Commands dialog box and select your macro. For our example, select TEST.
- 2 Press the Links button. The Macro Links dialog box appears.

If you press the down arrow in the Object Type list box, you’ll notice that it displays the Application object only. This is because a global macro (which is the type you just created) can only be linked to an application object.
- 3 Open the Events list box. This displays all of the events to which you can link your macro. Let’s link it to the KeyStroke event.
- 4 Select KeyStroke.

The KeyStroke event has several options for keys. You can choose to use any key on your keyboard. When an event has several options, ReportSmith displays them in the scroll box on the right.

Now link the test macro to a specific key.

- 5 Scroll through the scroll box until you see `[Ctrl+A]` and select it. Press the Link button. ReportSmith links the TEST macro to the `[Ctrl+A]` keystroke sequence. Notice that the Link Number field (which originally displayed “No Links”) now displays “1 of 1.” This indicates you have linked the macro to one event.

If you like, you can link the same macro to more keystrokes or you can link it to another event.

- 6 Select Application Startup from the Event list box, and choose Link. The Link Number field displays “2 of 2,” indicating that you have linked two events to your macro.

Also notice that the scroll box on the right displays “No Items.” Unlike the KeyStroke event, the Application Startup event doesn’t have items tied to it.

- 7 To scroll through the links of a given macro, press the Previous Link and Next Link buttons.
- 8 Press OK to return to the Macro Commands dialog box.

Saving a macro to a .MAC file

ReportSmith lets you save a macro in a .MAC file. If you save your macros with this extension you can share the macro with other ReportSmith users. You can also put it on a disk or send it over a network to another workstation. You may even want to copy it and use it in another report on your own system. You might want to save macros in this format for archiving.

Whether you save the macro in a .MAC file or not, ReportSmith saves it within the macro facility, and you can access it by selecting Tools | Macro. For report macros, ReportSmith saves the links with the report itself in the .RPT file. For global macros, ReportSmith saves the links with the ReportSmith application.

When you save a macro in a .MAC file, you save only the macro code itself. Links to events and objects are not stored in the .MAC file since they might not be appropriate for other users.

Assuming you want to share your TEST macro (the one you created in the previous exercises) with other users or reuse it elsewhere, let’s save it in a .MAC file.

To save a macro:

- 1 In the Macro Commands dialog box, choose a macro from the list of active macros. For our example, select the TEST macro that you created in the previous exercise.
- 2 Press the Save As button. The Save Macro dialog box appears. Use the Drives list box and the Directories scroll box to specify the path where you want to store the macro.
- 3 In the File Name field, enter a name for your macro file.

A macro file name can be up to eight characters and it doesn’t have to be the same name as the macro itself. Characters, numbers, and underscores are allowed but the first character must be a number or letter.

- 4 Press OK to save the macro under the path and file name you specified and to return to the Macro Commands dialog box.
- 5 Press OK to close the Macro Commands dialog box.

Loading a macro

After you save a macro to a .MAC file you might want to edit its routine or bring it into another report. You can do this by loading the file into the macro facility.

To load a macro:

- 1 Select Tools | Macro to display the Macro Commands dialog box.

To successfully load a macro, a macro with the same name in the first line of its routine must *not* appear in the Active Macros list box. If you attempt to do this, ReportSmith displays the following error message: "Name not unique. Remove test macro from active list first."

To remove a macro with the same name from the list box, highlight it and press the Remove button.

- 2 Press the Load button. The Load Macro dialog box appears.
- 3 From the scroll box on the left, double-click the macro file you want. The macro you select becomes the active macro in the Macro Commands dialog box.
- 4 Choose Edit to display the Edit Macro dialog box. Make a valid change in the Macro Formula text box. Use the Test button to test for errors.
- 5 Press OK and press Save to update your macro file. Press OK to exit the dialog box.

Advanced macro concepts

This section discusses advanced concepts for ReportSmith's macro facility, starting with macro commands.

There are two types of macro commands: Functions and Statements.

Functions return values and must have their arguments surrounded by quotes and use required parentheses around the list of arguments. *Statements* have quotes around their arguments as well, but no parentheses. Statements do not return a value. Some commands can be used as either functions or statements.

You can use commands as either a function that requires a return value, or a statement, which does not require a return value. You can use the DataSet control object for various tasks such as making a connection or setting your selection criteria. We'll show you how to create a DataSet control object and we'll define methods and properties that you can use with them.

This section describes using macro commands as functions and statements, using arguments, calling other macros, using macros with DDE, and using DataSet Control objects.

Commands as functions or statements

You can use macro commands as either functions or statements. Although you can use many of them as both, some can be used only as a function or a statement. A *function* must return a value and its argument must be within parentheses; a *statement* doesn't use a return value and doesn't use parentheses.

There may be times when you'll want to build a macro that does require a return statement. For example, suppose that every time a user exits ReportSmith, you want a message to appear asking the user if they really want to quit. Based on the user's response to the message, the macro can either close ReportSmith or cancel the exit. In this case, you would use the MsgBox command as a function (rather than a statement) as the following routine shows.

Note The following piece of code is an example. An apostrophe before a line (') indicates a comment.

```
Sub ConfirmExit()  
  ' This is the code to tell MsgBox the message and buttons you want.  
  x = MsgBox ("Do you really want to Exit ReportSmith",4)  
  ' When used as a function MsgBox returns 7 if the user presses  
  ' the NO button.  
  If x = 7 then  
    ' This command will tell ReportSmith not to close if this macro is linked to the close  
    event.  
    ResumeEvent(0)  
  End if  
End Sub
```

Using arguments

ReportSmith macros can accept arguments. An *argument* lets you pass information from one macro to another and back again. When you insert an argument into a macro, it can call another macro.

To insert an argument into a macro:

- 1 Select Tools | Macro. Place the macro in the Macro Name text box.
- 2 Choose Edit to display the Edit Macro dialog box. The code for the macro you selected appears in the Macro Formula text box.
- 3 Enter the arguments you want to insert within the parentheses () that follow the macro name. Make sure you separate the argument variables with commas.

Now you must tell ReportSmith the default values these variables should take when you run the macro.

- 4 Choose Set Default Arguments to display the Edit Macro Arguments dialog box.
- 5 In the Default Arguments text box, enter the default values (separated by commas) for the argument variables in the same order as the variables appear in the macro code. Press OK.

- 6 Press OK in the Edit Macro Arguments dialog box, again on the Edit Macro dialog box, and again on the Macro Commands dialog box.

Note You can also use the RunMacro command to provide a string to specify the values for the argument variables. If you specify a null string, ReportSmith uses the default arguments.

The following two macros show you how to pass arguments. In this case, you only have to place the argument in the called macro. If you are going to pass it back again, you need to place it in the calling macro as well.

```
Sub calling()  
  ' This variable allows you to pass along the double quotes and the value of the  
  variable.  
  q$=chr$(34)  
  var1$=q$+ "It is Monday"+q$  
  RunMacro "called",var1$  
End Sub  
  
Sub called(var1$)  
  msgbox "Hey!"&var1$  
End sub
```

Note For this macro make sure that you press the Set Default Arguments button in the Edit Macro dialog box. For string arguments, you must place it in double quotes. This gives it a value when it is first being compiled.

Arguments and events

When you link a macro to events, you can specify alternative values for the argument variables in that macro for each individual event. If you choose not to supply alternative values, ReportSmith uses the default values you specified in the procedure above.

To supply values for macros (containing arguments) on a link-by-link basis:

- 1 Select your macro. In the Macro Links dialog box, choose Links.
- 2 Use the Previous Link and Next Link buttons to display the event for which you want to specify argument variables.
- 3 Press the Set Arguments button to display the Edit Macro Arguments dialog box.
- 4 In the Default Arguments text box, enter the values (separated by commas) for the argument variables in the same order as the variables appears in the macro code. Press OK to return to the Macro Links dialog box. When you run the macro using the event you highlighted in Step 4, ReportSmith will replace the default argument values with the values you specified here.
- 5 To exit the macro facility, press OK in the Macro Links dialog box. Press OK in the Macro Commands dialog box.

Calling other macros

After you create a macro (A), you can create a second macro (B) that uses the functionality of macro A. You can use arguments to pass information between the two macros. When you do this, ReportSmith replaces the default argument values in macro A with the argument values supplied by macro B.

To run macro A from macro B, use the RunMacro command. The RunMacro command lets you run macro B and supply a string that represents the arguments in macro A.

Note The RunMacro command takes two strings. The first string is the name of the macro you want to run. The second string is the list of arguments whose values you want to replace in that macro. If your argument list contains strings, you'll need to use chr\$(34) to place the double quotes character within that second string.

The RunMacro command

The RunMacro command takes two string arguments. The first argument is the name of a macro to call. The second argument is a list of arguments to pass to the macro you are calling. If you have a macro that takes a string argument, the data for that argument must be surrounded by double quotes. One way to embed double quotes in a string is to use the CHR\$ function to create the quote character and then use the string concatenation operator to add it.

For example, to call a macro that takes four string arguments, RunMacro expects a string that would look like the following (if you display it with the MSGBOX command):

```
"String one", "String two", "String three", "String four"
```

You can generate this basic string as follows:

```
q$=chr$(34)
Arglist$=q$+"String one"+q$+", "+
q$+"String two"+q$+", "+
q$+"String three"+q$+", "+
q$+"String four"+q$
```

Variables can be passed only to a macro “BY VALUE” when using the RunMacro command, so the called macro cannot modify the variable value. You must use the variable to generate the string. For example, to call a macro named TestMacro that takes a number and a string as arguments, with the number contained in a variable (NUMVAR%), and the string contained in a variable (STRVAR\$), you can call the macro with the following code:

```
q$=chr$(34)
Arglist$=str$(NUMVAR%)+", "+q$+STRVAL$+q$
Status=RunMacro("TestMacro",arglist$)
```

Note that we used the STR\$ command to convert the numeric value to a string. Also note that we embedded quotes around the string variable that we passed. When executing this code through DDE, you must add carriage return and line feed characters between each line of code, or alternatively you can use two vertical bar characters (| |).

To use DDE to execute an active global macro called TEST_ARGS and pass two string arguments, ReportSmith expects the following string to be sent to its COMMAND topic:

```
q$=chr$(34)||RunMacro "Test_ARGS",  
q$+"Test String One"+q$+", "+  
q$+"Test String Two"+q$
```

The macro TEST_ARGS might look like this:

```
Sub TEST_ARGS (First_Argument$, Second_Argument$)  
    MsgBox "The first argument was:"+First_Argument$  
    MsgBox "The second argument was:"+Second_Argument$  
End Sub
```

Using macros with DDE: Two examples

This section shows two examples for using macros with Dynamic Data Exchange (DDE):

- Loading a Report Through SQLWindows
- DDE statement passing a string literal value to a report variable

You can use the following DDE parameters when calling ReportSmith from other Windows applications. All DDE calls to another application have three basic parameters: application, topic and item.

The application is always ReportSmith. There are two valid topics: System and Command. If you use the System topic, the item is a menu specification such as File | Open. If you use the Command topic, the item is a ReportBasic command such as LoadReport.

Note Refer to the calling application's documentation for instructions.

ReportSmith's sample macros are placed into the MACROS directory. You can see a list of them and a brief description of their functions using the Tools | Macro command. The following examples are also in the MACRO directory.

Example 1: Loading a report through SQL Windows

This code is from an SQL Windows application which sends macro commands to ReportSmith through DDE. It runs when a user presses a SQL Windows button called "pb1." The button initializes the values of datafield objects in a dialog box and then establishes a DDE connection to ReportSmith.

Note A ! indicates a comment.

```
Pushbutton: pb1  
Message Actions  
On SAM_Create  
    Set dfService ='ReportSmith'  
    Set dfTopic ='COMMAND'  
    Set dfItem = ' '  
    Set dfStatus ='Not Started'  
    Set dfSendit = 'DIM DS as Dataset'|| '||' || 'DS.CONNECT'
```

```

12, "", "SYSADM", "SYSADM", "SPA" || ' || ' || 'LOADREPORT' || ' || 'C:\SPAFON.RPT' || ' ', ' || '
"

! dfService, dfTopic, dfItem, dfStatus, dfSendit are SQL Windows data fields
! dfService can be ReportSmith (if you communicate to
! RPTSMITH.EXE) or
! dfService can be RS_Runtime (if you communicate to RS_RUN.EXE)
! The string c:\SPAFON.RPT in the dfSendit String is a report file
! USE UPPERCASE CHARACTERS ONLY WITH THE CONNECT COMMAND
! On SAM_Click When the user clicks the pb1 button, Connect toReportSmithCall
SalWaitCursor(TRUE)
If NOT SalDDEStartSession(mlBucket, dfService, dfTopic, dfItem, 10000)
Call SalWaitCursor(FALSE)
Call SalMessageBox('Could not connect to ReportSmith', '-Title', MB_Ok|MB_IconHand)
Set dfStatus = 'Connection failed'
Else
    Set dfStatus = 'Connected to ReportSmith'
    Call SalWaitCursor(FALSE)
! Runs when a user presses "pb2."
    Pushbutton: pb2
    Message Actions
    On SAM_Click
    Call SalWaitCursor(TRUE)
    If not SalDDESendExecute(frmMain, dfService, dfTopic, dfItem, 10000, dfSendit)
    Call SalWaitCursor(FALSE)
    Call SalMessageBox('DDE send to ReportSmith failed',
! Violated!, MB_Ok| MB_IconHand)
Else
    Set dfStatus = 'Sent DDE data to ReportSmith'
    Call SalWaitCursor(FALSE)

```

Example 2: Assigning a value to a report variable

This code shows you how to assign a string value to the macro language along with double quotes. Commands in the macro language require string variable parameters to contain double quotes. Therefore, when passing string values into the macro language from DDE, you need to pass the quotes as well.

```

q$ = Chr$ (34)
' Chr$(34) is the code for double quotes {"}
s$ = "1993-01-30"
e$ = "1993-12-30"
DDECommand$ = "LoadReport" + q$ + "C:\sample.RPT" + q$ + ", " + q$ + "@startdate = <" + s$
+ ">, @enddate = <" + e$ + ">" + q$

```

Using the Creation event

The creation event can be linked to two objects: Group Header and Group Footer. When you link a macro to the Header/Footer creation event you must choose the grouping level of the header or footer you want to link it to.

A macro linked to these events can call the ResumeEvent command with an argument of 0 to suppress the creation of an individual group header or footer based on the data in the report.

Suppressing a group footer

For example, suppose you have a set of groups and each has a number of records. If one of these groups has an insignificant number of items, the following macro would suppress the group footer for that group.

- 1 Create a new report using the CUSTOMER table.
- 2 Select the CUST_ID field and press the Footer button on the toolbar to sort and group by that field.
- 3 Choose Tools | Macro. Select the Report Macros option and enter a name, "ConditionalFooter", into the Macro Name box. Choose New button to open the Edit Macro dialog box. ReportSmith inserts the first and last lines of your macro into the formula for you.

- 4 Enter the following code between the first and last lines of your formula:

```
if Field$("CSTMTR_ID") = "C17" then ResumeEvent 0
```

ResumeEvent 0 cancels creation of the "C17" footer object, but only when linked to the Group Footer object Creation event.

- 5 Test your macro and after you receive a success message, press OK to return to the Macro Commands dialog box. Press Save and give the macro a name.
- 6 To link the macro to your report, choose the CUST_ID field from the Data Fields list, the Creation event from the Events list and the Group Footer from the Object Type list. Choose Links button, then press OK to return to the Macro Command dialog box. Press Save to give the macro a name.
- 7 Press Run to execute the macro in your report. ReportSmith runs the macro and suppresses the Group footer for the "C17" field.

Using the DataSet Control

The DataSet Control lets you define a description of report data. For example, you can use its properties and methods to make a connection, set a list of tables, set selection criteria, obtain a list of available tables, get a list of table owners, create a default columnar report, and refresh a temporary table for a report.

Creating a DataSet Control Object

You can create a DataSet control with Dim and Global statements. For example, to create a DataSet control called MyData, your code might look like this:

```
Dim MyData as DataSet
```

Modular scope

If you want all subroutines in a macro to be able to use your control, place the Dim statement outside of all subroutines. A macro declared in this way has modular scope and is destroyed when the macro ends.

Local scope

If you just want to use the control in a single function, place the Dim statement *inside* that function. This gives the control local scope. It is created when the function is called and destroyed when the function is complete.

This code fragment shows a function which uses a DataSet control to close the active report and to remove its connection:

```
Sub CloseActiveReport()  
    ' Make a local DataSet object for the report to be removed.  
    DIM DoomedData as DataSet  
    ' Associate this DataSet with the Active report  
    Trouble=DoomedData.SetFromActive()  
    If Trouble Then  
        ' Let the user know if there is a problem like no active report.  
        MsgBox DoomedData.Error$  
    Else  
        ' Otherwise close the report.  
        CloseReport 1  
        ' And Remove the Connection.  
        Trouble = DoomedData.DisConnect()  
    End If  
End Sub
```

Global Scope

You can use the Global statement to create a DataSet control that is known to all macros in ReportSmith. The Global statement must appear in all modules which refer to the global DataSet. Before the object is used, the global statement should appear on the first line of your macro before any subroutines or functions. These two macros, SetName and GetName, both use a common DataSet control.

Note Even though the control has the same name in both macro commands, they both must include the global statement in order to refer to the same Global control.

```
' Declare a Global Dataset  
Global GDS as DataSet  
Sub SetName()  
    GDS.Name$="Global Control"  
End Sub
```

Or,

```
' Declare a Global Dataset  
Global GDS as DataSet  
Sub GetName()  
    MsgBox GDS.Name$  
End Sub
```

Properties and methods

Properties

The DataSet control is used by employing its properties and methods. *Properties* are data values that make sense for the control. They are similar to Basic variables and can

have any of the Basic variable types. You can access the properties of a control by using the name of the control, followed by a period and the name of the property.

One property of the DataSet control is Selection\$. It is a string variable that defines the SQL text following the WHERE clause in an SQL statement. You can use this property to set the selection criteria for a DataSet as in the example below.

```
' Display the Old Selection Criteria.
Msgbox MyData.Selection$

' Set the New Selection Criteria.
MyData.Selection$ = "SALARY > 40000"
```

You can access these variables in the same way as you access object methods by using the object name followed by a '.' and the property name. You use this just like any other variable. Some properties are read-only while others can be both read and written.

The following table lists some examples of how properties are used.

Use command	<code>msgbox MyData.name\$</code>
Use it in an expression	<code>total_recs=data1.recordcount+data2.recordcount</code>
Assign values to	<code>MyData Name\$="My Name"</code>

Methods

Methods are functions or statements that perform actions on the control. The SetFromActive method associates the DataSet control with the currently active report. The Save method can be used to store a DataSet control in a DOS file.

Sample macros

ReportSmith provides sample macros in the Video subdirectory. The sample macros show how you can use the macro language to perform sophisticated menu tasks, like conditional formatting, creating derived fields, menu customizing, printing and loading a series of reports.

You can create macros to produce more detailed tasks, such as calculating a percent of total. Or, you can use macros as "wizards" to perform commonly requested functions, like filling list boxes with .RPT files in a given directory, allowing users to run and print them from the dialog box.

This section documents the actual code for several of the available sample macros. These samples show you how to:

- Create conditional formatting
- Prompt for report variables
- Load a series of reports
- Calculate a person's age
- Connect to a database
- Display a greeting
- Place a line counter field on a report

- Determine a percent of total
- Sum the value of another Macro Derived Field
- Generate a dialog box using all available functions
- Refresh a temporary table with the DataSet object

Each sample includes comments to help familiarize you with the macro process. For detailed information on the other sample macros, see the README.TXT file.

Note The first two sample macros are report macros that use the *EX_MAC01.RPT* sample report. It displays a list of film titles, the film type, and the price of each film. Before you review the sample macros, open *EX_MAC01.RPT* so you can see the effect the sample macros have on that report.

Example 1: Conditional formatting

The code fragment in this section creates a sample macro that performs conditional formatting. When you want a macro to do conditional formatting, you create it based on criteria to which certain values in the report columns can correspond.

This macro changes the color of the data fields to which the macro is linked. The film type, which is specified in the "Film_Type" column of the report, determines what color the data field is. Later you can link the macro to other data fields so you can see how the same macro can be used to conditionally format more than one column in the same report.

When you create a conditional formatting macro, you can use the `FieldFont` and `FieldText` macro commands. These commands format the values that fulfill the specified criteria.

```
Sub Color_Type()

    ' Use the Rgb command to make the colors and assign them to variables.
    Red = RGB( 255,0,0 )
    Blue = RGB( 0,0,255 )
    ' Get the film type for the current record and store it in a local variable,Film_Type$.
    Film_Type$.FILM_TYPE")
    ' Calling Field Font with a color value of negative one (-1) means don't change color.

    Color = -1
    ' Use different colors for different film types.
    If Film_Type$ = "Action"           Then Color = Red
    If Film_Type$ = "Drama"            Then Color = Blue
    If Film_Type$ = "Science Fiction"  Then Color = Green
    If Film_Type$ = "Cartoons"         Then Color = Purple
    ' Use the Field Font function to change the fonts color

    FieldFont "",0,-1,Color,-1
End Sub
```

Example 2: Setting report variables

Report variables let you create prompts or dialog boxes that let a user specify values for a query before running a report. The report displays data that corresponds to the values

specified by the user. Using the scenario from the previous example, suppose you created a report variable that prompts the user to choose a film type to display in a report. ReportSmith uses the report variable in the selection criteria to determine the correct records to display in the report. For example, if the user specifies the “foreign” film type, the report displays foreign film types.

In addition, suppose the user wants to display all “classic” films. To do that, the user changes the value of the report variable from “foreign” to “classic.” In this case, automating the process of changing the film type is desirable.

You can automate this by creating the following macro and linking it to the **Before Report Load** event. This way, you can prompt the user to set all the report variables before ReportSmith runs the report.

```
' Asks you what type of movies to include in the report each time you load the report.
Sub Choose_Type()
' Define the dialog box that this macro uses to determine what film types to include in
the report.
Begin Dialog ChooseTypes 120, 130
    Caption "Choose Film Types"
    CheckBox 15,8,99,12,"Action", .Action
    CheckBox 15,20,99,12, "Cartoons", .Cartoons
    CheckBox 15,32,99,12,"Clasics", .Classics
    CheckBox 15,44,99,12,"Drama", .Drama
    CheckBox 15,56,99,12,"Foreign", .Foreign
    CheckBox 15,68,99,12,"Musical",.Musical
    CheckBox 15,80,99,12,"Science Fiction",.SciFi
    OkButton 30,100,60,18
End Dialog
' Create an instance of the ChooseTypes dialog box and call it MovieTypes.
Dim MovieTypes as ChooseTypes
' Execute the MovieTypes dialog box.
DIALOG MovieTypes
' Set the Report Variables for each film type according to
' the state of the checkboxes when the dialog is done.
If MovieTypes.Action Then SetRepVar "Type1"," 'Action' "
else SetRepVar "Type1"," 'Not Included' "
If MovieTypes.Cartoons Then SetRepVar "Type2"," 'Cartoons' "
else SetRepVar "Type2"," 'Not Included' "
If MovieTypes.Clasics Then SetRepVar "Type3"," 'Classics' "
else SetRepVar "Type3"," 'Not Included' "
If MovieTypes.Drama Then SetRepVar "Type4"," 'Drama' "
else SetRepVar "Type4"," 'Not Included' "
If MovieTypes.Foreign Then SetRepVar "Type5"," 'Foreign' "
else SetRepVar "Type5"," 'Not Included' "
If MovieTypes.Musical Then SetRepVar "Type6"," 'Musical' "
else SetRepVar "Type 6"," 'Not Included' "
If MovieTypes.SciFi Then SetRepVar "Type7"," 'Science Fiction' "
else SetRepVar "Type7"," 'Not Included' "
End Sub
```

Example 3: Loading a series of reports

This macro loads a series of reports. Because the macro can affect many reports, it's a global macro, linked to the ReportSmith application. When you run the macro, it displays a dialog box that prompts users to enter the reports they want to run, separated by a comma.

For each report you list, you must enter the complete path where the report is located, including the .RPT file extension.

```
Sub Report_Loader()  
    ' Get the list of reports from the user.  
    Reports$ = InputBox( "Enter the list of reports to run, separated by commas")  
    ' Start with the first report in the list.  
    ReportNumber = 1  
    LoadErrors$ = "Multiple Report Loading Result"  
    ' Get the first item in the list of reports, which are separated by commas.  
    Report$ = GetField$(Reports$,ReportNumber,",")  
    ' When you get to the end of the list, you get a NULL string.  
    While Report$ <> ""  
        ' Load the report.  
        Status = LoadReport( Report$, "" )  
        ' If the report failed to load, then build a message to let the user know.  
        If Status <. 0 Then  
            LoadErrors$ = LoadErrors$ + Chr$(13) + "Error Loading: " + Report$  
        End If  
        ' Get the name of the next report.  
        ReportNumber = ReportNumber + 1  
        Report$ = GetField$(Reports$,ReportNumber,",")  
    Wend  
    ' If the LoadErrors$ String has not changed, no errors occurred, so add a message  
    If LoadErrors$ = "Multiple Report Loading Result" Then  
        LoadErrors$ = LoadErrors$ + chr$(13) + "All Reports Opened Successfully!"  
    End If  
    ' Display the result to the user.  
    MsgBox LoadErrors$  
End Sub
```

Example 4: Connecting to a database

This macro shows you how to create a connection through the macro language. The example below connects to an Oracle database.

```
Sub db_login()  
    Dim DB as DataSet  
    DB.Connect 7,"x:orasrv7","system","manager",""  
End Sub
```

Example 5: Displaying a greeting

This macro displays a greeting, using the name stored in the USER environment variable. You can link it to the opening of ReportSmith, the opening of a specific report, or both.


```

Sub Greeting()
' Use the Environ function to get the user name and store it in a string variable.
UserName$ = Environ$("USER")
' If the USER environment variable is not defined a null string will be returned.
If UserName$ = "" then
MsgBox " Welcome to the Sample Macro Report ",0," Macro Greeting "
else
MsgBox "Hello" + UserName$ + ", Welcome to the Sample Macro Report ",0,"Personal Macro
Greeting "
End If

```

Example 6: Creating a counter field

This macro defines a derived field that numbers the records in a report.

```

Sub RecordNumber()
' Current gets the current record number.
' Str$ converts the number to a string.
' DerivedField sets the string as the value to be used for the derived field.
DerivedField Str$(Current)
End Sub

```

Example 7: Determining a percent of total

This exercise shows you how to create a macro which determines a percent of a total. Typically, ReportSmith uses SQL code to get the results of a derived field formula, so your database calculates the derived data. To derive a percent of a total, you need to select the **Defined by a Macro** option to create the derived field locally. This is because SQL code can't access summary fields, which are calculated locally.

To create a percent-of-total derived field:

- 1 Enter the name for the percent of total derived field you're creating in the Derived Field Name box in the Derived Fields dialog box. Turn on the Defined by a Macro option.
- 2 Choose Add.
- 3 Enter a name for the macro in the Macro Name text box and an optional description of it in the Description box.
- 4 Choose Edit. The Edit Macro dialog box appears.
- 5 Enter the macro code in the Macro Formula window:

```

Sub PerOfTot()
' Calculate the percentage of the total quantity of this
' employee's sales that the current record's quantity of sales represents.
' The Field$ function gets the QUANTITY record as a string.
' The Val( ) function converts it to a number which is stored in the variable
"Quantity".
Quantity = Val(SumField$("SUBTOTAL","c:\rptsmith\samples\invoices.dbf",2,"Sum"))
' The Next line gets the total quantity for the current employee group and assigns it
to a variable. The number 1

```

```

' refers to group level 1 which in this case is the employee group.
' 0 would indicate the sum for all groups.
TotalQuantity = Val(SumField$("SUBTOTAL", "c:\rptsmith\samples\invoices.dbf", 1, "Sum"))
' Because the above functions may fail, test for a zero value before doing division.
' A divide by zero error will halt macro execution.
If TotalQuantity > 0 then
PercentOfTotal = Quantity/TotalQuantity
' The Derived Field statement sets the derived field value and is only valid in
' macros used to define Derived Fields.
' It takes a string value, so the Str$ Function is used to convert the Number to a
string.
    DerivedField Str$(PercentOfTotal)
End If
End Sub

```

6 Choose OK. The Create a Macro dialog box appears. Choose OK to exit the dialog box, then press the OK button again to exit the Derived Fields dialog box.

7 Choose Insert | Field to drop the derived field as a column of data in the report.

Our example report shows percent of total sales for William Blake for each of his customers. We want to show that percentage in the customer group footer next to the sum of sales for each customer.

8 Select and delete the column for the percent of total derived field and then insert it into the customer group footer.

9 Use the Percent button on the ribbon to format the derived field.

The following are examples of macro derived fields that sum the values of another macro derived field. In the examples, substitute the field name of the macro derived field to be summarized in place of "MDF." Also, substitute the name of the field that the data is grouped on, in place of "GROUP," if you are working with grouped data. If the report is grouped on more than one field and you want a summary for each group, create a macro derived field for each group.

Example 8: Creating a summary

This macro summarizes the value of another macro derived field in a report with a group break.

```

Sub sum_mac_group()
    Group$ = Field$("GROUP")
    while (Group$ = Field$("GROUP") AND Current() > 1)
        TOTAL = TOTAL + Val(Field$("MDF"))
        GetPrevious
    wend
    if (Group$ = Field$("GROUP") AND Current() = 1) then
        TOTAL = TOTAL + Val(Field$("MDF"))
    end if
    DerivedField Str$(TOTAL)
End Sub

```

Example 9: Sample dialog box

This macro generates a dialog box which contains examples of various types of dialog box functions. The value contained within a variable as a result of selecting a radio button, push button, list box item, or drop-down list box item is 0 for the first item, 1 for the second, and so on. The variable value for a Check Box is 0 unchecked, 1 checked. The variable value for a Text Box is the value typed into the Text Box. To use a dialog box variable as a part of another ReportBasic statement, the variable name must be preceded by the dialog box name, for example (Test_Box.RadioValue).

```
Sub Sample_Dialog_Box()
' Values to populate the list boxes
ListValue$ = "Lucy"+Chr$(13)+"Ricky"+Chr$(13)+"Fred+
Chr$(13)+"Ethel"
' An error routine is necessary for the Cancel Button to function correctly
On Error GoTo Cancel:
' All of the following functions are described in detail in the ReportBasic online help
Begin Dialog Samp_Dialog 275,200
Caption "Sample Dialog Box"
GroupBox 15,10,60,60, "Radio Buttons"
OptionGroup .RadioValue
OptionButton 25,25,40,12,"Larry"
OptionButton 25,37,40,12,"Moe"
OptionButton 25,49,40,12,"Curly"
GroupBox 15,90,60,60, "Check Boxes"
CheckBox 25,105,40,12,"Andy",.AndyValue
CheckBox 25,117,40,12,"Barney",.BarneyValue
CheckBox 25,129,40,12,"Opie",.OpieValue
GroupBox 90,10,100,34, "Text Box"
TextBox 100,22,80,12,.TextValue
GroupBox 90,56,100,46, "List Box"
ListBox 100,69,80,36,ListValue$,.ListValue
GroupBox 90,115,100,35, "Drop Down List Box"
ComboBox 100,128,80,36,ListValue$,.ComboValue
GroupBox 205,10,55,140, "Push Buttons"
ButtonGroup .ButtonValue
Button 215,25,35,15,"Greg"
Button 215,45,35,15,"Peter"
Button 215,65,35,15,"Bobby"
Button 215,85,35,15,"Marsha"
Button 215,105,35,15,"Jan"
Button 215,125,35,15,"Cindy"
OkButton 50,175,50,15
CancelButton 175,175,50,15
End Dialog
DIM Test_Box as Samp_Dialog
' Dialog box execution
Dialog Test_Box
Cancel:
Exit Sub
End Sub
```

Example 10: Refreshing a temporary table

This macro prompts a user to refresh the temporary table from which a report has been built. Link this report level macro to the **Before Report Load** event.

```
Sub load_ds()  
  ' Display message box asking user if they want to refresh the temp table data.  
  refresh = MsgBox("Refresh Temp Table mytable",4,"Temp Table")  
  ' If answered yes, refresh. Otherwise end the macro.  
  if refresh = 6 then  
    ' refresh table and drop existing data.  
    execsql "drop table c:\rptsmith\data",55,"dBase","","",""  
    ' Create a DataSet object to hold the temp table values.  
    dim ds as dataset  
    ' Load in the report definition file that created the temp  
    ' table.  
    ds.load("c:\rptsmith\test3.rqf")  
    ds.name$ = "test3"  
    ' Rerun the query to return the newest information.  
    ds.recalc  
    ' Export the table to the database for this new report to use.  
    ds.exporttable "c:\rptsmith\data\mytable",55,"dBase","","",""  
  end if  
End Sub
```

Printing a report

This chapter explains how to print an entire report, a range of report pages, and the report page displayed in your report window at the time of printing. The printer you use influences how ReportSmith shows and prints your report.

With ReportSmith, you can decide what data, headers, or footers to display and print in a report. This is useful when you want to create summary reports. You can also determine where page breaks occur in a report.

In this chapter, you'll learn how to set up the printer and print a report.

Setting up the printer

To prepare for printing, tell the computer what type of printer you're going to use. Not all printers have the same capabilities. The dialog box that appears when you select the Print Setup command depends on the type of printer you use.

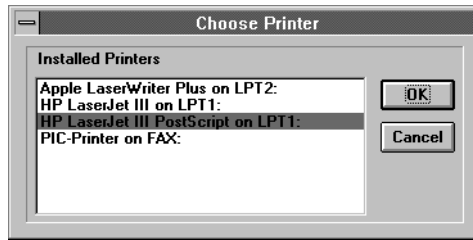
Most printers can adjust for various paper sizes, different orientations, and paper sources. You can set the default paper size, and so forth, for your reports with the Page Setup command on the File menu. You can also use the Print Setup command on the File menu to do this for specific reports or parts of reports.



The settings in the Page Setup dialog box override the settings in the Print Setup dialog box if there's a difference between the two.

Selecting the printer

Choose File | Print Setup. The Choose Printer dialog box appears.



- 1 Choose the printer you'll use from the Installed Printers list box.
- 2 Press OK. The dialog box for the printer you choose appears. Available options depend on the printer you've selected, but typically you set paper source and paper size.
- 3 Select a paper source option. The following options typically appear for most printers:

Default Tray	Tells the printer to use the default printer tray when printing the first page of the report.
Upper Tray	Tells the printer to use the upper tray when printing the first page of the report.
Lower Tray	Tells the printer to use the lower tray when printing the first page of the report.
Manual Feed	Tells the printer that you want to manually feed each page when printing the report.
Envelope Feed	Tells the printer that you want to print an address on an envelope before printing the report.

- 4 Choose the paper size you want to use from the drop-down list.

Note Depending on the printer you use, you may have to set other options.

Printing a report

You can print

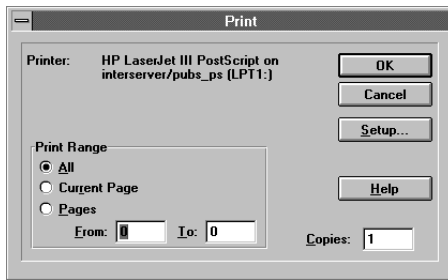
- The current page appearing in a report window
- A range of pages from a report
- An entire report

Use the Presentation command on the View menu to see how your report looks before you print it. Before printing, always remember to save your report.

To print the current page :

- 1 Open your report.

- 2 Choose File | Print. The Print dialog box appears.



- 3 Select Current Page.
- 4 Enter the number of copies you want to print in the Copies window. The default is 1.
- 5 Press OK. ReportSmith prints the page displayed in the report window.

To print a range of pages in a report:

- 1 Choose File | Print. The Print dialog box appears.
- 2 Under the Pages option, type the beginning page number into the From Box and the ending page number into the To box.
- 3 Enter the number of copies you want to print in the Copies window. The default is 1.
- 4 Press OK. ReportSmith prints the pages you selected.

To print an entire report:

- 1 Choose File | Print. The Print dialog box appears.
- 2 Do one of the following:
 - Under the Print Range, select All. Enter the number of copies you want to print in the Copies window. The default is 1.
 - Press the Print button on the toolbar.
- 3 Press OK. ReportSmith prints the whole report.

To cancel printing, click the Cancel button in the Print dialog box.

Customizing ReportSmith

You can customize the ReportSmith environment to suit your needs using the Options dialog box. Specify whether to load data locally to the client memory or disk, keep it on the server, or let ReportSmith determine the best method for your report. Additionally, you can specify units options, tell ReportSmith where to find datasource connections, and more. This chapter shows you how to set ReportSmith's options, customize ReportSmith's menu and toolbar, and run a report from an icon.

Setting options

To customize ReportSmith options:

- 1 Choose Tools | Options to open the Options dialog box.

Options

Dynamic data access parameters:

☐ Client memory

☐ Client disk

☐ On server

☒ Automatic

☐ Minimize report data memory usage

☒ Limit report data memory usage to

☐ Limit report data memory usage to

50

% of memory

KB of memory

Buffer 100 records

Program startup options:

☐ Open report

☐ New report

☒ None

Units: ☒ Inches

☐ cm

Granularity: 300

☒ Update units during file open

Options:

☐ List fields alphabetically

Font types listed:

☐ Screen

☒ Printer

☐ Screen and Printer

☐ TrueType

Connection file:

Picture file search path:

C:\WINDOWS\rptsmith.con

c:\rptsmith\samples

OK

Cancel

Help

Choose how to load data

Choose startup options

Set rulers to inches or centimeters

List fields in dialogs alphabetically

Specify a default font type

Specify where to store connections

Specify where to find graphics

- 2 Specify the settings you want for each option.

Dynamic Data Access	Controls which resources ReportSmith uses to store data. ReportSmith uses Dynamic Data Access (DDA) to recognize the size of data and to determine the best strategy for transferring it into a report.
Program Start-up	Sets what dialog box you see when the ReportSmith application opens.
Units	Sets the default measurement system ReportSmith uses to inches or centimeters. You can also set the resolution (granularity) ReportSmith uses when you place objects in a report.
List Fields Alphabetically	Check this option to see all items in dialog boxes listed in alphabetical order.
Font Types Listed	Sets the default font types ReportSmith uses to Screen, Printer, Screen and Printer, or True Type fonts.

Connection File	Sets the directory where ReportSmith stores your named connections set up with the Connections command on the File menu.
Picture File Search Path	Points to the location(s) where ReportSmith can find graphic images to use with the Display Data as Picture feature. ReportSmith searches this path for any graphic files you have in your database. Separate multiple directories with commas, semicolons, or spaces.

Controlling data access

The options you select in Dynamic Data Access parameters control which resources ReportSmith uses to store and move data. Select from the following parameters:

Option	Description
Client Memory	Click this option if you always want the data loaded into client (meaning “your local machine’s”) memory.
Client Disk	Click this option if you always want the data loaded onto disk. Specify the data path for the disk in the RPTSMITH.INI file
On server	Click this option and ReportSmith leaves all the records on the server except the number of records specified in the Buffer option
Automatic	Click this option to have ReportSmith evaluate where to place the data. Automatic is the default option. You’ll see the following settings when you click this option: Minimize Report Data Memory Usage Evaluates available disk space and keeps the data on the server if there isn’t enough room on the disk. Limit Report Data Usage to Percentage of Memory Limits memory usage to 50% by default. ReportSmith estimates the memory requirements for data storage. If the data requires more than 50% of the available memory, ReportSmith looks at the available disk space. If the disk space is insufficient, the data stays on the server. You can increase or decrease this setting by entering the percentage you want to use. Limit Report Data Memory Usage to Kilobytes ReportSmith estimates memory requirements for data storage using the number of kilobytes (KB) you enter for this option instead of a percentage limit. Enter the number of KB in the text box for this option. Buffer Records Controls the size of the buffer when ReportSmith accesses data left on a server.

Choosing start-up options

The options you select in Program Start-up control which dialog boxes appear when you start the ReportSmith application. Select one of the following:

Option	Description
Open Report	Click this option and ReportSmith automatically displays the Open Report dialog box when the application opens.
New Report	Click this option and ReportSmith automatically displays the Create A New Report dialog box for you to choose a table and create a new report.
None	Click this option and ReportSmith opens to the ReportSmith workplace, with no dialog boxes displayed. You can then select the command you want.

Selecting measurements

Use the Units option to set the default measurement system to inches or centimeters.

Option	Description
Inches	Click in this option if you want ReportSmith rulers displayed in inches as the scale of measurement. This is the default setting.
Cm	Click in this option if you want ReportSmith rulers displayed in centimeters as the scale of measurement.
Granularity	The Granularity setting determines the finest resolution for placing objects in a report. Higher granularity gives a finer resolution on the placement of objects in the report, but limits the maximum size of the page and workspace in the report. ReportSmith strongly recommends using a granularity of at least 300 for inches, 118 for centimeters. Using a higher granularity is acceptable, but if you use a lower granularity, it's likely to cause problems. You would only use a lower granularity if you need to display extremely wide or tall pages.
Update Units During File Open	If your report was created with different units options than specified in the Options dialog box, choose this option to have ReportSmith convert your report to use the options you specify. (See Granularity, above, for further information.)

Listing items alphabetically

Check the List Fields Alphabetically option to list all items in dialog boxes in alphabetical order.

Choosing a font list

The options you select in the Font Types parameters let you specify which list(s) of fonts you can choose from when using ReportSmith. Whenever you need to select a font in ReportSmith, the list you are presented with is controlled by this option. Select from the following parameters:

Option	Description
Screen	Click this option to select screen fonts each time you use ReportSmith. (When you choose this option, ReportSmith includes TrueType fonts.)
Printer	Click this option to select only fonts that are installed on your printer each time you use ReportSmith.
Screen and Printer	Click this option to select both screen and printer fonts each time you use ReportSmith. (When you choose this option, ReportSmith includes TrueType fonts.)
True Type	Click this option to use TrueType fonts (scalable fonts that you can change and that appear exactly as they'll print) each time you use ReportSmith.

Storing connections

The Connection File option stores the connections you create through File | Connections. This option sets the location and name of your connection file.

The default Connection file name is RPTSMITH.CON. ReportSmith stores this file in your Windows directory. If your Windows working directory is on a local area network (LAN), give the connection file for each workstation a unique file name, with a three-

character .CON file extension. To change this option, place the text cursor in the text box to enter the drive, directory, and file name for the connection file.

Pointing to graphics files

The Picture File Search Path option sets the locations (directories) that you want ReportSmith to search for graphics files used in conjunction with the Display As Picture option.

Enter the drive and directory path of the files you want ReportSmith to search for, separated by semicolons and backslashes. Instead of searching through all directories, ReportSmith saves time by only searching through the files you specify here.

Customizing the menu and toolbar

You can easily customize the ReportSmith environment to suit your needs by changing its menu options or the toolbar. You can use the ReportBasic macro language to add, enable or disable menu items or toolbar icons.

The following list describes the macro commands you use most commonly to perform these tasks:

Command	Function
AddMenu	Allows you to add your own commands to ReportSmith's menu.
EnableMenu	Enables or disables a menu command.
EnableIcon	Enables and disables icons and combo boxes on the toolbar and ribbon.
IsMenuChecked	Lets you determine if a given menu item is turned on.
IsMenuEnabled	Lets you determine if a given menu item is enabled or disabled.
KillMenu	Lets you remove one of the ReportSmith menu items.

In the following exercises, we'll show you how to write a macro that disables the New command and the New Report button. Then, we'll show you how to add your own menu item that automatically loads a report that you use often.

Disabling File|New

Suppose you don't want a user to be able to create a new report (you only want the user to be able to open existing reports). You can disable the File | New command and the New Report button on the toolbar. We'll show you how to disable these options in the following example. In it, we'll create a global macro, CUSTOM_MENU that is linked to the Application Startup. Each time you start ReportSmith, it runs the CUSTOM_MENU macro, disabling the New command and toolbar button.

To disable the New command and the New Report button:

- 1 Open a report and choose Tools | Macro. The Macro Commands dialog box appears.

- 2 In the Macro Name text box, enter a name for your macro. In our example, we named our macro CUSTOM_MENU.
- 3 Turn on the Global Macros option.
- 4 Press the New button. The Edit Macro dialog box appears.

Notice that ReportSmith has placed the first and last lines of your formula into the text window for you. You do not need to enter them again.

- 5 Enter the following formula into the Macro Formula text box:

```
Sub custom_menu()  
    EnableMenu "File|New",0  
    EnableIcon 1,1,0  
End Sub
```

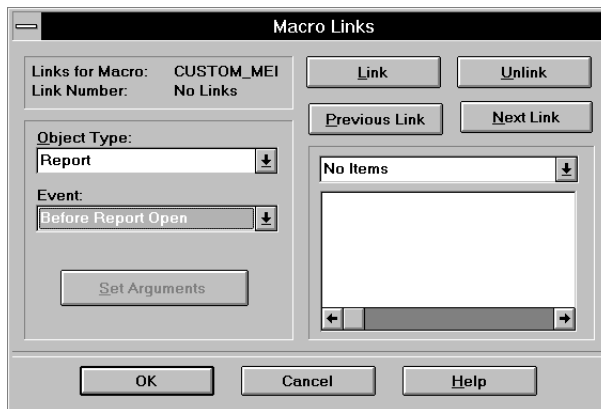
Your macro formula should look like this:

```
Sub CUSTOM_MENU()  
EnableMenu "File|New",0  
EnableIcon 1,1,0  
End Sub
```

- 6 After you've entered your formula, press OK to return to the Macro Commands dialog box.

Now, you'll link the macro to an event.

- 7 Press the Links button to open the Macro Links dialog box.



You can link your macro to a variety of events including a keystroke, before printing a report, or after opening the report. For this example, you'll link the macro to the Application Start Up event.

- 8 Under Object Type, choose Report. Under Event, choose Before Report Open and choose Link. ReportSmith displays "1 of 1" in the Link Number text box, indicating that you have created one link and you have a total of one link. Press OK.

Adding a custom menu item

Suppose you want ReportSmith to automatically load a report you use often. You can write a macro that creates a custom menu item that, when you select it, loads the report for you. The following example shows you how.

First you'll create a global macro and link it to the Application Startup event.

To add a custom menu item:

- 1 Open a report and choose Tools | Macro.
- 2 Turn on the Global Macros option.
- 3 In the Macro Name text box, enter a name for your macro. In this example, we named our macro BRING_UP_REPORT.
- 4 Press the New button. The Edit Macro dialog box appears.

Notice that ReportSmith has placed the first and last lines of your formula into the text window for you.

- 5 Enter the following formula into the Macro Formula text box:

```
Sub bring_up_report()  
    AddMenu "My Report", "load_mac", "File|New", "This loads my report"  
End Sub
```

Your formula tells ReportSmith to add a menu item called "My Report" after the New command on the File menu.

Note

For more information on the AddMenu command, see online Help.

When you choose the My Report command, it runs a macro called LOAD_MAC. LOAD_MAC contains the following code:

```
Sub load_mac()  
    LoadReport "c:\before.rpt", ""  
End Sub
```

LOAD_MAC tells ReportSmith to load the report called BEFORE.RPT.

- 6 When you've finished entering your macro formula, press OK.

Running a report from an icon

You can run a report automatically from an icon. To do this, you need to create an icon which, when selected, starts ReportSmith and points to a specific report.

To load a report from an icon, you'll create a new icon in the Windows Program Manager.

- 1 In the Windows Program Manager, File | New to open the New Program Object dialog box.
- 2 Choose the Program Item option and press OK. The Program Item Properties dialog box appears.

- 3 In the Description text entry box, type text you'd like to appear for your icon. For example, you could place the name of your report here.
- 4 Under Command Line, enter the RPTSMITH.EXE, followed by a space and the location of your report.

For example, you might type the following to run a report called "Sales" from your MYRPTS directory:

```
RPTSMITH.EXE C:\MYRPTS\SALES.RPT
```

This command automatically starts the ReportSmith application with the sales report loaded in it.

- 5 Under Working Directory, enter the directory where the ReportSmith executable is located.

For example, you might type the following:

```
C:\RPTSMITH
```

- 6 Press OK and your icon appears. You can double-click on it whenever you want to run your report. When you select the icon it runs the ReportSmith program, so you should select the icon when ReportSmith is not running.

Runtime Viewer

The Runtime Viewer lets an unlimited number of users use ReportSmith without owning the full product. Users can execute, view, and print reports, but they cannot create or modify reports. The Runtime Viewer supports all functions that ReportSmith performs through Dynamic Data Exchange (DDE) and ReportBasic, so you can create dynamic reports that change according to your user's needs.

The Runtime Viewer allows you to:

- Set up dialog boxes containing options that end users can fill in to customize their reports, such as sorting, selections, and range of records, and pass this information to the Runtime.
- Set up multiple buttons in your front-end application, such as a “Print Summary Report” button or a “Print Sales Report” button to automatically print frequently used reports.
- Prompt for report variables, display user-defined dialog boxes, and respond to DDE messages.

Overview

Follow these steps to set up the Runtime Viewer for your end users.

- Install Runtime.
- Setup necessary drivers. For Runtime to operate you need to set up the user's environment similar to the environment in which the reports were created in, including data source names, named connections, Dynamic Data Access options and global macros.
- To make system administration easier and to ensure that Runtime operates smoothly for the end user, set up ReportSmith's configuration files so they are compatible with your reports.

Installing the Runtime Viewer

Note Before installation, we recommend that you create backup copies of disks and that you install Runtime using these backup copies.

To install Runtime,

- 1 Insert **Disk 1** into your floppy drive.
- 2 In Windows, choose File | Run from the Program Manager. In the Run dialog box, type A:\INSTALL (or B:\INSTALL) and choose OK.
- 3 Follow the instructions in the dialog boxes that appear. If you have questions about the options in any dialog box, choose Help from that dialog box.
- 4 When prompted, insert the specified disk into your disk drive and press *Enter*. You will be prompted to choose the type of installation you want.
- 5 After installation, INSTALL prompts you to open the Release Notes. They contain important information that supersedes the printed documentation. Choose View Release Notes to read them, or Exit to return to the Program Manager.



Release
Notes

Note To print the Release Notes, use a text editor (such as Notepad).

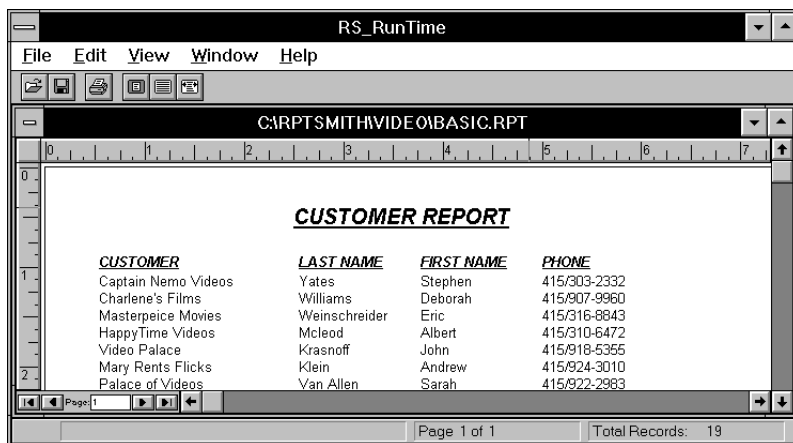
You are now ready to use ReportSmith's Runtime Viewer.

Using Runtime Viewer



RunTime
Viewer

To use Runtime Viewer, double-click on the Runtime Viewer icon in the ReportSmith program group. The Runtime Viewer appears as shown here displaying a report.



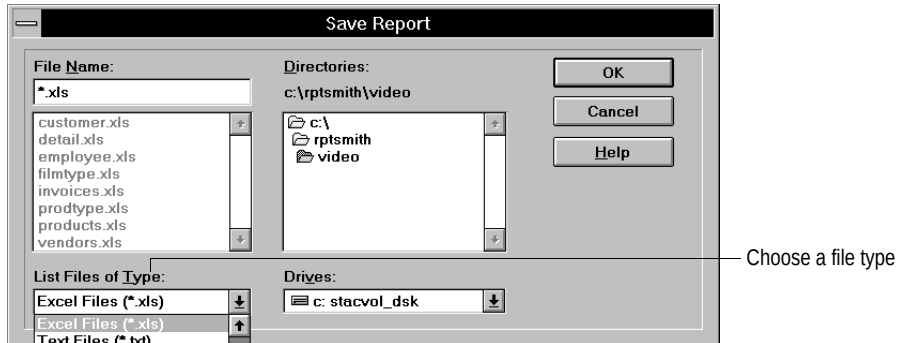
The menus and toolbar contain commands that you can use to open, view, and print reports. In the Runtime Viewer, you can:

- Open multiple reports at one time.
- Scroll and page through a report.
- Use the Zoom buttons to view reports.

- Use the Save button to export data to different file formats such as Excel, Lotus, and ASCII text.
- Set Options for the Runtime Viewer. See the following section.

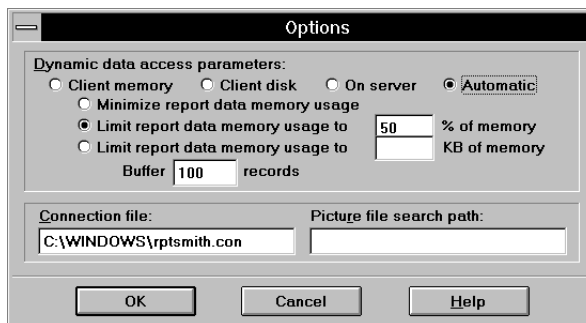
Exporting data

With the Save As feature, you can export data for use in other applications. However, you cannot save changes to the report format using the Runtime Viewer. To modify a report, use the full ReportSmith product. To export data, use the Save button, or the File | Save As command. Choose a file type from the List Files of Type drop-down list. Choose a location and name for your file and choose OK.



Setting options

The Runtime Viewer allows you to set Dynamic Data Access (DDA) parameters using the File | Options command. The Options dialog box is shown below.



The options you specify here are saved in the RS_RUN.INI file in your Windows directory.

Sample DDE code

The following example allows you to control ReportSmith's Runtime Viewer from Delphi. Any application that can send DDE commands can also control the Runtime

Viewer. The following code fragment shows you how to drive Runtime using a VisualBasic application. This code is executed when the end user presses the button "Command 1." It loads and prints "MyReport."

Note The Runtime Viewer must be running for this code to execute.

```
Sub Command1_Click()
    q$=Chr$(34)
    ' Open a DDE conversation with ReportSmith's system topic.
    Label.LinkTopic="RS_RUNTIME|Command"
    Label1.LinkMode=2
    ' Send a DDE Command to ReportSmith using the Service Topic.
    ' The text in the DDE Command is of the form: LoadReport "MyReport.rpt", "
    ' the second argument is used to initialize report variables.
    DDECommand$="LoadReport"+q$+text1.Text+q$+", "+q$+q$
    ' Create the command to print the report.
    SecondCommand$="PrintReport0,0,"+q$+q$+", "+q$+q$+", "+q$+q$
    ' Create the command to close the report.
    ThirdCommand$="CloseReport(0)"
    ' Multiple commands may be separated by two vertical bar characters.
    DDECommand$=DDECommand$+"||"+SecondCommand$+"||"+ThirdCommand$
    ' Execute the command string that was built above.
    Label1.LinkExecutedDDECommand$
End Sub
```

Creating and using a data dictionary

The ReportSmith Data Dictionary allows you to present users with a simplified view of data. You can filter your data by defining subsets of databases, connection information, table and field aliases, headings, and groups of functionally related tables.

For example, an end user might typically use data from several SQL Server and dBASE databases in one working session. You can create a dictionary that stores connection information, tables and fields, aliases, defined groups, and headings for these databases.

A data dictionary allows you to:

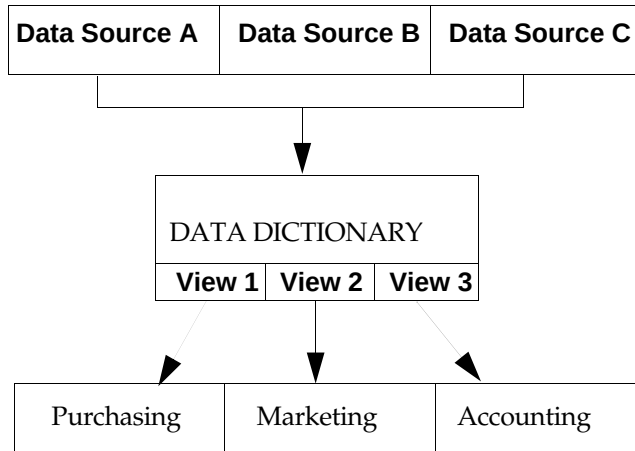
- Rename tables and fields with easily recognizable names.
- Present users with a subset of the total set of tables and fields within a database to simplify their interactions.
- Present users with defined groups, so they can view various groupings of functionally related tables.

A data dictionary can contain information about any supported data sources. However, the data dictionary information must be stored in one of five data sources: ORACLE, SQL Server, dBASE, SQLBase, or InterBase.

Understanding a data dictionary

A data dictionary enables you to create a unique view of information for a department based on its reporting needs. For example, the Purchasing, Marketing, and Accounting departments all view data based on the activity of selling. Each department has its own needs and requires a different view of the same data. The Purchasing department needs to see items that need to be restocked, the Marketing department needs customer information, and the Accounting department needs invoice information.

You can create a data dictionary that presents each department with a customized view of the data.



Overview

To create a data dictionary, you can:

- Select File | New from the administration tool and enter a name, connection type, and location for your dictionary.
- Create a view and associate a view parent.
- Choose tables and fields to include in the view.
- Group related tables in the view.
- Create aliases for tables and fields.
- Create headings to appear on the report surface.
- In ReportSmith, choose Tools | Data Dictionaries and establish a dictionary in ReportSmith.

The items listed above are described in detail in the following sections.

The data dictionary workplace

The data dictionary workplace is shown below. Press a button at the top of the dialog box that corresponds to the action you'd like to define: Views, Tables, Groups, Aliasing, or Headings. The dialog box below shows settings for the Tables dialog box.



Since directory information is stored in the data dictionary for local connections, make sure the tables you add to the data dictionary are in the same place as they will be when you use them in your production environment.

Creating a view

A view controls what the end user sees. For example, the SALES dictionary can contain two views, "East Coast" and "West Coast," each containing regional data. A dictionary can contain numerous views, and you can grant access to more than one view per department or group.

When creating a view, use the Views dialog box to perform these tasks:

- Type a name for your view.
- Choose a view parent.

Choosing a view parent

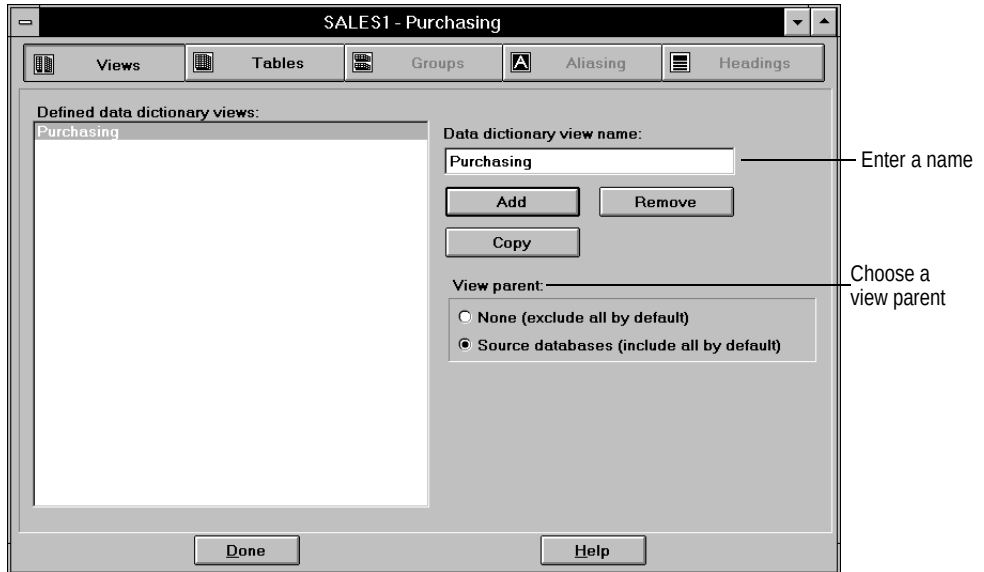
When creating a view, you need to choose a view parent. You have two options:

- **None:** If you do not want to display available tables in the Tables dialog box, choose this option. You will need to explicitly choose tables and fields in this view to display

in ReportSmith. This is less efficient in terms of data dictionary space usage than Source Databases described below.

- **Source Databases:** When you select Source Databases as the view parent, you get, by default, all the tables and fields that are normally part of a connection. This option is the most efficient in terms of data dictionary database usage, since entries are not made in the data dictionary for each table or field when you want a table or field to be visible in ReportSmith. All tables and fields are visible in ReportSmith by default, including new tables and fields that are added at a later time. Records only get entered into the data dictionary when tables or fields are excluded, aliased, or field headings defined.

The illustration below shows the Views dialog box.



Options in the Views dialog box are described in the following table:

Option	Description
Data Dictionary View Name	The name of your view.
Add	Add a view to the Defined Views list.
Remove	Delete a view from the Defined Views list.
Copy	Copies the currently selected view to another view. A new name is prompted for when the command is executed. The new view can be modified at will without changing the original view. If the view being copied is large, this command could take some time.
None (exclude all by default)	Only tables and fields explicitly included in the data dictionary will be displayed in ReportSmith.
Source databases (include all by default)	All the tables and fields that are normally part of a connection are included by default.

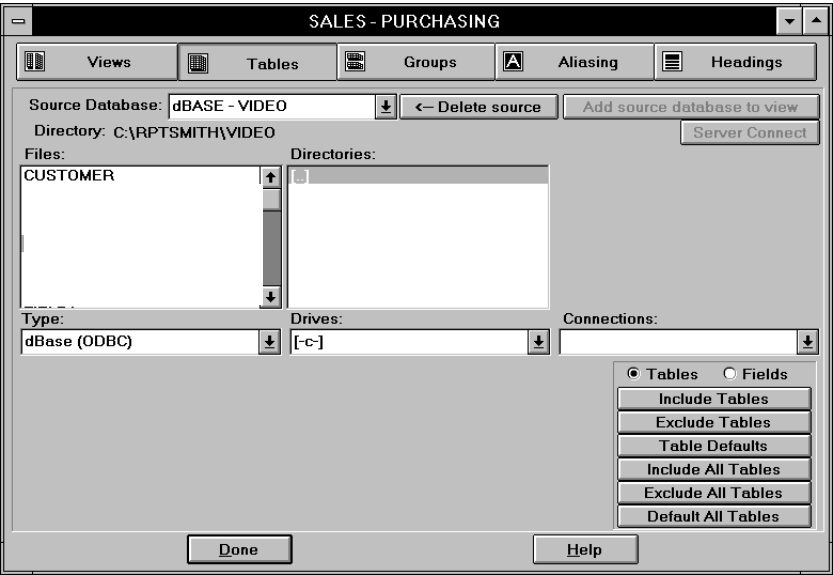
Adding tables and fields

After creating a view, choose the Tables button to specify tables and fields to include in or exclude from your data dictionary. To enable the Tables button, you must first select a View.

If you've chosen the Data Source option in the Views dialog box, establish a connection in the normal manner, and press the Add Data Source to View button. A dialog box appears requesting a name for the data source. The name you type for the data source then becomes part of the data dictionary and is associated with the view, and all tables and fields normally shown when that connection is established are available in ReportSmith.

You can add multiple data sources to a view. Each data source you add appears in the Data Sources combo box. Once you add a data source to a view, tables and fields for that data source appear in black in the list boxes.

The Tables dialog box with a data source defined is shown below:



Dialog boxes contain extended selection list boxes so you can drag the mouse to select more than one item in a list. To select separate items in a list, use **Ctrl+click**. When scrolling through a list, associated list boxes scroll in tandem with you.

Options in the Tables dialog box using the Data Source option are described in the following table:

Option	Description
Source Database	A name for the data source associated with this view.
Delete Source	Select the data source and press Delete Source to remove it.
Add Source Database to View	Press this button to add a data source to the view.

Option	Description
Server Connect	Remote connections require that you click on server connect after selecting the "Type."
Files	List of tables or fields from which you can choose. The contents of this list varies depending on the option selected: Tables or Fields.
Directories	Directory location of table.
Type	Type of database.
Drives	Use this list box to navigate to a drive.
Connections	Choose a named connection from the list to associate it with this view.
Tables	Choose this option to display a list of tables in the dialog box.
Fields	Choose this option to display a list of fields in the dialog box.
Include	Press this button to include the selected tables in this view.
Exclude	Press this button to exclude the selected tables from this view.
Include All	Include all tables or fields in this view.
Exclude All	Exclude all tables or fields from this view.
Table Defaults	Press this button to set the state of the selected tables as they were when the data source was added to the view.
Default All Tables	Press this button to set the state of all listed tables as they were when the data source was added to the view.



Right mouse button menus are generally available over the list boxes, and are a convenient alternative to the push buttons.

The Tables dialog box that appears using the None option is shown below:



Options in the dialog box are the same as those described in the previous table and include the following options:

Option	Description
Add All Tables to View	This causes all the tables in the Files or Tables list box to be added to the view. It also adds the fields in the tables to the view. Added tables and fields are displayed in black. If you just want to add the tables but not the fields to the view, select Include All Tables.
Table Owners	Displays the list of table owners for this connection.
Add Selected Tables to View	This causes the selected tables in the Files or Tables list box to be added to the view. It also adds the fields in the tables to the view. Added fields are displayed in black. If you just want to add the tables but not the fields to the view, select Include Tables.
Servers	This is a list of the servers found in the ReportSmith connection file. When making a connection to a remote server, selecting the Type, then the Server, and then clicking on the Server Connect button causes a connection to be made. It is more efficient, however, to set up a named connection with all this information. The named connection will then be listed in the Connections combo box.
Databases	This is a list of the available databases for this connection.

Creating groups

Use the Groups to group related tables together. This filters what appears in the Table Open dialog box in ReportSmith. The Groups dialog box is shown below:



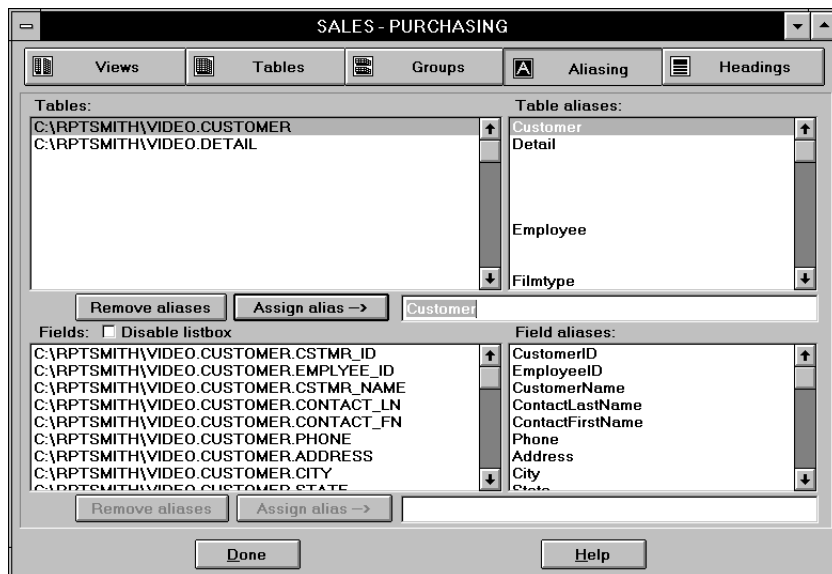
Options that appear in the Groups dialog box are described in the following table:

Option	Description
Group	Enter a name for the group into this text entry box.
Groups List	Groups you create appear in this list.
Add	Enter a name for the group into the text entry box and choose Add to place it into the Groups list.
Remove	Select a group to delete and choose Remove.
Include	Select tables by which to group and choose Include to include them in a group.
Exclude	Select one or more tables and choose Exclude to exclude it from a group.
Include All	Choose Include All to include all available tables in a group.
Exclude All	Choose Exclude All to exclude all tables from a group.

Assigning aliases

You can change the field or table name to display a name that pertains to a specific view. An alias is another name for a field or table, usually one that is easily recognizable by the department using the view. For example, the Purchasing Department might use the Vendors table aliased as “Sellers,” while the Production Department might use the same with the alias “Suppliers.”

To assign aliases, choose the Aliasing button to use the Aliasing dialog box, shown below.



Dialog boxes contain extended selection list boxes so you can drag the mouse to select more than one item in a list. To select separate items in a list, *Ctrl*+click. When scrolling through a list, associated list boxes scroll in tandem with you.

Options that appear in the Tables dialog box are described in the following table:

Option	Description
Tables	Select a table to which you want to assign an alias.
Table Aliases	Table aliases appear in this list box beside their corresponding tables.
Fields	Select a field from this list to which you want to assign an alias.
Disable List box	Select this check box to disable the loading of fields in the Fields list box. This improves performance of this administrative tool when defining table aliases.
Field Aliases	Field aliases appear in this list box beside their corresponding fields.
Remove Aliases	Select a table or field and press this button to remove its alias.
Assign Aliases	After you enter a table or field alias, press Assign Aliases to add it to the Aliases list or press <i>Enter</i> .

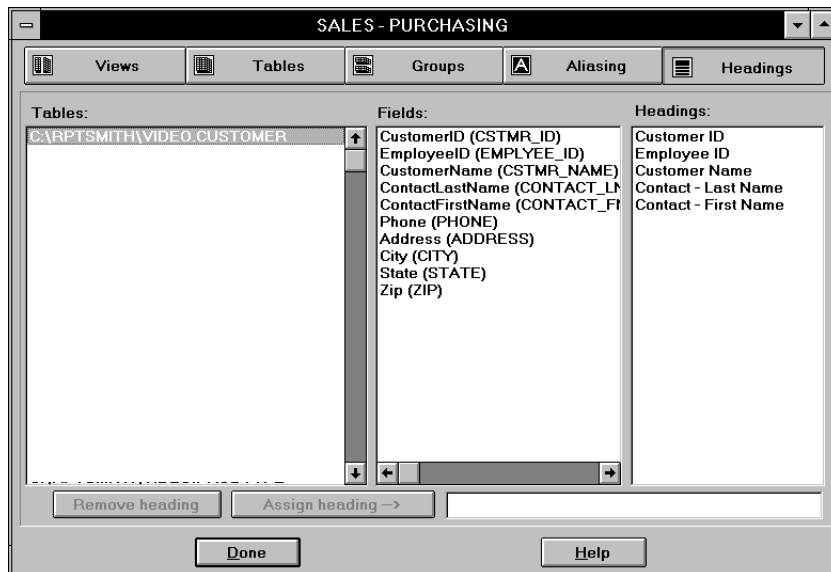


This dialog is designed with the touch typist in mind. When a single table or field is selected, the input focus is moved to the alias edit window. Then, after the alias is defined, press *Enter* to assign it to the field and to select the next item in the list. Also, if you have a long list of tables or fields, when you get near the bottom of the list, the list boxes start scrolling automatically.

Creating headings

A heading consists of text that appears at the top of a column as the column label. Choose the Headings button to create a heading. Headings appear in place of aliases or field names. You may use spaces.

For example, suppose you have a field, `CSTMTR_ID`, that you would like to appear on the report surface as “Customer ID.” You would assign this heading in the Headings dialog box, shown below.



Options in the Headings dialog box are described in the following table:

Option	Description
Tables	Select a table which contains fields to which you want to assign headings.
Fields	A list of fields and/or their aliases associated with the selected table appears in this list.
Heading	The heading assigned to a field appears beside it in this list.
Remove Heading	Select a heading and press Remove Heading to delete it from the list.
Assign Heading	Select a field and type a heading for it into the text entry box. Choose Assign Heading to add it to the Headings list, or just press <i>Enter</i> .

Copying a view

The amount of work involved in setting up a view can be considerable. It is possible that every table and field will need to be renamed and redefined. In addition, when setting up additional users, it is likely that a view will vary only slightly from an existing view. For this reason, the data dictionary supports the ability to copy views.

To copy a view,

- 1 Select a view in the Views dialog box.
- 2 Choose the Copy button. You will be prompted to enter a view name to which you'd like to copy the view. Enter the name of the new view and choose OK. The new view appears in the list box containing the view names. You can modify the new view without changing the original view.

Using a data dictionary from ReportSmith

To enable the use of a data dictionary in ReportSmith you must do the following things, all of which define the contents of the RS_DD.INI file that resides in your Windows directory:

- Add an entry in the RS_DD.INI file for each dictionary you wish to enable for a user. This is done automatically by the administrative tool when you create a dictionary. It can also be done from ReportSmith using the Tools | Data Dictionaries dialog box and choosing Add.
- Set the view for that dictionary. This puts an entry in the RS_DD.INI file for that dictionary. To set a view, select the Set View button from the Tools | Data Dictionaries dialog box.
- Associate that dictionary with various connections. This puts entries in the [Connections] section of the RS_DD.INI file. To associate a dictionary with a connection, select the Associate button from the Tools | Data Dictionaries dialog box. Whenever the user chooses the connection within ReportSmith, the specified data dictionary and view are automatically used.

After you have enabled the use of your dictionary by following the above steps, you can set up additional users by distributing the RS_DD.INI file to each user, or use the

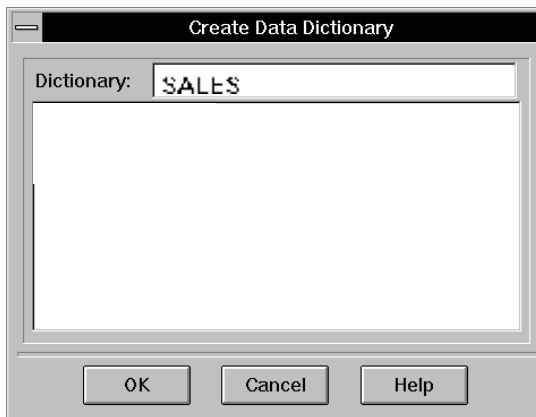
Creating a sample data dictionary

This tutorial shows you how to create a data dictionary using the sample dBASE tables in the VIDEO directory, unless you placed them elsewhere during installation. In this exercise you will create a data dictionary called “Sales” that contains a view for the Purchasing Department.

Note Make sure you have installed the sample tables.



- Double-click on the ReportSmith Data Dictionary icon to open the Data Dictionary. In the Startup dialog, choose “I want to make a new data dictionary” and press OK. Or, choose File | New Dictionary.
- Type SALES into the Create Data Dictionary dialog and press OK. This is the name of your data dictionary.

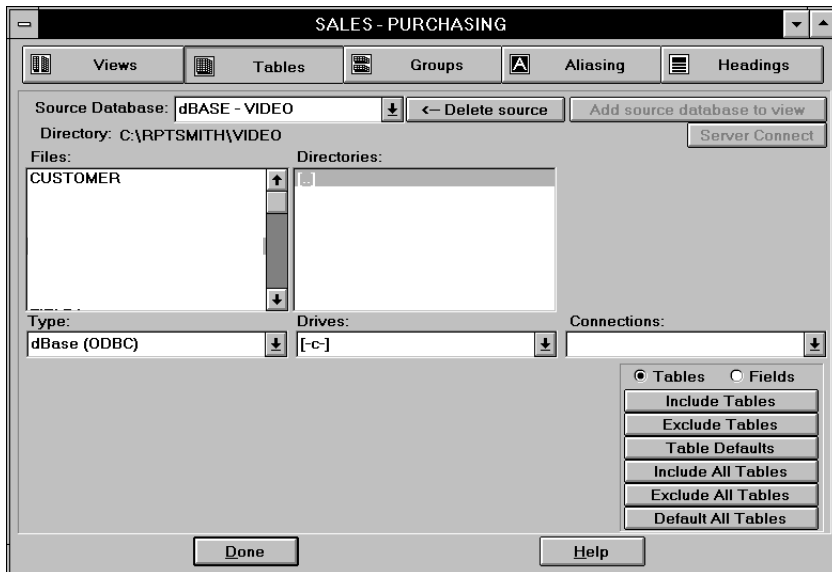


- In the Data Dictionary Location dialog box, choose the type and navigate to the location in which you want to place the data dictionary. This defines the location of the data dictionary tables. Choose OK.
- In the Views dialog box, type “Purchasing” in the text entry box. This is the name of your view.
- While you are still in the Views dialog box, choose the Source Databases option.

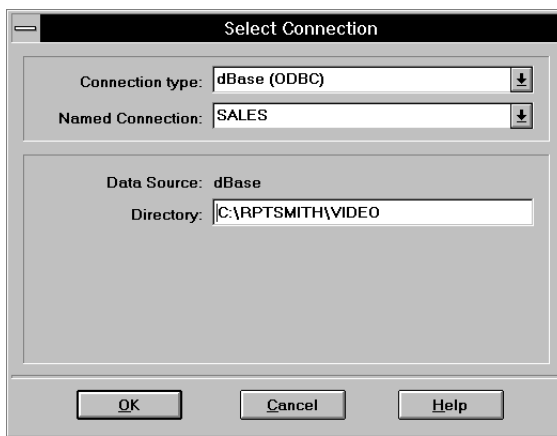
- Press Add to place the view name into the Defined Views list box.



- In the Tables dialog box, press the Add Source Database to View button and type a name for the data source into the Database Source Name box. This is a connection that you have declared to be a part of the data dictionary. The name can reflect the connection and directory which the data source points to. Type "dBASE - VIDEO" into the text entry box and press OK to return to the Tables dialog box.



- Select the Tables or Fields option in the lower right portion of the dialog box. Choose tables and fields to display in your view by selecting tables or fields, right-clicking the mouse, and choosing an option from the pop-up menu, or by pressing the Include button.
- In the Groups dialog box, type a name that pertains to your group into the text entry box, “Purchasing,” and press Add. Choose one or more tables from the Tables list box to include in the group, as shown above. Choose Include Tables.
- In the Aliasing dialog box you can create table and field aliases that are easily recognizable to a particular department. Select a table or field and type an alias (without spaces) into the text entry box, as shown in the following illustration. An alias appears in ReportSmith in place of the field or table name. If you do not enter an alias, the table or field name appears. To assign an alias, select a field or table, type an alias into the text entry box and press *Enter* or choose the Assign Alias button.
- Use the Headings dialog box to create headings that appear on the report surface. You may use spaces in heading names, as shown in the following illustration. If you do not enter a heading for a field, the alias or field name appears in place of the column label on your report.
- After you have created the dictionary, exit the Data Dictionary and open ReportSmith. You can not open ReportSmith and a Data Dictionary at the same time. In ReportSmith, choose Tools | Data Dictionaries to see the Data Dictionaries dialog box.
- Select the Data Dictionary you created (“SALES-Purchasing”) and press the Associate button. Associating a data dictionary means that whenever the end user uses a certain connection, for example, dBASE, ReportSmith uses the information specified in the Data Dictionary.
- Select a connection to which you want to associate the Data Dictionary. In this example we selected the dBASE (ODBC) connection named SALES. It points to the C:\RPTSMITH\VIDEO directory.



- After you have chosen a connection, press OK. Establish the dictionary in ReportSmith using the Tools | Data Dictionaries command.

You are now set up to use the SALES dictionary each time you choose the SALES connection in ReportSmith.



Using ReportSmith with other applications

You can use ReportSmith with other Windows applications, combining your work from several products into one application. ReportSmith supports this function through Dynamic Data Exchange (DDE) and Object Linking and Embedding (OLE). DDE lets you call ReportSmith from another application, and vice versa. OLE lets you include objects from other applications in your ReportSmith reports.

This appendix describes both methods (OLE and DDE), and presents examples for each, including the following topics:

Using DDE

- Initiating DDE conversations
- Starting ReportSmith from another application
- Sending menu commands to ReportSmith
- DDE synchronization

Using OLE 2.0

- Dragging information between applications

Using DDE

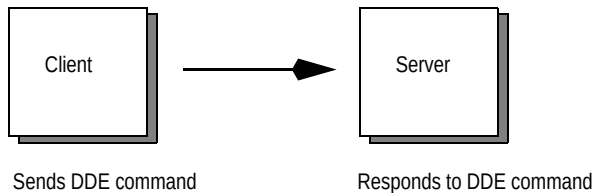
Dynamic Data Exchange (DDE) allows two Windows applications to communicate by sending and receiving data and commands.

ReportSmith can be either a DDE client or a DDE server. A *client* application makes requests of another Windows application. In turn, a *server* application responds to requests from another application. Requests can be commands or requests for specific data.

With ReportSmith, you can execute menu commands by Dynamic Data Exchange (DDE) command from another Microsoft Windows application. DDE is a protocol for Microsoft Windows that enables two applications to talk to each other. For example, using DDE, an application can tell ReportSmith to open and print a report, leaving ReportSmith minimized. This allows another application to present its user interface for printing previously built reports.

Initiating DDE conversations

Two applications using DDE to pass commands are having a DDE conversation. DDE establishes a channel between the two applications to transmit information. As shown in the illustration below, the DDE client is the application that initiates the conversation. The DDE server is the application that responds to the DDE client. ReportSmith can function as a client or a server.



To begin a DDE conversation, your code must specify:

- The name of the server to talk to (the application name.)
- The topic of the conversation (the *system* or *command* topic.)
- The items to be performed.

When ReportSmith receives a request for a conversation about a topic, it opens a channel. Once you establish a DDE conversation, you can't change topics or applications. To begin a conversation with a different server or with the same server about a different topic, you must start a new conversation on a different channel.

After a conversation has begun, the two applications exchange messages concerning particular items. An item is usually a reference to a piece of data in the topic. Items vary from topic to topic; each server recognizes different topics. ReportSmith supports the System topic for passing menu commands. It also supports the Command topic for passing ReportBasic commands.

Applications

Every Windows-based application that can participate in DDE conversations has a unique application (or app) name. The table below shows the app names for some common Windows applications:

Application	App name
ReportSmith	ReportSmith
Microsoft Access	MSAccess
Microsoft Excel	Excel

Application	App name
Program Manager	Progman
Word for Windows	WinWord
ReportSmith Runtime	RS_Runtime

Topics

A *topic* defines the subject of the DDE conversation and represents some unit of data meaningful to the DDE server application. ReportSmith supports the System topic and the Command topic, as well as user-defined topics.

Items

An *item* is a reference to a piece of data or, in ReportSmith's case, a command to execute. For example, an item could be a menu command such as File | Open or a ReportBasic command such as LoadReport.

Note Strings used for the application name, topic, or item in a DDE conversation aren't case-sensitive.

Calling ReportSmith

To start ReportSmith from another application:

- 1 First you must start ReportSmith from within another application. For example, the following Delphi code starts ReportSmith(minimized and without focus):

```
x = Shell("c:\rptsmith\rptsmith.exe", 6)
```

- 2 You can now execute DDE conversations.

Note Examples of this are shown in the following sections.

Calling another application

ReportSmith can act as a DDE client. Other Windows applications may be called from a ReportSmith macro using the Report Basic language. The commands used are listed below.

Report Basic Command	Description
DdeExecute	Send a DDE execute command to a DDE server application.
DdePoke	Poke data to a DDE server application.
DdeRequest	Get data from a DDE server application.

Refer to the online Help for details on the commands used to call other applications.

Sending menu commands

The syntax for a DDE command in ReportSmith consists of the main menu name, a vertical bar, the menu item, and, optionally, a list of parameters enclosed within parenthesis.

The illustration below identifies the parts of a DDE command:

view|zoom ()

Menu name Command name

You can also specify the option you want to use in the DDE command. For example, if you want your application to magnify the object in its window up to 150%, you enter the DDE command as follows:

`view|zoom(150)`

Following are examples of some other commands:

- File | Open(BASIC.RPT)
- File | Close
- File | Print
- View | Zoom(150)

You can send a command to ReportSmith using the following Delphi code examples.

DDE synchronization

The following code is an example of utilizing DDE synchronization to set up a conversation between Delphi and ReportSmith to tell ReportSmith to display a message.

```
unit DDEtest;
interface
uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, DDEMan;
type
  TForm1 = class(TForm)
    DDEClientConv1: TDDEClientConv;
    Button1: TButton;
    Label1: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
```

```

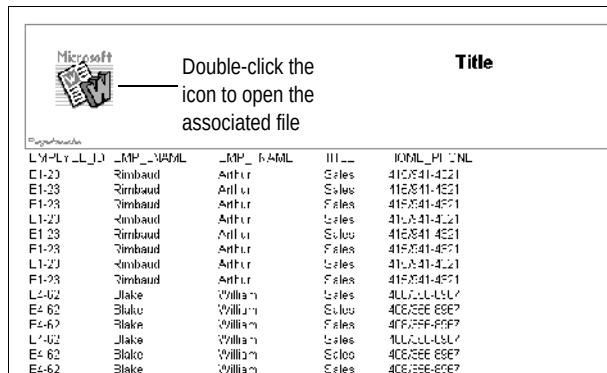
implementation
{$R *.DFM}
procedure TForm1.Button1Click(Sender: TObject);
Var
    DDECommand      :PChar;
    Success          :Boolean;
    RS_Window_Handle :HWND;
    LoadReportDone   :Boolean;
begin
    { * Set up DDE Client conversation *}
    DDEClientConv1.DDEService := 'RS_RunTime';
    DDEClientConv1.DDETopic   := 'Command';
    DDEClientConv1.ServiceApplication := 'RS_RUN';
    { * allocate enough space for null-terminated DDE Command string * }
    DDECommand := StrAlloc(256);
    { * Copy command from Pascal string into DDE Command buffer *}
    StrPCopy(DDECommand, 'MsgBox "Hello"'); }
    { * Execute DDE command to cause ReportSmith to bring up a message box *}
    Success := DDEClientConv1.ExecuteMacro(DDECommand, TRUE);
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
    { * change caption of the form and button *}
    Caption := 'Delphi - ReportSmith DDE Example';
    Button1.Caption := 'Execute DDE command';
end;
end

```

Using OLE 2.0

You can include information or objects created in other Windows applications in your report with ReportSmith's linking and embedding (OLE) features. For example, a ReportSmith report can contain a Lotus worksheet, or a Microsoft Word document, or an Excel spreadsheet; these are all examples of *objects*. To access a linked document within the application it was created in, double-click an icon representing the object in your report. For example, a report with an embedded Microsoft Word document

displays the Microsoft Word icon on the report. When you double-click on the icon, the Word document appears.



				Title
101-123-123	MP_123	MP_123	MP_123	MP_123
E1-23	Rimbaud	Arthur	Files	415841-4521
E1-23	Rimbaud	Arthur	Files	415841-4521
F1-23	Rimbaud	Arthur	Files	415841-4521
L1-23	Rimbaud	Arthur	Files	415841-4521
E1-23	Rimbaud	Arthur	Files	415841-4521
F1-23	Rimbaud	Arthur	Files	415841-4521
L1-23	Rimbaud	Arthur	Files	415841-4521
F1-23	Rimbaud	Arthur	Files	415841-4521
E4-62	Blake	William	Files	415841-4521
E4-62	Blake	William	Files	415841-4521
F4-62	Blake	William	Files	415841-4521
L4-62	Blake	William	Files	415841-4521
E4-62	Blake	William	Files	415841-4521
E4-62	Blake	William	Files	415841-4521
E4-62	Blake	William	Files	415841-4521

The main difference between linking and embedding an object is where you store the data. When you *embed* an object in a report you are inserting information into that report. Embedded objects become a part of the report itself. When you *link* an object to your report, the report stores just the location of the object and displays a representation of the linked data such as an icon. Linked objects are stored in a source file and retain a connection to the original Windows application.

Use embedding to include information that becomes a part of the report and is always available, even if the original source file is absent or moved. You can also use embedding to include a file that may not always be available, such as a file residing on a network.

Use linking to include data in your report that is maintained in a separate file and to save space. The ReportSmith report will reflect any changes made to the source file and always displays the last-saved version. Also, you can use linking to include a very large file such as a video or sound object in your report.

Embedding objects in a report

To embed an existing object from another application:

- 1 Create and save the object in its application.
- 2 Select the object, and use Edit | Copy to copy it to the Clipboard.
- 3 Return to ReportSmith by minimizing the application or clicking outside the application window.
- 4 Select Edit | Paste.
- 5 With the mouse, position the Object cursor within the header or footer where you want to place the object. Left-click the mouse. ReportSmith places the object where you clicked in the report.
- 6 Double-click the header or footer if the object exceeds its boundaries. A box with sizing handles surrounds the header or footer. Click and drag one of the corner

To create an object and embed it in a report:

- ReportSmith can accept and embed objects from any applications you have installed on your workstation that fully support OLE. In this example, we'll insert a Microsoft Graph into the page header of our report.

- The Object cursor appears.

- 5** Create the object.

- The diagram shows a 3D storage device with a title 'Title' and a legend for 100, 102, 103, 104, 105. The device is a rectangular block with a circular base. The legend indicates that the device is composed of five parts: 100 (black), 102 (white), 103 (gray), 104 (light gray), and 105 (dark gray). The device is shown in a perspective view, with the top surface being white (102) and the base being black (100). The sides are composed of the other parts.

Editing an embedded object

To edit an embedded object:

- Appendix A, Using ReportSmith with other applications 217

2 Select Edit | Object.

Note The application name appears on the Edit menu, along with the Object command. For example, Edit Microsoft Graph Object appears as the command name if the object you select was created in Microsoft Graph. ReportSmith opens the application and displays the selected object you want to edit.

3 Edit the object.

4 Select the application's Update command from the File menu when you finish editing and close the application.

Linking objects to a report

To create a link to an object from another application:

1 Create and save the object in its application.

2 Select the object and copy it to the Clipboard with the Copy command on the Edit menu.

3 Return to the report by minimizing the application or clicking outside the application window.

Note Don't close the application from which you're copying the object until you've selected the Paste Link command. If you close it before, Paste Link command won't be available on the Edit menu.

1 Select the Paste Link command on the Edit menu.

2 With the mouse, position the Object cursor within the header or footer where you want to place the object. Click the mouse to place the object in the report and create a link to the original file in the application. The next time you open your report, ReportSmith asks if you want to reestablish the links for the report. If you do, answer yes. ReportSmith updates the object if there have been changes.

To edit a link:

1 Select Edit | Edit Links to open the Links dialog box.

2 Click the link you want to edit in the Links list box. Choose the Update option you want to use for the link you selected in the Links list box.

- Click Manual if you don't want ReportSmith to automatically update the object when you open your report.
- Click Automatic if you want ReportSmith to automatically update the object when you open your report.

3 Press the Update Now button if you have Manual set as the Update option and you want to update the object or if the report the object is in was open when changes were made to the original file. ReportSmith updates the object.

To cancel a link:

- 1 From the Edit menu choose Edit Links. The Links dialog box appears. The Edit Links command is available only if you've created links. If you haven't, the command is dimmed on the Edit menu.
- 2 Click the link you want to cancel in the Links list box.
- 3 Press the Cancel Link button. ReportSmith cancels the link and the Links options for that link are dimmed.

To change the location of a link:

- 1 Select the Edit Links command from the Edit menu. The Links dialog box appears. The Edit Links command is available only if you've created links. If you haven't, the command is dimmed on the Edit menu.
- 2 Click the link you moved to a new location.
- 3 Press the Change Link button. The Change Link dialog box appears.
- 4 Type in the new path for the linked object. ReportSmith changes the link so it can continue to update the object if any changes are made.

Dragging data between applications

With ReportSmith, you can drag and drop data or objects from one application to another. However, in order to use this feature your other application must support OLE Version 2.0.

To copy data from one application to another:

- 1 Open ReportSmith and display your report.
- 2 Open the other application and display it on screen along with ReportSmith. To do this, use the Windows | Tile command.

Note It is important that both applications be displayed on screen at the same time.

- 1 Select the object you want to copy.
- 2 Drag it across into the other application and drop it into place. ReportSmith copies the object from the first application and copies it into the second application.

To copy an object to the cursor position in an application:

- 1 Position the cursor where you want to place the object.
- 2 In the other application, select the object you want to place.
- 3 Holding the *Shift* key, drag and drop the object into the receiving application.

The object appears in the receiving application.

To move an object from one application to another:

- 1 Select the object you want to move.
- 2 Holding the *Ctrl* key, drag the object into the receiving application. ReportSmith cuts the object, removing it from the first application and pasting it into the second.

B

Macro reference

This reference chapter consists of two sections:

- “Macro object reference” lists and describes the events to which you can link a macro, such as Application Start Up. It also describes basic macro conventions and data types for variables.
- “Command reference” lists and describes each ReportBasic and DataSet Control command and SoftBridge Basic commands you can use to build ReportSmith macros.

Macro object reference

You can link macros to events. An event is some action in ReportSmith that can trigger the execution of a macro. This section lists and describe the available events. They are categorized according to the three object types to which you can link a macro: the application object for global macros, the report object, and the data field object for report macros.

For a conceptual understanding of macros and how they link to objects and events, see “Introduction to macros” on page 159.

Note You link macros to events and objects using the Macro Links dialog box.

Basic conventions

When evaluating expressions, ReportBasic gives precedence to operators. To override the default precedence you can use parentheses. The following table describes each

operator. The operators are listed in order of precedence; the first operator is the first to be evaluated.

Expression	Description
()	Array element.
.	Record member—the left operand must be a record variable, and the right operand must be the name of a field.
Imp	Implication—operands can be Integer or Long. The operation is performed bitwise. (A Imp B) is the same as ((Not A) OR B).
Eqv	Equivalence—operands can be Integer or Long. The operation is performed bitwise. (A Eqv B) is the same as (Not (A X or B)).
Xor	Exclusive Or—operands can be Integer or Long. The operation is performed bitwise.
Or	Inclusive Or—operands can be Integer or Long. The operation is performed bitwise.
And	And—operands can be Integer or Long. The operation is performed bitwise.
Not	Unary Not—operand can be Integer or Long. The operation is performed bitwise (one's complement).
>, <, =, <=, >=, <>	sequence used by the language specified by the user using the Windows Control Panel. The result is 0 for FALSE and -1 for TRUE.
-, +	Numeric addition and subtraction. The + operator is also used for string concatenation.
Mod	Modulus or Remainder. The operands can be Integer or Long.
\	Integer division. The operands can be Integer or Long.
*, /	Numeric multiplication or division. For division, the result is a Double.
-, +	Unary minus and plus.
^	Exponentiation.

Variable data types

A variable declared inside of a procedure has scope local to that procedure. A variable declared outside of a procedure has scope local to the module. It is permissible for a procedure to declare a variable with a name that matches a module variable. When this happens, the module variable is not accessible by the procedure.

The Shared keyword is included for backward compatibility with older versions of Basic. It is not allowed in Dim statements inside of a procedure. It has no effect.

Basic allows a variable to be automatically declared, without the use of a Dim statement. If a variable is first used with a type character as a suffix to its name, the variable is automatically declared to be a local variable of the specified type. If no type character is specified, the variable is automatically declared to be a local variable of type Double. It is considered good programming practice to declare all variables and not make use of this feature. It is also recommended that you place all procedure-level Dim statements at the beginning of the procedure.

Numbers

The four numeric types are:

Type	Range
Integer	From -32,768 to 32,767
Long	From -2,147,483,648 to 2,147,483,647
Single	From -3.402823e+38 to -1.401298e-45, 0.0, 1.401298e-45 to 3.402823466e+38
Double	From -1.797693134862315d+308 to -4.94065645841247d-308, 0.0, 2.2250738585072014d-308 to 1.797693134862315d+308

Numeric values are always signed.

Basic has no true boolean variables. Basic considers 0 to be FALSE and any other numeric value to be TRUE. Only numeric values can be used as booleans. Comparison operator expressions always return 0 for FALSE and -1 for TRUE.

Integer constants can be expressed in decimal, octal, or hexadecimal notation. Decimal constants are expressed by simply using the decimal representation. To represent an octal value, precede the constant with "&O" or "&o" (e.g., &o177). To represent a hexadecimal value, precede the constant with "&H" or "&h" (e.g., &H8001).

Example `NumberVar = 5`

Strings

Basic strings can be either fixed or dynamic. Fixed strings have a length specified when they are defined. The length cannot be changed. Fixed strings cannot be 0 length. Dynamic strings have no specified length. Any string can vary in length from 0 to 32,767 characters. There are no restrictions on the characters which can be included in a string. For example, the character whose ANSI value is 0 can be embedded in strings. Local string variable names require a \$ at the end of the name of the variable.

Example `MyStringVar$ = "A text string or field"`

Records

Record variables are declared by using an As clause and a typeName which has previously been defined using the Type statement.

The syntax for Records looks like this:

```
Dim variableName As typeName
```

Records are made up of a collection of data elements called fields. These fields can be of any numeric, string, or previously-defined record type.

For details on accessing fields within a record see "Type statement" on page 354.

Arrays

Arrays are created by specifying one or more subscripts at declaration or Redim time. Subscripts specifies the beginning and ending index for each dimension. If only an

ending index is specified, the properties of the beginning index are based on the Option Base setting. Array elements are referenced by enclosing the proper number of index values in parentheses after the array name, for example, `arrayname(i,j,k)`.

The syntax for Arrays can be either of the following:

```
Dim variable( [ subscriptRange, ... ] ) As typeName
Dim variable_with_suffix([ subscriptRange, ... ])
```

where `subscriptRange` is of the format:

```
[ startSubscript To ] endSubscript
```

For more information, see “Dim statement” on page 307.

Example

```
Dim DS as DataSet 'Declare a DataSet
Dim A(5,2) 'Declare an array
Dim A as Integer 'Declare a variable
```

Application Data Types (ADTs)

Application Data Types are specific to each application that embeds the macro language. ADT variables have the appearance of standard Basic records. The main difference is that they can be dynamic; creating, modifying or querying the ADT or its elements causes application-specific actions to occur. ADT variables and arrays are declared just like any other variable, using the Dim or Global statements.

Dialog box records

Dialog box records look like any other user-defined data type. Elements are referenced using the same `recname.elementname` syntax. The difference is that each element is tied to an element of a dialog box. Some dialog boxes are defined by the application, others by the user.

For more information, see “Begin Dialog ... End Dialog statement” on page 293.

Conversions

Basic automatically converts data between any two numeric types. When converting from a larger type to a smaller type (for example, Long to Integer), a runtime numeric overflow can occur. This indicates that the number of the larger type is too large for the target data type. Loss of precision is not a runtime error (for example, when converting from Double to Single, or from either float type to either integer type.)

Basic also automatically converts between fixed strings and dynamic strings. When converting a fixed string to dynamic, a dynamic string which has the same length and contents as the fixed string is created. When converting from a dynamic string to a fixed string, some adjustment can be required. If the dynamic string is shorter than the fixed string, the resulting fixed string is extended with spaces. If the dynamic string is longer than the fixed string, the resulting fixed string is a truncated version of the dynamic string. No runtime errors are caused by string conversions.

No other implicit conversions are supported. In particular, Basic does not automatically convert between numeric and string data. Use the functions Val and Str\$ for such conversions.

Trappable errors

This table lists the runtime errors which the macro language returns.

These errors can be trapped by the “On Error statement” on page 337. The “Err function” on page 313 can be used to query the error code, and the “Error\$ function” on page 314 can be used to query the error text.

Error code	Error text	Error code	Error text
5	Illegal function call	62	Input past end of file
6	Overflow	63	Bad record number
7	Out of memory	64	Bad file name
9	Subscript out of range	68	Device unavailable
10	Duplicate definition	71	Disk not ready
11	Division by zero	74	Can't rename with different drive
14	Out of string space	75	Path/File access error
19	No resume	76	Path not found
20	Resume without error	102	Command failed
28	Out of stack space	901	Input buffer would be larger than 64K
35	Sub or function not defined	902	Operating system error
48	Error in loading DLL	903	External procedure not found
52	Bad file name or number	904	Global variable type mismatch
53	File not found	905	User-defined type mismatch
54	Bad file mode	906	External procedure interface mismatch
55	File already open	907	Push-button required
58	File already exists	908	Module has no MAIN
61	Disk full	910	Dialog box not declared

Application events

You can link a macro to an action, such as the keystroke event, *Ctrl+R*, or to an event that coincides with report loading. You cannot link application macros to report events. For information on how to link events to specific reports, refer to “Report Events” on page 230.

The following sections describe each application event and provide specific scenarios for real-world applications.

KeyStroke

When you link a macro to the KeyStroke event, it can be tied to any key or key combination on your keyboard. A macro linked to the KeyStroke event runs when you press the key or key combination you specify.

For example, suppose you want to link a macro to the keystroke, *Ctrl+R*, so that whenever you press *Ctrl+R*, ReportSmith loads a report that you work on often.

If the active report has a report macro linked to the same keystroke, then that report macro is executed first (see “KeyStroke” on page 226).

Before new report

A macro linked to this event runs after you choose File | New (or press the New button to create a new report), but before you choose the tables that you want to include in the report. This event lets you perform tasks, such as verifying that a user has permission to create reports.

For example, you can have a global macro display a dialog box that prompts for a password. If the password’s correct, you can let the user continue with the New operation. If it’s incorrect, you can cancel the New operation and display an informative message describing why the operation is canceled.

After new report

A macro linked to this event runs after you choose File | New to create a new report and after ReportSmith executes a query based on the tables you choose in this report. You might want to link to this event to change the default configuration for new reports.

For example, you can specify the display mode (draft or presentation), turn the report boundaries on or off, turn the grid on or off, and so on.

You can also link to this event to send DDE commands to another Windows application. For example, suppose you are using PowerBuilder to drive ReportSmith, and PowerBuilder’s role is to provide filename conventions. You can link a global macro to this event and send a DDE message to PowerBuilder informing it to save all new reports using the filenames it generates.

Application start up

A macro linked to this event runs immediately after you click the ReportSmith icon to open the application and after ReportSmith displays its splash screen. You can link a macro to this event to change the default ReportSmith environment or perform other tasks before you open ReportSmith.

For example, you can add new menu items, disable or remove existing menu items, load the reports you use on a daily basis, or execute a menu command, such as New or Open on the File menu.

You can also launch other Windows applications simultaneously, such as Visual Basic, Excel, and PowerBuilder.

Before executing SQL

This event occurs each time ReportSmith generates a SQL statement, before that statement executes. You can use it to allow a macro to obtain a copy of the ReportSmith SQL statement each time it changes.

Note You can retrieve the last SQL statement using the GetSQL function.

In rare cases, you might find it necessary to modify the ReportSmith SQL statement. To do this, you can use the macro command, SetSQL, to replace the string generated by ReportSmith.

Note Do not modify the query in the Before Executing SQL event. Instead, use the DataSet Control methods (such as SetUserSQL, AddTable, and IncludeFields). Changes made during the Before Executing SQL event are lost the next time you generate a SQL statement. Use the SetSQL command with caution. Use it to make small changes to the SQL statement to support servers which have unusual or non-standard SQL requirements.

Before report print

A macro linked to this event runs after you select Print from the File menu, but before the report is actually sent to the printer. You can link a macro to this event when you want to perform print-related tasks.

For example, you can have a macro display a dialog box that specifies the printer being used. Or you might want to display an interactive dialog box that lets the user enter printer parameters, such as margin specifications, number of copies, page orientation, paper size, and so on.

You can also link a macro to this event to warn users when a report is large, and then have the macro display a dialog box that gives them the opportunity to cancel the print.

Before report load

A macro linked to this event runs after you select Open from the File menu and after you select an .RPT file from the Open Report dialog box. However, it runs before ReportSmith actually opens and displays the corresponding report. You can use this event to determine if the report can actually open.

For example, suppose you want to insure that a user has only one .RPT file open at a given time. You can create a global macro and link it to this event so that it saves and closes all active reports before allowing any new report to be opened.

After report load

A macro linked to this event runs after you select a report using File | Open, and after ReportSmith actually opens and displays the selected report.

You can display all reports in draft mode by creating a macro that turns draft mode on when a report is opened.

Or suppose you want to keep track of all report files opened by a given user. A macro linked to this event can identify the title of each report and write its title and the time it was opened in an ASCII file.

Before report save

A macro linked to this event runs after you select Save (or Save As) from the File menu, but before ReportSmith actually saves the active report in an *.RPT* file. If this is your first time saving the report or if you're saving an existing report under a different name, the macro runs before the dialog box prompts you for a filename.

For example, by linking a macro to this event, you can prevent a user from overriding certain *.RPT* files that you don't want modified. Suppose you want to prevent users from modifying all reports created in the month of May. Simply have the macro display a message indicating that the report can't be modified whenever a user attempts to modify and save a May report, and then cancel the Save (or Save As) operation. In a similar way, you can use this event to verify sufficient disk space, and then display a warning message if sufficient disk space is unavailable.

After report save

A macro linked to this event runs after you select Save (or Save As) from the File menu and after ReportSmith saves the active report in an *.RPT* file. A macro linked to this event can for example, close reports immediately after ReportSmith saves them. In a similar way, you can use this event to automatically create backup copies of saved reports in a backup directory.

Before application close

A macro linked to this event runs after you select Exit from the File menu to close ReportSmith, but before ReportSmith actually closes. You can link a macro to this event for tasks such as preventing ReportSmith from closing under certain circumstances.

For example, you might not want ReportSmith to close if a driving application, such as PowerBuilder, is still open. In this case, you can have a macro cancel the close, and then display a message informing the user to close the driving application first.

You could also link a macro to this event to automatically save all open reports before ReportSmith closes, rather than have ReportSmith prompt the user to save each open report individually.

On SQL Error

Each time a SQL execution error occurs, the On SQL Error event is triggered. A macro linked to this event can execute the `LastError` function to retrieve error text (usually from an ODBC driver). It can attempt to fix the error by changing the report query from which it originated. If that macro executes a `ResumeEvent` command with an argument of 0, the `FixReport` dialog box will not appear and the SQL re-executes.

Note If the macro does not correct the error in FixReport, this event is called again. Use caution when using ResumeEvent 0 to avoid being caught in an endless loop.

New File Icon Click

This event is called when the user presses the New Report button on the toolbar. A macro linked to this event can call ResumeEvent command passing a 0 (FALSE) value which would cause the regular new report processing to be aborted. This might be done to replace the standard new report behavior with custom behavior, defined by a macro.

SQL Icon Click

This event is called when the user selects the SQL button on the toolbar. A macro linked to this event can call ResumeEvent command passing a 0 (FALSE) value, which would cause the regular SQL processing to be aborted. This might be done to replace the standard SQL viewing/editing behavior with custom behavior defined by a macro.

After Report Connects

This event is called after a report connects, before ReportSmith executes the report query. It is the application level version of the report event, "Before Report Open." This event is useful for macros that wish to modify a report's query before it is run. A dataset control object could be associated with a loading report with the .SetFromLoading command.

Before Report Close

This event is called before a report closes.

After Report Close

This event is called after a report closes.

Creation Events

Creation Events can be linked to two objects: Group Header and Group Footer. When you link a macro to the Header/Footer creation event, you must choose the grouping level of the header or footer to which you want to link.

A macro linked to these events can call the ResumeEvent command with an argument of 0 to suppress the creation of an individual group header or footer based on the data in the report.

Report Events

A report macro is linked to a specific report and becomes a part of that report. You link report macros to report events.

Note To link events to the ReportSmith application or to reports in general, see “Application events” on page 225.

The following section describes each report event and provides specific scenarios to show you how to use them in common reports.

KeyStroke

A macro linked to the KeyStroke event runs when you press the key or key combination you specify, and the report to which the macro is also linked is the active report.

Note If the active report has a report macro linked to the same keystroke, the report macro is executed first. If the report macro is intended to replace the application macro (linked to the same keystroke), the report macro should call the ResumeEvent command passing a value of zero (false). In effect, the ResumeEvent code tells ReportSmith it should not execute other processes normally tied to that event.

A KeyStroke event linked to a report macro overrides one that is linked to a global macro.

For more detailed information and for examples of how you can use the KeyStroke event, refer to “KeyStroke” on page 226.

Before report open

A macro linked to this event runs after you open a specific report, but *before* ReportSmith executes the query to run the report. You can use this event to automatically set report variables for the report’s selection criteria.

Report variables let you create dialog boxes that prompt users for query values before a report is run. If you create a macro that sets report variables automatically, ReportSmith knows not to display the dialog box.

If the macro linked to this event calls the set report variable command (refer to “SetRepVar” on page 260), it sets the report variables for the report being loaded, rather than the active report.

After report open

A macro linked to this event runs after you open the specific report that the macro is linked to, and after ReportSmith runs the query and displays the report. You can use this event to trigger an action immediately after the report opens.

For example, you can automatically send a specific report to the printer after it first opens. Or you can have the opening of a specific report trigger the loading of additional, related reports.

You can also use this event if you want a specific report to appear in a different display mode than all other reports in your company. For example, suppose all reports are set up to appear in presentation mode when you first open them, and you want a specific report to appear in draft mode instead. You can create a report macro and link it to this event so that only the specific report appears in draft mode every time you open it.

Before SQL Execution

This event is the same as the application level event, except that it applies only to the report to which the macro is linked.

Before print

A macro linked to this event runs after you select File | Print, but before ReportSmith actually prints the specific report to which the macro is also linked. Suppose you have a specific report that is particularly large and you don't want to tie up the department's printer. You can link a macro to this event to automatically send that report to another printer, which is not used as often.

You can also use this event to warn the users that the specific report is particularly large, and that printing it will be a time-consuming process. Then you can give them the opportunity to cancel the print operation.

Before report save

A macro linked to this event runs after you attempt to save a specific report (to which the macro is also linked), but before ReportSmith actually saves it in an *RPT* file. If this is your first time saving the report, the macro runs before the dialog box prompts you for a filename.

By linking a macro to this event, you can prevent a user from overriding a specific report that you don't want modified. Simply have the macro display a message indicating that the report can't be modified, and then cancel the save operation. In a similar way, you can use this event to send a DDE command to the e-mail program your company uses so that you can notify other users that a new version of a specific report is available.

Before report close

A macro linked to this event runs after you select File | Close to close a specific report, but before ReportSmith actually closes it. You can link a macro to this event to restore options set by a macro that was linked to the After Opening the Report event.

For example, suppose the macro that was run after you opened the report changed the display mode from Presentation to Draft. You can link another macro to this event to restore the display mode back to Presentation.

On SQL error

Each time a SQL execution error occurs, the On SQL Error event is triggered. A macro linked to this event can execute the `LastError` function to retrieve error text (usually from an ODBC driver). It can attempt to fix the error by changing the report query from which it originated. If that macro executes a `ResumeEvent` command with an argument of 0, the FixReport dialog box will not appear, and the SQL will re-execute.

This event applies only to the report to which the macro is linked.

Note IF the macro does not correct the error in FixReport, this event is called again. Use caution when using `ResumeEvent 0`, to avoid being caught in an endless loop.

Selecting a menu item

This event lets you link to specific *menu items*.

A macro linked to this event runs when you select the menu item to which the macro is also linked, but before the corresponding action of that menu item takes place. You can use the `ResumeEvent` command to determine whether or not the corresponding action is executed.

Suppose you have your own help file built for a specific report. And you want this file to appear (rather than ReportSmith's standard help file) whenever you select Index from the ReportSmith Help menu. You can link a macro to this event to replace the ReportSmith help file with the new help file for the specific report only.

Data field events

Currently, ReportSmith uses only one data field event called the Display event. A macro linked to this event runs when ReportSmith generates a value for the data field object to which the macro is linked.

The primary purpose of the Display event is to allow conditional formatting. To perform conditional formatting, it must be based on criteria to which values in report columns can correspond.

When creating a conditional formatting macro, you can use the `FieldFont` and `FieldText` macro commands. These commands let you tell the macro how to format the values that fulfill the specified criteria.

Header/Footer events

Macros can be linked to the Group Header or Group Footer objects. These objects currently have one event, the Creation event.

When you link a macro to the Header/Footer event, you must choose the grouping level of the header or footer to which you want to link. In the Items list box of the Macro Links dialog box, you can view a list of the groups in the active report. For a macro to execute each time you create a department group footer, select the Group Footer object, and the Department group, and press the Link button.

A macro linked to the Header/Footer event can call ResumeEvent, passing an argument of 0 (FALSE) to suppress the creation of an individual group header or footer based on the data in the report. The current record in the report is set to the first record of the group when the macro is called. The current record of the dataset is the first record of the specified group. For example, if you group by department and want to know what department header or footer is being called, execute Field\$("Dept") to retrieve the department name.

Command reference

You can use four types of commands when you create a macro: ReportBasic, DataSet Control commands, Report Control commands written for ReportSmith, and core Basic commands common to the Basic language.

Data types and type conversion

You can use variables in your macros to ease use and readability. Variables can be declared implicitly on first reference by using a type character; if no type character is present, the default type of Double is assumed. Alternatively, the type of a variable can be declared explicitly with the Dim statement. In either case, the variable can only contain data of the declared type.

Variables of user-defined type must be explicitly declared. The macro language supports standard Basic numeric, string, record and array data. The macro language also supports Dialog Box Records and Application Data Types (which are defined by the application).

Variables can be used globally or locally in macros. Global variables can be recognized and used by other macros. Local variables can be used only within the macro that they are declared.

Global variables are declared before the Sub statement.

Example

```
Global my_var as Number
Sub My_macro()
:
End Sub
```

Local variables are declared within the Sub statement.

Example

```
Sub My_macro()
Number_var = 55
String_var$ = 'MYNAME'
:
End Sub
```

ReportBasic command summary

The following table and subsections describe each ReportSmith command.

If a ReportBasic command can be used only as a function *or* a command the word function or command appears after it in this table. Otherwise, an item can be used as *both* a function and a command.

Command	Description
ActiveTitle\$() function	Gets window title of currently active report.
AddMenu	Allows you to add your own commands to the main menu.
CloseReport	Lets you close the active report.
CloseRS	Closes ReportSmith.
Connect	Opens a connection to a database server.
CreateDDEItem statement	Allows you to create a new Dynamic Data Exchange Item in ReportSmith's DDE server.
CreateDDETopic statement	Allows you to create a new Dynamic Data Exchange topic in Report Smith's DDE server.
Current	Returns the record number to which the data set is pointing.
CurrentPage function	Returns the page number that is displayed in the active report.
Date2Str	Allows you to convert from a Julian date code to a date string.
DateField	Gives you date values.
DDEExecute	Lets you send a DDE execute command to a DDE client application.
DDEPoke	Lets you perform a DDE Poke string to a DDE client application.
DDERequest	Gets data from a DDE Server application as a string.
DerivedField	Sets the value of a Derived Field.
DoEvents	Allows other Windows applications to process messages.
EnableIcon	Enables and disables icons and combo boxes on the Toolbar and Ribbon.
EnableMenu	Enables or disables a menu command.
EnableRMenu	Enables or disables right mouse menus by object type.
ExecuteMenu	Lets you simulate a user clicking one of the ReportSmith menu items.
ExecSQL	Allows you to execute stored procedures and execute other SQL strings that don't return a result set.
Field\$	Retrieves the value of the specified field for the record number to which the data set of the currently active report is pointing.
FieldFont	Changes the font type, style attribute, point size, or color of a field in your report.
FieldText	Changes the display of the field.
GetAllFields\$ method	Gets a list of the fields available for a given table.
GetDataSources\$ method	Gets the available data sources as a string list.
GetDDEItem function	Allows you to retrieve the data that a DDE client application would receive in response to a DDE request to the specified item.
GetFieldName\$ function	A global filter that returns the column name of the data field.
GetIncludePath\$	Gets the default directory for macro included files.
GetNext	Causes the data set in an active report to point to the next record.
GetPrevious	Causes the data set in an active report to point to the record before the current record.
GetRandom	Causes the data set in an active report to point to the record specified by the RecordNumber% argument.
GetRecordLimit function	Gets the total number of records that ReportSmith will download for any loaded or created report.

Command	Description
GetRepVar	Retrieves the value of a report variable in the active report.
GetSQL()	Returns a string of the text of the last SQL statement executed.
hWin_Active() function	Gets the window handle of the currently active report.
hWin_RS() function	Allows you to get the ReportSmith main window handle.
IsMenuChecked	Lets you determine if a given menu item has a check mark next to it.
IsMenuEnabled	Lets you determine if a given item is enabled or grayed.
KillMenu	Removed one of the ReportSmith menu items.
LastLoaded\$() function	Allows you to get the filename and path of the last <i>.rpt</i> file loaded into ReportSmith.
LoadReport	Loads a report.
PrintReport	Prints the specified pages of the active report to the specified printer.
Recalc statement	Re-executes the currently active report to reflect changes made to report variables or with an associated dataset control object.
RecordCount	Gives the total number of records in the data set belonging to the currently active report.
ResumeEvent	Lets an event-linked macro determine if the event should be executed or aborted.
Rgb	Returns a value for the color the FieldFont command uses for its color assignment.
RunMacro	Lets you execute a macro for another macro.
SetDataFilter statement	Allows you to specify a macro that will be executed before any datafield column value is calculated.
SetRepVar	Stores a value in a report variable in the active report.
SetDDEItem statement	Allows you to set or change the data that a DDE client application will receive in response to a DDE request to the specified item.
SetIncludePath	Sets to default directory for macro include files.
SetRecordLimit statement	Sets the total number of records ReportSmith will download for any loaded or created report.
SetSQL	Replaces the SQL string that would normally be generated by ReportSmith.
ShowRS	Lets you hide, show, minimize and maximize ReportSmith.
Str2Date command	The Str2Date command allows you to convert from a date string to a Julian date code.
SelectReport statement	Allows you to set the input focus to the report that has the indicated title.
SumField	Gives you the value of a summary field.
TotalPages function	Returns the number of pages in the currently active report.

ActiveTitle\$ function

Gets the window title of the currently active report.

Syntax ActiveTitle\$

Comments For saved reports, the title contains the path name and file name of the report's .RPT file.

Example

```
ActiveReport$=ActiveTitle$
Msgbox "Your report is called" + ActiveReport$
```

AddMenu

Allows you to add your own commands to ReportSmith's Main Menu to execute a macro when the command is chosen.

Syntax AddMenu MenuText\$, Macro\$, AfterMenu\$, HelpText\$

Parameters

MenuText\$	A string that specifies the text of the new menu item.
Macro\$	A string that specifies which macro should be run when the menu item is selected by a user.
AfterMenu\$	A string that specifies which existing menu item the new menu item should be placed after. You specify this menu name in the following format (omitting accelerator characters, underlines, and so forth): (MenuName SubMenuName)
HelpText\$	A string that specifies the text of a help line on the status bar.

Comments You can have ReportSmith execute an active report macro or global macro by specifying a macro name. (The name that appears in the active macro list in the macro dialog box.) You can also have ReportSmith execute a .MAC file by passing a string with a path and file name of the .MAC file you wish to run.

Note If one of these commands was entered in a global macro that was tied to the Startup of the application, then the menu would be customized each time ReportSmith is brought up.

This command can be used to check the state of the menu that is used for new reports by default by placing an exclamation point (!) before the menu name. This can be done whether the menu item is specified by command or relative location.

Examples

```
AddMenu "Open Sales Reports", "LoadSales", "!File|Open"
```

Or,

```
AddMenu "Open Sales Reports",  
"C:\RPTSMITH\MACROS\LSALES.MAC", "File|Open"
```

CloseReport

Closes the active report.

Syntax CloseReport Conditional%

Parameters

Conditional% If the integer argument is FALSE (zero), the report closes unconditionally. If CloseReport is called with a non-zero value, reports that were modified since last opened prompt the user to save the report before closing it and allow the user to cancel the close.

Returns Returns 0 if the active report was closed successfully. It returns 1 if there is no active report, or -1 if the user canceled a conditional close.

Comments When you use this command as a function (rather than a statement), you must enclose its arguments within parentheses. For more information on the differences between functions and statements, refer to “Introduction to macros” on page 159.

Example CloseReport 1

CloseRS

Closes ReportSmith.

Syntax CloseRS Conditional%

Parameters

Conditional% If the argument “Conditional%” is TRUE (non-zero), then ReportSmith brings up dialog boxes prompting the user to do certain tasks, such as save unsaved files. If the argument is FALSE (zero), then ReportSmith closes without bringing up dialog boxes or allowing the user to cancel the process.

Returns Returns non-zero if a conditional close was canceled.

Comments When you use this command as a function (rather than a statement), you must enclose its arguments within parentheses. For more information on the differences between functions and statements, refer to “Creating and working with macros” on page 162.

Example

```
If User_Response$ = 'No'
CloseRS 0
End if
```

Connect

Opens a connection to a database server.

Syntax Connect Type, Server\$, UserId\$, Password\$, Database\$

Parameters

Type	A number that identifies the type of database to which you are connecting:
	0 Named Connection
	1 Reserved
	2 DBASE
	3 EXCEL
	4 PARADOX
	5 ASCII
	6 SQLSRVR
	7 ORACLE
	8 DB2
	10 SYBASE
	11 BTRIEVE
	12 GUPTA
	13 INGRES
	16 TERADATA
	17 DB2/GUPTA
	19 UNIFY
	40 DBASE (via ODBC)
	41 EXCEL (via ODBC)
	42 PARADOX (via ODBC)
	48 BTRIEVE (via ODBC)
	55 Generic ODBC driver (use this for most ODBC connections)
	61 PARADOX (via IDAPI)
	62 DBASE (via IDAPI)
	67 INTERBASE (via IDAPI)
Server\$	Identifies the server that will be used to make the connection.
UserId\$	Identifies the user making the connection.
Password\$	The password of the user making the connection.
Database\$	The name of the database to connect to or the file name of a local database.

Comments For local databases (such as dBASE), Server\$, UserId, Password\$, and Database\$ should be set to an empty string. If any of these parameters are not valid for your connection type, use a null string.

Example Connect 6, "sqlsrvr", "myuser", "mypassword", "mydb"

CreateDDEItem statement

Creates a new Dynamic Data Exchange item in ReportSmith's DDE server.

Syntax CreateDDEItem TopicName\$, ItemName\$, RequestMacro\$, Pokemacro\$

Parameters

TopicName\$	The name of a user-defined DDE topic.
ItemName\$	The name of a user-defined DDE item.
RequestMacro\$	The name of a macro to call before a DDE request on this topic is serviced.
Pokemacro\$	The name of a macro to call after a DDE poke to this topic is serviced.

Comments User options include specifying the name of a macro to execute before data is requested from this item and specifying another name of a macro to be executed after data is poked into this item. Those macros could use other commands to retrieve or change the data in the DDE item. In order for this function to execute successfully, a Topic must exist that matches the TopicName\$ parameter.

This function enables you to create duplicate item names. In this case, only the first item of a given name can be accessed. At this time, a DDE item remains in effect as long as ReportSmith is executing. Eventually, functions are added to get a list of available DDE items and to remove existing items.

Example `CreateDDEItem "MyTopic", "MyItem", "RequestProc", "PokeProc"`

CreateDDETopic statement

Allows you to create a new Dynamic Data Exchange topic in ReportSmith's DDE server. After a topic is created, DDE items can be added to it.

Syntax `CreateDDETopic TopicName$`

Parameters

TopicName\$	The name of a user defined DDE topic.
-------------	---------------------------------------

Comments This function allows you to create duplicate topic names. In this case, only the first topic of a given name can be accessed. At this time a DDE topic remains in effect as long as ReportSmith is executing. Eventually, functions will be added to get a list of available DDE topics and to remove existing topics.

Example `CreateDDETopic "MyTopic"`

Current

Returns the record number to which the data set (belonging to the active report) is pointing. In other words, it tells you what record number the Field\$ function returns when executed.

Syntax `Current`

Returns This function returns the record number to which the data set of the currently active report is pointing.

Example

```
If Current = 1 then
  MsgBox "At the beginning"
EndIf
```

CurrentPage function

Returns the page number that is displayed in the currently active report.

Syntax CurrentPage

Returns Returns the displayed page number.

Comments This function is useful in changing the active report for functions that work on the currently active report.

Example MsgBox "The active report is on page: " + str\$(CurrentPage)

Date2Str

Converts date from a Julian date code to a date string.

Syntax Date2Str\$(JulianDate, Format%)

Parameters

JulianDate The number of days since January 1, 0000.

Format% Code to indicate the format of the date string returned. The output is a character string in the format:

- 1 "mmddyy"
- 2 "mmddyyyy"
- 3 "mm/dd/yy"
- 4 "mm/dd/yyyy"

Example Suppose you want to generate a date string for three days before the current date taking into account Leap Year and month boundaries. You could accomplish this the following way:

```
Today$ = Date$
```

Note See "Date\$ function" on page 304.

```
JDate = Str2Date(Today$,1)
ThreeDaysAgo$ = Date2Str(dDate,3)
```


DateField

Gives you date values.

Syntax DateField (Field\$, Code%)

Parameters

Field\$	A date DataField in your report.
Code%	The number from the table in the Comments section below that indicates the portion of the date you want returned: 1 Gets the number of days since the date field (based on a Julian date). 2 Gets the month of the date as a number. 3 Gets the day of the month. 4 Gets the year. 5 Gets the day of the week. 6 Gets the day of the year out of 365 days. Any other value returns a zero and indicates an error.

Returns Specific date information.

Comments To use this command, choose a date field from the Data Fields list box and add a code. The number specifies the type of information you want about the date.

Example `Year_Hired = DateField ("Hire_Date", "4")`

DDEExecute

The DDEExecute command lets you send a DDE execute command to a DDE client application.

Syntax DDEExecute Service\$, Topic\$, Command\$

Parameters

Service\$	A string that specifies the DDE service to which you send the Execute command. (This is usually the name of the application that receives the command.)
Topic\$	A string that identifies the DDE topic to which you send the command. (Most applications accept DDE Execute messages that are sent to their SYSTEM topics.)
Command\$	A string that identifies the DDE command itself. See the documentation of the application to which you're sending the DDE command for details of the DDE Execute commands it accepts.

Returns Zero if the command is sent successfully, or nonzero on error.

Comments When you use this command as a function (rather than a statement), you must enclose its arguments within parentheses. For more information on the differences between functions and statements, refer to “Creating and working with macros” on page 162.

Example The following line of code will send a DDE Execute command causing Excel to beep.

```
Success = DDEExecute("Excel", "System", "[Beep(0)]")
```

DDEPoke

Lets you poke a string to a DDE client application.

Syntax DDEPoke Service\$, Topic\$, Item\$, Data\$

Parameters

Service\$	A string that specifies the DDE service to which you’re going to poke. (This is usually the name of the application receiving the command.)
Topic\$	A string that identifies the DDE topic to which you send the command. (Most applications use the topic to identify the document into which you want to poke the data.)
Item\$	A string that identifies the DDE item into which you want to poke data.
Data\$	A string, which is actually the data poked into the other application. See the documentation of the application to which you’re sending the DDE poke command for details on the accepted DDE poke commands.

Returns Zero if the command is sent successfully or nonzero on error.

Comments When you use this command as a function (rather than a statement), you must enclose its arguments within parentheses. For more information on the differences between functions and statements, refer to “Creating and working with macros” on page 162.

Example The following line of code puts the string “My Data” in the first cell of a default Excel spreadsheet.

```
Success = DDEPoke("Excel", "Sheet1", "R1C1", "My Data")
```

DDERequest

Gets data from a DDE server application as a string.

Syntax DDERequest(Service\$, Topic\$, Item\$)

Parameters

Service\$	A string that specifies the DDE service to which you're going to request. (This is usually the name of the application that receives the command.)
Topic\$	A string that identifies the DDE topic to which you're going to send the command. (Most applications use the topic to identify the document into which you want to retrieve the data.
Item\$	A string that identifies the DDE item into which you want to retrieve data.

Returns A string that has the requested data, or the string "<ERROR>" if the operation is not performed successfully.

Comments See the documentation of the application to which you're sending the DDE poke command for details on the accepted DDE poke commands.

Example The following line of code puts the data in the first cell of a default Excel spreadsheet into a string variable called Data\$.

```
Data$ = DDERequest( "Excel", "Sheet1", "R1C1" )
```

DerivedField

Sets the value of a Derived Field.

Syntax DerivedField Value\$

Parameters

Value\$	A quoted value or string variable.
---------	------------------------------------

Comments The DerivedField command only has an effect in macros you use as derived fields. It takes a string that is used for the derived field. If the value for the derived field is a number, convert it to a string using the STR\$ function.

Example DerivedField "Bill Smith"

DoEvents

Allows other Windows applications to process messages.

Syntax DoEvents

Comments Use this function when you want your Basic code to yield processor time to allow other applications to process messages.

Example DoEvents

EnableIcon

Enables and disables icons and combo boxes on the Toolbar and Ribbon.

Syntax EnableIcon GroupNo, ItemNo, EnableFlag

Parameters

GroupNo	Index of the icon group to which the icon or combo box belongs.
ItemNo	Index of the item within the group.
EnableFlag	0 is to <i>disable</i> , 1 is to <i>enable</i> .

Example The command below disables the Italic button so that it is not available:

```
EnableIcon 14,2,0
```

EnableMenu

Enables or disables a menu command.

Syntax EnableMenu Menu\$, EnableCode%

Parameters

Menu\$	The Menu name and menu submenu you want to be affected.
EnableCode%	Specify 1 to <i>enable</i> , 0 to <i>disable</i> .

Returns Zero if a menu was removed successfully and -1 if a menu of the given name was not found when used as a function.

Comments This command takes a string that specifies a menu item or a submenu item. The string uses this format:

```
"MenuName|SubMenuName"
```

The names must match our menu commands, not including keyboard accelerators and ellipsis (...) characters. If you omit the pipe and submenu name, the routine assumes you're working with a top-level menu. If a top-level menu is disabled, all of its submenu items are also disabled.

When you use this command as a function (rather than a statement) you must enclose its arguments within parentheses. For more information on the differences between functions and statements, refer to "Creating and working with macros" on page 162.

Example The following line of code *disables* the ReportSmith File|New menu item.

```
Success = EnableMenu( "File|New", 0 )
```

EnableRMenu

Enables or disables right-mouse menus by object type.

Syntax EnableRMenu ObjectType, EnableCode%

Parameters

ObjectType A code that specifies the object type to disable right mouse menu items for.

- 1 Character
- 2 Border
- 3 Field Format
- 4 Text Alignment
- 5 Display As Picture
- 6 Column Width
- 7 Field Height
- 8 Selection Criteria
- 9 Translate

EnableFlag 0 Disables the menu
 Non-zero enables the menu

Comments This command used to check the state of the menu that is used for new reports by default by placing an exclamation point (!) before the menu name. This can be done whether the menu item is specified by command or relative location.

Example The following example will disable the right mouse abilities for text fields.

```
EnableRMenu 1,0
```

ExecuteMenu

Simulates a user clicking one of the ReportSmith menu items.

Syntax ExecuteMenu Menu\$

Parameters

Menu\$ The menu name, submenu name, or both that you want to execute.

Returns Zero if a menu was executed successfully, and -1 if a menu of the given name wasn't found.

Comments This command takes a string that specifies a menu item or a sum menu item. The string uses this format:

```
"MenuName|SubMenuName"
```

The names must match our menu commands, not including keyboard accelerators and ellipsis (...) characters. If you omit the pipe and submenu name, the routine assumes you're working with a top level menu.

When you use this command as a function (rather than a statement), you must enclose its arguments within parentheses. For more information on the differences between functions and statements, refer to "Creating and working with macros" on page 162.

Note This command can be used to check the state of the menu that will be used for new reports by default by placing an exclamation point (!) before the menu name. This can be done whether the menu item is specified by command or relative location.

Example Execute Menu "File|Exit"

ExecSQL

Executes stored procedures and other SQL strings that don't return a result set.

Syntax ExecSQL SQL\$, Type, Server\$, UserID \$, Password\$, Database\$

Parameters

R	The SQL statement to be executed.
Type	A number that identifies the type of database to which you are connecting:
	0, 1 Reserved
	2 DBASE
	3 EXCEL
	4 PARADOX
	5 ASCII
	6 SQLSRVR
	7 ORACLE
	8 DB2
	9 NETSQL
	10 SYBASE
	11 BTRIEVE
	12 GUPTA
	13 INGRES
	14 DB2GUPTA
	15 OCELOT
	16 TERADATA
Server\$	Identifies the server used making the connection.
UserID\$	Identifies the user making the connection.
Password\$	Identifies the user's password.
Database\$	The name of the database to connect to or the file name of a local database.

Comments For local databases (such as dBASE), Server\$, UserID\$, and Password\$ should be set to an empty string.

Example `ExecuteSQL "sp_my_proc", 10, "mysrvr", "myun", "mypw", "mydb"`

ExecuteMenu

Syntax `ExecuteMenu Menu$`

Definition The ExecuteMenu command lets you simulate a user clicking a ReportSmith menu item.

Parameters

`Menu$` The menu name and/or submenu name that you want to execute.

Returns This function returns 0 if the menu was executed successfully, and -1 if a menu of the given name wasn't found.

Comments It takes a string that specifies a menu item or a submenu item. The string uses this format:

`"MenuName | SubMenuName"`

The names must match our menu commands, not including keyboard accelerators and ellipses. If you omit the pipe and submenu name, the routine assumes you are working with a top level menu.

When you use this command as a function (instead of a statement), it requires the return value, and you must enclose its arguments within parentheses.

Example `Execute Menu "View | Zoom"`

Field\$

Retrieves the value of the specified field for the record number to which the data set of the currently active report is pointing. This statement is always used as a function.

Syntax `Field$( FieldName$ )`

Parameters

`FieldName$` A field in your report that you want the value of.

Returns The value of the specified field as a string regardless of the retrieved field's data type. If the specified field is not found, returns "N\A" or "<ERROR>".

Comments The name of the field should exactly match the database column name.

You can link two tables together that have one or more column names in common. In this case, it's necessary to use the fully-qualified field name to insure that you're getting the correct field. The fully-qualified field name includes the table name followed by a period (.) followed by the field name. You can also use a field or table alias. The best way

to get the fully-qualified field name is to drag the field you want from the list box that appears on the left of the Edit Macro dialog box into the Formula box.

Example `NextId$ = Field$("Employee_Id")`

FieldFont

Changes the font type, style attribute, point size, or color of a field in your report. This command is usually used in a macro that is linked to the display event of a field.

Syntax `FieldFont Facename$, PointSize, Style, ForColor, BackColor`

Parameters

Facename\$	The font name.
PointSize	The point size.
Style	The style code indicating: 1 Text fields 2 Sections 3 Draw windows 4 Crosstabs 5 Crosstab cells 6 General 7 Reserved
ForColor	The foreground text color.
BackColor	The background text color.

Comments Add the style codes to combine attributes. For example, a 3 designates bold italic. (Using the codes on the previous page, 1 + 2 = 3). For the fourth and fifth arguments, use a color value for the last two color arguments.

You can specify one of 16 million colors using the `Rgb` command. Windows substitutes the closest color to the one you select.

Note If you do not want to change the attribute, point size, or color, use -1 (negative one).

Example The following example changes the font to red, italic, 14 point Arial:

```
Field Font "Arial", 14, 2, RGB(255,0,0), -1
```

FieldText

Applies to macro fields that are linked to the text event of a data field object. It changes the display of the field.

Syntax `FieldText Text$`

Parameters

Text\$ The text used to replace a field's data.

Example If you have a field that contains a person's first name, but would prefer to use their nickname, use a FieldText command similar to this one:

```
If field$("FirstName")="James" then
    FieldText "Jim"
End If
```

GetAllFields\$ method

Gets a list, separated by commas, of the fields available for a given table.

Syntax [object.]GetAllFields (Table\$, Database\$)

Note First a connection must be made and the table added to the data set before its field list can be retrieved.

Parameters

Table\$ The name of the table for which you're getting the list of fields.

Database\$ The name of the table's database. This applies to tables with databases.

Returns A list of all fields that are available in the given table.

Comments The Table\$ parameter is the path and file name for local databases. For database servers, it takes the form Owner.TableName. For local databases or servers that don't require that a database be specified the dBASE parameter should be left blank.

Example MsgBox GetAllField\$ ("dbo.emp", "Indigo")

GetDataSources\$ method

Gets the available data sources, separated by commas, as a string list.

Syntax [object.]GetDataSources

Returns A list of all of the data sources available to ReportSmith, including ODBC Sources, separated by commas.

Example MsgBox "The available datasources: " + GetDataSources\$

GetDDEItem function

Retrieves the data that a DDE client application would receive in response to a DDE request to the ReportSmith item.

Syntax `GetDDEItem$(TopicName$, ItemName$)`

Parameters

`TopicName$` The name of user defined DDE topic.

`ItemName$` The name of a user defined DDE item.

Returns Data in the DDE item.

Comments A macro that is named as the poke response macro for a user-defined DDE item can use this function to get the data that a DDE client has poked into the item. You can use this technique to create a DDE item that will load a report when a DDE client application pokes the filename of that report into the item.

Example `TheData$=GetDDEItem("MyTopic", "MyItem")`

GetFieldName\$ function

This function is a global filter, returning the column name of the data field for which the filter or conditional formatting macro is being called.

Syntax `GetFieldName$()`

Returns Returns the column name of the data field.

Comments See also “SetDataFilter statement” on page 260.

Example `CurrentValue$=Field$(GetFieldName$())`

GetIncludePath\$ command

Gets the default directory for macro include files.

Syntax `GetIncludePath$()`

Returns A string that is the default path for macro include fields.

Example `MsgBox"Looking for include files in"+GetIncludePath$()`

GetNext

Causes the data set in an active report to point to the record that comes immediately after the record to which it is currently pointing.

Syntax GetNext

Example This example steps through the entire report counting the employees with the first name 'John.'

```
Get Random (1)
  For x = 1 to RecordCount()
    If Field$("First_Name")="John" then
      Count=Count+1
    Get Next
  Next X
```

GetPrevious

Causes the data set in an active report to point to the record that comes immediately before the record to which it is currently pointing.

Syntax GetPrevious

Comments The GetPrevious command can be used to change the current record of a data set. The current record determines what data the Field\$, SumField\$ and DateField\$ functions retrieve. This function could be used in a macro-defined summary field. When a macro defined summary field is dropped in a group footer the current record is the last record in that group. For this reason, a macro derived field can step backwards through the group performing custom summary operations. See also “GetNext” on page 250, “GetRandom” on page 251, “Field\$” on page 247, “Current” on page 239.

Example

```
'Position to the 3rd record
GetRandom 3
'Now go to 2
GetPrevious
```

GetRandom

Causes the data set in an active report to point to the record specified by the RecordNumber% argument, if that record number exists.

Syntax GetRandom RecordNumber%**Parameters**

RecordNumber% The number of the record in the data you want to navigate to.

Comments See also “GetPrevious” on page 251, “GetNext” on page 250, “Field\$” on page 247, “SumField” on page 264, “DateField” on page 241, “Current” on page 239, and “TotalPages function” on page 265.

Example

```
'Point to the 23rd record in a set of data
GetRandom 23
```

GetRecordLimit function

Gets the total number of records that ReportSmith will download for any loaded or created report.

Syntax GetRecordLimit

Comments This limit is set with the function SetRecordLimit.

Example TheLimit=GetRecordLimit

GetRepVar

Retrieves the value of a report variable in the active report. This command is only used as a function.

Syntax GetRepVar(ReportVariable\$)

Parameters

ReportVariable\$ The name of a report variable in your report.

Returns The value of the specified report variable as a string. This function returns "<ERROR>" if a report variable of the specified name cannot be found in the active report.

Comments GetRepVar takes a string argument that specifies the name of the report variable being retrieved. See also "SetRepVar" on page 260, "Val function" on page 355, and "Str2Date" on page 263.

Example Var_name\$=GetRepVar("Rep_var_name")

GetSQL

Returns a string that is the text of the last SQL statement that ReportSmith executed.

Syntax GetSQL

Comments This function can be used along with the SetSQL statement in a macro that is linked to the "Before SQL is Executed" to change the SQL string "On the Fly."

Example The following stores the last generated SQL statement in a variable called The_SQL\$.

```
The_SQL$ = GetSQL$
```

hWin_Active() function

Gets the window handle of the currently active report.

Syntax hWin_Active()

Comments This function can be used along with Windows API functions. You make the API functions available to ReportBasic through the use of the declare function.

Example

```
'Force the active report to an Icon
Result=ShowWindow(hWin_Active(),2)
Declare Function ShowWindow Lib "User" (ByVal hWnd As Integer, ByVal nCmdShow As Integer) As Integer
```

hWin_RS() function

Gets the ReportSmith main window handle.

Syntax hWin_RS()

Comments You can use this function along with Windows API functions. You make the API functions available to ReportBasic through the use of the declare function.

Example RS_Handle=hWin_RS()

IsMenuChecked

Determines if a given menu item has a check mark next to it. This command is only used as a function.

Syntax IsMenuChecked (Menu\$)

Parameters

Menu\$ The menu name and/or submenu name that you are interested in.

Returns Returns 1 if the menu is checked, 0 if it isn't checked, and -1 if a menu of the given name was not found.

Comments This function takes a string that specifies a menu item or a submenu item. The string is of the form "MenuName|SubMenuName." The names must match our menu commands, not including keyboard accelerators and ellipsis (...) characters. If you omit the pipe and submenu names, the routine assumes you're working with a top level menu.

This function does not correctly return the state of a menu item if it is called before the menu is visible as in the case of a macro linked to the 'Application Startup' event.

Note This command can be used to check the state of the menu that is used for new reports by default by placing an exclamation point (!) before the menu name. This can be done whether the menu item is specified by command or relative location. Refer to the second example. Also, refer to “EnableMenu” on page 244, “KillMenu” on page 255, “AddMenu” on page 236, “ExecuteMenu” on page 245 and “IsMenuEnabled” on page 254.

Example

```
Success = IsMenuChecked ("View|Boundaries")
```

Or,

```
If IsMenuChecked ("!View|Boundaries") = 1 Then
  ExecuteMenu "View|Boundaries"
End if
```

IsMenuEnabled

Determines if a given menu item is enabled or grayed. This command is used only as a function.

Syntax IsMenuEnabled (Menu\$)

Parameters

Menu\$ The menu name and/or submenu name that you are interested in.

Returns 1 if the menu is enabled, 0 if it is disabled, and -1 if a menu of the given name was not found.

Comments It takes a string that specifies a menu item or a submenu item. The string uses this format:

```
"MenuName|SubMenuName"
```

The names must match ReportSmith menu commands, not including keyboard accelerators and ellipsis (...) characters. If you omit the pipe and submenu name, then the routine assumes you’re working with a top level menu.

Note This command can be used to check the state of the menu that will be used for new reports by default by placing an exclamation point (!) before the menu name. This can be done whether the menu item is specified by command or relative location. Refer to the second example. Also, refer to “IsMenuChecked” on page 253, “EnableMenu” on page 244, “KillMenu” on page 255, “AddMenu” on page 236 and “ExecuteMenu” on page 245.

Example

```
Success = IsMenuEnabled ("Edit|Cut")
```

Or,

```
'Check if the tables Menu is enabled
If IsMenuEnabled ("Tools|Tables") = 1 Then
MsgBox "Tables Menu Enables"
End if
```

KillMenu

Removes one of the ReportSmith menu items.

Syntax KillMenu Menu\$

Parameters

Menu\$ The menu name and submenu name, or both, that you are interested in.

Returns 0 if a menu was removed successfully, and -1 if a menu of the given name was not found.

Comments It takes a string that specifies a menu item or a submenu item. The string uses this format:

```
"MenuName|SubMenuName"
```

The names must match ReportSmith menu commands, not including keyboard accelerators and “...” characters. If you omit the pipe and submenu name, the routine assumes you’re working with a top-level menu.

This command can be used to change the state of the menu that will be used for new reports by default by placing an exclamation mark (!) before the menu name. This can be done whether the menu item is specified by command or relative location. (!) removes the menu item for all future reports for this session of ReportSmith, instead of just the active report.

When you use this command as a function (rather than a statement), you must enclose its arguments within parentheses. For more information on the differences between functions and statements, refer to “Creating and working with macros” on page 162.

Example The following code fragment will remove the File|New menu Item from ReportSmith.

```
Success = KillMenu( "File|New" )
```

LastLoaded\$ function

Gets the filename and path of the last .rpt field that was loaded into ReportSmith.

Syntax LastLoaded\$

Example MsgBox"The last report Loaded was:"+LastLoaded\$

List RepVar

Syntax ListRepVar\$(Index%)

Definition ListRepVar returns the name of the report variable at the specified index in the active report. If no report variable exists at the given index, then the command returns a NULL string. If this command is executed in a macro linked to the “Before Report Open” event of the Report Object, or the After Data Read event of the application object, the command will return the report variable names for the loading report. ListRepVar can be used to get the list of variables in a report and initialize them before they are prompted.

Parameters

Index% Index% specifies the report variable name. If there is no report variable at this index, the function returns a null string.

Example

```
' Put the names of the report variables in an array
dim Variables$(20 )
'Allow for a maximum of 20 report variables
while ListRepVar( CurrentVar )<>""
    Variables$( CurrentVar ) = ListRepVar( CurrentVar )
    CurrentVar = CurrentVar + 1
wend
```

LoadReport

Loads a report.

Syntax LoadReport Filespec\$, InitString\$

Parameters

Filespec\$ The directory path and name of the report (*.rpt* file) you would like to run.
 InitString\$ You can use the InitString argument to set report variables before SQL is executed. Report variables set in this manner will not prompt the user to enter their values.

This is the format for the InitString argument:

```
@Report_Variable1=<Value1>,@ReportVariable2=<Value2>, ...
```

Specify the report variables you would like to set in the report.

Returns Non-zero on error.

Comments Enter the full path name to the *.RPT* file in which you want the macro to load, then make it the active report.

When you use this command as a function (rather than a statement), you must enclose its arguments within parentheses. For more information on the differences between functions and statements, refer to “Creating and working with macros” on page 162.

Example `LoadReport "c:\summary.rpt", "@Repvar1=<40>,@Repvar2=<'Smith'>"`

PrintReport

Prints the specified pages of the active report to the specified printer. This init string cannot be used to initialize variables that belong to the detail section in master/detail reports. All arguments are optional.

Syntax `PrintReport [StartingPage%], [EndingPage%], [Printer$], [Port$], [Driver$],[Copies%]`

Parameters

<code>StartingPage%</code>	The number of the page of the report you would like to start printing.
<code>EndingPage%</code>	The number of the page you would like to stop printing.
<code>Printer\$</code>	The name of the printer to which you would like to print the report.
<code>Port\$</code>	The correct port specification.
<code>Driver\$</code>	The correct driver specification.
<code>Copies%</code>	The number of copies to print.

Returns Non-zero on error.

Comments To print all report pages, use 0 for the start and end page parameters. To use the default printer, use null strings for the `Printer$`, `Port$`, and `Driver$` arguments. To specify a printer, see the Devices section of your WIN.INI file. You'll see printers listed in this format:

```
Printer=Driver,Port1,Port2, ...
```

When you specify the printer, driver and port, use the same text you see in the WIN.INI file.

When you use this command as a function (rather than a statement), you must enclose its arguments within parentheses. For more information on the differences between functions and statements, refer to “Creating and working with macros” on page 162.

Example `PrintReport 1,5, "HPLaserJet 4/4M", "LPT3", "HPPCL5E"`

Recalc statement

Re-executes the currently active report so that the report surface will reflect changes made to report variables or changes made with an associated dataset control object.

Syntax `Recalc`

Example The following example sets the starting date report variable to today and recalculates.

```
SetRepVar("StartDate",Date$)
Recalc
```

RecordCount

Gets the total number of records in the data set that belongs to the currently active report.

Syntax RecordCount

Returns The number of records in the currently active report or 0 if no active report exists.

Comments It's useful when writing macros that step through the data in a report.

Example The following code fragment counts all of the customers from the city of Palo Alto in a customer database and displays the result.

```
GetRandom 1
For X = 1 to RecordCount
If Field$("CITY") = "Palo Alto" then Count = Count + 1
GetNext
Next X
MsgBox "The total number of customers located in Palo Alto are: "+ Str$(Count)
```

ResumeEvent

Lets a macro which is linked to an event determine if the event should be executed or aborted.

Syntax ResumeEvent ResumeCode%

Parameters

ResumeCode% The code which indicates the event:

- 0 Abort the event to which this macro is linked.
- 1 Perform the event as usual. (Default)

Comments This only applies to certain events. For most events, 0 means abort and 1 means proceed. Some events recognize more codes. See the individual event for more information.

Example The following example stops an event.

```
ResumeEvent 0
```

Rgb

Returns a value for the color that the FieldFont command uses for its color assignment.

Syntax Rgb (Red%, Blue%, Green%)

Parameters

Red%, Blue%, Green% The intensity of red in color, blue in color, or green in color.

Returns An integer representing the color you've specified.

Comments This command uses three numbers from 0 to 255. The first number designates the intensity of red, the second number of blue, and the third of green. It provides 16.5 million color combinations. Windows then chooses the closest match to the specified color.

Example For example, this tells the FieldFont command to use the color red.

```
Dim Red as Integer
Red = Rgb (255, 0, 0)
```

Tip Use the Custom Color option in the Windows Control Panel to select a color. Take note of the Rgb settings for the color you select. Then use these values in the Rgb function to get the color you need.

RunMacro

Executes a macro from another macro.

Syntax RunMacro Macro\$, Arguments\$

Parameters

Macro\$ Specifies the macro that's being run.

Arguments\$ The arguments defined in your macro if they exist. If your macro contains no arguments, just use a null argument ("").

Returns 0 if a macro is found and successfully executed.

Comments The macro language first looks for an active global macro that matches the name and then searches for active macros that belong to the active report. You can also specify the file name of a .MAC file. To make sure the correct .MAC file is executed, specify the full path of the .MAC file.

When you use this command as a function (rather than a statement), you must enclose its arguments within parentheses. For more information on the differences between functions and statements, refer to "Introduction to macros" on page 159.

Examples

```
RunMacro "c:\rptsmith\macros\greeting.mac", ""
```

```
RunMacro "two_arguments", "1.2"
```

SetDataFilter statement

Executes before any data field column value is calculated. The specified macro can then use the FieldText and FieldFont functions to change either the text or the appearance of the data.

Syntax SetDataFilter MacroName\$

Parameters

MacroName\$ The path and file name of a .MAC file or name of an actual global macro to be used as the macro data filter.

Comments This function can be costly in performance under some circumstances as the specified macro is executed once for each field visible on the report surface. The filter function can be disabled by calling this function with a null macro name.

Example SetDataFilter "FilterMacro"

SelectReport statement

Sets the input focus to the report that has the indicated title.

Syntax SelectReport ReportTitle\$

Parameters

ReportTitle\$ The title of the report to set focus.

Comments This statement is useful in changing the active report for functions that work on the currently active report.

Example SelectReport "c:\rptsmith\sales.rpt"

SetRepVar

Stores a value in a report variable in the active report.

Syntax SetRepVar ReportVariable\$, Value\$

Parameters

ReportVariable\$ A string argument that specifies the name of the report variable being set
Value\$ A string argument that specifies what value to set the report value to.

Returns Non-zero on error.

Comments When you use this command as a function (rather than a statement), you must enclose its arguments within parentheses. For more information on the differences between functions and statements, refer to “Creating and working with macros” on page 162.

Example `SetRepVar "Repvar1", "Smith"`

SetDDEItem statement

Sets or changes the data in one of ReportSmith’s DDE items. This could be done in response to a DDE request to the specified item.

Syntax `SetDDEItem TopicName$, ItemName$, Data$`

Parameters

TopicName\$ The name of a user defined DDE topic.
ItemName\$ The name of a user defined DDE item.
Data\$ The data to set into a DDE item.

Comments A macro that is named as the request response macro for a user-defined DDE item can use this function to set the data that a DDE client receives in response to a DDE request. This technique could be used to create a DDE item that would report the name of the currently active report or other information that might be useful to a DDE client application.

Example `SetDDEItem "MyTopic", "MyItem", "Mydata"`

SetIncludePath command

Sets to default directory for macro include files.

Syntax `SetIncludePath Path$`

Parameters

Path\$ The path where RS_BASIC expects to find include files.

Returns Non-zero on error.

Comments The macro compiler first looks in this directory for include files, then it looks in the standard search path. The beginning value is the default macro path in the RPTSMITH.INI file.

Example `SetIncludePath "c:\macros\include"`

SetRecordLimit statement

Sets the total number of records that ReportSmith downloads for any loaded or created report. A value of 0 allows an unlimited number of records to be downloaded.

Syntax `SetRecordLimit Limit`

Parameters

Limit The maximum number of records that ReportSmith will download for a single report.

Comments This statement is helpful for implementing a draft mode where you can work with a subset of a large report until you are ready to work with the entire report. Some operations continue to work against the entire result set, such as selections, sorting, and summary fields, so that for these operations performance does not change.

Example `SetRecordLimit 100`

SetSQL

Replaces the SQL string that would normally be generated by ReportSmith.

Syntax `SetSQL SQL$`

Parameters

SQL\$ A quoted, valid SQL statement.

Comments The SetSQL statement is only valid in a macro that is linked to the “Before SQL is Executed” event. Care should be used when executing this function as the string is not verified before it is executed. This statement can be used along with the GetSQL function in a macro that is linked to the “Before SQL is Executed” event to change the SQL string “On the Fly.”

Example `SetSQL "Select ENAME, EMP_ID from SCOTT.EMP"`

ShowRS

Hides, shows, minimizes, or maximizes ReportSmith.

Syntax ShowRS Code%

Parameters

Code% The following codes are valid:

- 0 Hides ReportSmith and passes activation to another window.
- 1 Activates and displays ReportSmith. If ReportSmith is minimized or maximized, Windows restores it to its original size and position.
- 2 Activates ReportSmith and displays it as an icon.
- 3 Activates ReportSmith and displays it maximized.
- 4 Displays ReportSmith in its most recent size and position. The window that is currently active remains active.
- 5 Activates ReportSmith and displays it in its current size and position.
- 6 Minimizes ReportSmith and activates the top-level window in the system's list.
- 7 Displays ReportSmith as an icon. The window that's currently active remains active.
- 8 Displays ReportSmith in its current state. The window that's currently active remains active.
- 9 Activates and displays ReportSmith. If ReportSmith is minimized or maximized, Windows restores it to its original size and position.

Comments You minimize ReportSmith after it opens a modal dialog box, ReportSmith cannot be used until it's brought back to normal size. This may only be possible by sending another ShowRS command from DDE.

You cannot switch to it or execute any of its commands except through macros and DDE. ReportSmith can still bring up dialog boxes and do any other activity, but because it's not visible, these actions can only be started by DDE commands or macros which are already running.

Example

```
'Force ReportSmith to be an icon
ShowRS 2
```

Str2Date

Allows you to convert from a date string to a Julian date code.

Syntax Str2Date\$(Date\$,Type%)

Parameters

- Date\$** The date to convert as a string. The date can be provided in one of the following formats:
 "MM-DD-YYYY"
 "MM-DD-YY"
 "MM\DD\YYYY"
 "MM\DD\YY"
- Type%** Specifies the data information to be returned:
- 1 Gets the number of days since the date field (based on a Julian date).
 - 2 Gets the month of the date as a number.
 - 3 Gets the day of the month.
 - 4 Gets the year.
 - 5 Gets the day of the week.
 - 6 Gets the day of the year out of 365 days.

Returns A date value that depends on the Type parameter.

Comments The type argument has the same values as in the DateField\$ function.

Example Suppose you wanted to generate a date string for 3 days before a date provided by a user. You could accomplish this with the following code:

```
'Get a date from the user
UserDate$=inputbox("Enter a Date")
'Get the Julian Date Code for the user's date. 1 gets Julian date from a string.
JDate=Str2Date(UserDate$, 1)
'Get a string that represents a date that comes 3 days before the given date
ThreedaysBefore$=Date2Str(JDate-3)
```

SumField

Gives you the value of a summary field.

Syntax SumField\$ (Field\$, Table\$, GroupLevel, Operation\$)

Parameters

- Field\$** The name of the field being summed.
- Table\$** The name of the table being summed.
- GroupLevel** The group in the report at which the summary is reset.
- Operation\$** Summary operation performed on the Field.

Comments You can drag and drop this command from the list box. The best way to use it is to choose Summary Fields from the first Listbox and then double click on your Summary Field. ReportSmith uses the SumField\$ command and fills in the Parameters for you.

Example This is how a Summary Field should be referenced in macros.

```
my_var$= SumField$("QTY_FIELD", "OWNER.TABLE_NAME", 0, "Count")
```

TotalPages function

Returns the number of pages in the currently active report.

Syntax TotalPages

Comments This function is useful in changing the active report for functions that work on the currently active report.

Example MsgBox "There are" + str\$(Totalpages)+"in the current report"

DataSet control methods summary

Methods are functions or statements that perform actions on the control. The SetFromActive method associates the DataSet control with the currently active report. The Save method can be used to store a DataSet control in a DOS file.

Method	Description
AddGroup	Adds grouping criteria.
AddSort	Adds a sorting criteria.
AddSummary	Adds a summary field.
AddTable	Adds a table.
Connect	Replaces previous connection information with supplied connection information.
Disconnect	Removes a connection previously set with the Connect method.
Field\$	Returns the value of the specified data field for the current data set record.
GetAllFields\$	Returns a list of all fields available in the given table.
GetColumnAlias\$	Returns the alias for the specified field in the specified table.
GetDataSources\$	Returns a list of all data sources available to ReportSmith.
GetFieldList\$	Returns a list of all fields included in the given table.
GetGroup\$	Returns a string providing information about grouping.
GetSort\$	Returns a string indicating the sorting criteria at the given level.
GetSummary\$	Returns a string providing information about a summary field.
GetSQL\$	Returns the last SQL statement executed for the data set control object.
GetTable\$	Returns a string describing the table at the specified index.
GetTableAlias\$	Returns the alias for the specified table.
GetTableLink\$	Returns a string providing information about the table link.
IncludeFields\$	Adds columns from a table to your dataset.
Load	Uses any previous connection information with information from the file specified by the Filename\$ parameter.
Recalc	Re-executes the SQL for the data set control object.
RemoveGroup	Removes a grouping criteria from a report.

Method	Description
RemoveSort	Removes the sorting criteria at the given level.
RemoveSummary	Returns a summary field from a report.
RemoveTable	Removes the table at the specified index.
RemoveTableLink	Removes the table link at the specified index from the dataset control object.
ReplaceTable	Replaces one table in a report with another.
Save	Replaces any previous connection information with information from a file specified by the Filename\$ parameter.
SetColumnAlias	Sets or changes the Alias for a column in a table.
SetFromActive	Replaces any previous connection information with a reference to the data description for the currently active report.
SetFromLoading	Associates the dataset control object with a report being loaded.
SetTableAlias	Sets or changes the Alias for a table in a report.
SetTableLink	Links two tables.
SetUserSQL	Places the data set control object into user entered SQL mode.
TestSelection\$	Returns a string telling how many records would be selected with the current selection criteria.

AddGroup

Adds grouping criteria at the specified level.

Syntax [Object].AddGroup Table\$, DataBase\$, Field\$, Level, Type, [NumRecs]

Parameters

Table\$,

DataBase\$,Field\$ Table\$, DataBase\$ and Field\$ serve to identify the field to be grouped upon.

Level

The level argument specifies the grouping level that you want information about. If an invalid level is specified, a null string is returned and the error\$ property is set to indicate the error.

The level argument specifies the grouping level that you want information about where 0 is the entire report group, 1 is the primary grouping criteria, 2 is the secondary grouping criteria, and so forth.

Valid levels are 1 to 1+ the current number of groups defined

Type

Specifies the type of grouping:

Type Groups on ...

0 Same value

1 Every *n* records (*n* is the NumRecs argument)

2 Daily

3 Monthly

4 Weekly

5 Annually

6 Quarterly

7 Hourly

8 Every minute

9 Every second

10 Second/10

11 Second/100

12 Second/100

Types 2–12 are only valid for date and /or time fields.

NumRecs

When grouping by every *n* records, this specifies how many records per group. This is only used with a Type of 1.

Returns 0 on success, a non-zero on error.

Comments If a group exists at the given level, all groups at that level and higher are adjusted up one level to accommodate the new group.

Example

```
'Group by DEPT_ID, break on same value
MyData.AddGroup "dbo.emp", "hr", "DEPT_ID", 1, 0, 0
```

AddSort

Adds a sorting criteria to the current dataset at the specified level.

Syntax [object].AddSort Table\$, Database\$, Column\$, Ascending, Level

Parameters

Table\$	Defines a string of the form: Owner. TableName or for local databases like dBASE, the Table\$ parameter is the file name of the local data base file.
Database\$	For local databases or servers that don't require that a database be specified, the Database\$ parameter should be set to a null string.
Column\$	The field that is being sorted.
Ascending	Set to zero to sort from smallest to largest, non-zero to sort from largest to smallest.
Level	Indicates its priority among other sorting criteria for this dataset. Valid values for this argument are 1 to the number of current sorting criteria +1.

Returns If an invalid index is specified the function fails and returns an error. This function returns a 0 on success, a non-zero on error.

Comments Valid levels are 1 to 1+ the current number of sorting criteria.

Example MyData.AddSort "dbo.emp", "HR", "DEPT_ID", 1, 1

AddSummary

Adds a summary field to the specified grouping level and index.

Syntax [object].AddSummary Table\$, DataBase\$, Field\$, Level, Type

Parameters

Table\$,	
DataBase\$,Field\$	Identifies the field to be summed.
Level	Specifies the grouping level for creating a summary field. The level argument specifies the grouping level that you want information about where 0 is the entire report group, 1 is the primary grouping criteria, 2 is the secondary grouping criteria, and so forth. Valid values for levels are zero to the number of groups defined.
Type	Type specifies the type of summary: 1 Sum 2 Daily 3 Count 4 Minimum 5 Maximum 6 Average 7 First 8 Last 9 Standard Deviation 10 Variance

Returns Non-zero value for an error and the Error\$ property is set to indicate the error.

Example MyData.AddSummary "dbo.emo", "HR", "SALARY", 1, 1

AddTable

Adds a table to a data set control object.

Syntax [object.]AddTable Table\$,DBase\$

Parameters

Table\$	Defines a string of the form: Owner. TableName or for local databases like dBASE, the Table\$ parameter is the file name of the local database file.
dBASE\$	The dBASE\$ parameter is for data servers that require databases. (For servers that don't require a database, it should be set to a NULL string.)

Note In 2.0 + this argument can now take the path to a local database file or both path and file name can be combined in Table\$ as before.

Returns 0 on success and a non-zero on error.

Comments Before you can add a table to a data set, you must establish a connection.

Example

```
'Add the EMP table from the PUBS database with 'the outer dbo
MyDataSet.AddTable "dbo.emp", "PUBS"
```

Connect

Replaces any previous connection information in a DataSet control with the supplied connection information.

Syntax [object.] Connect Type, Server\$, UserId\$, Pswrd\$, DBase\$

Parameters

Type The Type parameter can take on the following values

- 0,1 Reserved
- 2 DBASE
- 3 EXCEL
- 4 PARADOX
- 5 Reserved
- 6 SQLSRVR
- 7 ORACLE
- 8 DB2
- 9 Reserved
- 10 SYBASE
- 11 BTRIEVE
- 12 SQL Base
- 13 INGRES
- 14 DB2/GUPTA
- 15 Reserved
- 16 TERADATA

Server\$ Name of the data server or local data file name.

UserId\$ Name of the user to make the connection for connections that require a user ID.

Pswrd\$ The user password.

dBASE\$ The database name for connections that require a database. (Null for Oracle.)

Returns 0 on success. If it is not zero, the Error\$ property contains text that describes the error.

Comments Note that for Oracle and other connections without databases, the dBASE\$ should be set to a Null String.

Using the Connect method clears any previously defined tables or table columns, sorting and grouping information, and so forth.

Example `ErrorCode=MyDataSet.Connect (6, "SQLSRVR", "John_Doe", "PW", "")`

Disconnect

Allows you to remove a connection that was previously set with the Connect method.

Syntax `[object].Disconnect`

Returns Error if an active report is using the connection. The return code for this function should be 0 on success. If it is not zero, then the `Error$` property contains text that describes the error.

Comments This function executes successfully only if there are no other DataSet controls or reports using the same connection.

This command returns an error if any other report or active DataSet object is using the connection that this object is trying to disconnect.

Example The following example shows you how to:

- Create a DataSet control
- Add a table
- Create a report
- Print a report
- Close a report
- Disconnect the connection

In the following example, we are assuming that we have a server called `X:ORASRV`, a user called `SCOTT` with a password of `TIGER`.

```
Sub MakeAReport ()
    Dim NewData as DataSet
    NewData.Connect 7, "X:ORASRV", "SCOTT", "TIGER", " "
    'Add a Table (DataBase is NULL for Oracle)
    NewData.AddTable "SCOTT.DEPARTMENT", " "
    'Create the Report Object
    NewData.Commit
    PrintReport 0,0,"",""
    CloseReport 0
    NewData.Disconnect
End Sub
```

Field\$

Returns the value of the specified data field for the current data set record. You can set the current record with the data set record property.

Syntax `[object].Field$(FieldName$)`

Parameters

FieldName\$ The name of the field to return data from.

Returns The value of the specified data field for the current dataset record.

Comments Before data can be retrieved from a data set object a connection must be made; links must be set and a commit or recalc must be successfully performed.

Example

```
MyData.record = 5
Salary = val(MyData.Field$("SALARY"))
```

GetAllFields\$

Lists all fields that are available in the given table.

Syntax [object].GetAllFields\$(Table\$,DBase\$)

Parameters

Table\$ The Table\$ parameter is the path and file name for local databases.

dBASE\$ The database that contains the table for connections that have databases.

Comments A connection must first be made and the table added to the dataset before its field list can be retrieved.

For database servers Table\$ takes the form:

```
Owner.TableName
```

For local databases or servers that don't require that a database be specified, the dBASE\$ parameter should be left blank.

See the "GetField\$ function" on page 319 to get an individual field out of the list of fields.

Example AvailableField\$=MyData.GetAllFields\$("dbo.emp", "hr")

GetColumnAlias\$

Gets the alias for the specified field in the specified table.

Syntax [object].GetColumnAlias\$(Table\$, Database\$, Column\$)

Parameters

Table\$	Defines a string of the form: Owner. TableName or for Local DataBases like dBASE, this parameter is the file name of the local database file.
Database\$	The name of the database that contains the table for connections that have databases.
Column\$	The column for which to set an alias.

Returns The alias for the specified field in the specified table.

Comments For database servers Table\$ takes the form:

```
Owner.TableName
```

For local databases or servers that don't require that a database be specified, the Database\$ parameter should be set to a null string.

Example

```
ColumnAlias$=MyData.GetColumnAlias$("dbo.emp",
"hr", "DEPT_ID")
```

GetDataSources\$

Lists all of the data sources available to ReportSmith, including ODBC Sources, separated by commas.

Syntax [objet.]GetDataSources\$

Returns Returns a list of all of the data sources available to ReportSmith.

Example DataSourcesAvailable\$=MyData.GetDataSources\$

GetFieldList\$

Lists all fields that have been included in the given table.

Syntax [object.]GetFieldList\$(Table\$,DBase\$)

Parameters

Table\$	Defines a string of the form: Owner. TableName or for Local DataBases like dBASE and Excel, the Table\$ parameter is the File Name of the local database file.
dBASE\$	Contains the table for connections that have databases.

Returns A list of all fields that have been included in the given table.

Comments A connection must be made first and the table added to the data set before its field list can be retrieved.

For database servers, Table\$ takes the form:

```
Owner.TableName.
```

For local databases or servers that don't require that a database be specified, the dBASE\$ parameter should be left blank.

Example `IncludedField$=MyData.GetFieldList$("dbo.emp", "hr")`

GetGroup\$

Gets a string that provides information about grouping at the specified level.

Syntax `[object].GetGroup$(Level)`

Parameters

Level Specifies the grouping level that you want information about where 0 is the entire report group, 1 is the primary grouping criteria, 2 is the secondary grouping criteria, and so forth.

Returns A string that provides information about grouping at the specified level.

Comments If an invalid index is specified, a null string is returned and the Error\$ property is set to indicate the error. Valid values for levels are zero to the number of groups defined.

Example `PrimaryGroup$=MyData.GetGroup(1)`

GetSort\$

Gets a string that indicates the sorting criteria at the given level if one exists.

Syntax `[object].GetSort$(Level)`

Parameters

Level Specifies the sorting level that you want information about where 1 is the primary sorting criteria, 2 is the secondary sorting criteria, and so forth.

Returns A string that indicates the sorting criteria at the given level if one exists.

Comments Valid values for the level argument are 1 to the number of current sorting criteria. If an invalid index is specified, a null string is returned and the Error\$ property is set to "Invalid Index."

Example `PrimarySort$=MyData.GetSort$(1)`

GetSummary\$

Gets a string that provides information about a summary field at the specified grouping level and index.

Syntax [object].GetSummary\$(Level, Index)

Parameters

Level Specifies the grouping level that you want information about where 0 is the entire report group, 1 is the primary grouping criteria, 2 is the secondary grouping criteria, and so forth.

Index Matches the order in which the tables are originally added.

Returns If an invalid index or level is specified a null string is returned and the Error\$ property is set to indicate the error.

Example SecondSummary\$=MyData.GetSummary(1,2)

GetSQL\$

Gets the last SQL statement that was executed for this data set control object.

Syntax [object].GetSQL\$

Returns The last SQL statement that was executed for this data set control object.

Example MySQL\$=MyData.GetSQL\$

GetTable\$

Gets a string that describes the table at the specified index if possible.

Syntax [object].GetTable\$(Index)

Parameters

Index Matches the order in which the tables are originally added.

Returns A string that describes the table at the specified index if possible.

Comments If an invalid index is given, this function returns a null string and the Error\$ property is set to an appropriate error message.

Example SecondTable\$=MyData.GetTable\$(2)

GetTableAlias\$

Gets the alias for the specified table.

Syntax [object].GetTableAlias\$(Table\$, DBase\$)

Parameters

Table\$ Defines a string of the form: Owner. TableName or for local data bases like dBASE, this parameter is the file name of the local database file.

dbase\$ Contains the table for connections that have databases.

Returns The alias for the specified table.

Comments For local databases or servers that don't require that a databases be specified, the dBASE\$ parameter should be set to a null string.

Example TableAlias\$=MyData.GetTableAlias\$("dbo.emp", "hr")

GetTableLink\$

Gets a string that provides information about the table link at the given index if one exists.

Syntax [object].GetTableLink\$(Index)

Parameters

Index The link for which to retrieve information.

Returns A string that provides information about the table link at the given index if one exists.

Comments If an invalid index is specified, a null string is returned and the Error\$ property is set to indicate the error.

Example SecondTableLink\$ = MyData.GetTableLink(2)

IncludeFields\$

Adds columns from a table to your data set.

Syntax [object.] IncludeFields\$,Table\$, DBase\$, IncludeList\$

Parameters

Table\$	Defines a string of the form: Owner. TableName or for local databases like dBASE, this parameter is the file name of the local database file.
dBASE\$	Contains the table for connections that have databases.
IncludeList\$	A list of fields in a table to include as part of a table.

Comments The field list should be one string with the names of the included fields separated by commas. The names should be provided exactly as they appear in our dialog boxes. It is case-sensitive.

Example `MyData.IncludeFields "dbo.emp","Pubs","First_Name, Last_Name, Dept, Emp_Id"`

Load

Replaces any previous connection information in a DataSet control with information from the file that is specified by the Filename\$ parameter.

Syntax [object].Load Filename\$

Parameters

Filename\$	The name of a file that the data set control object should be read from.
------------	--

Returns 0 on success. If it is not zero, then the Error\$ property contains text that describes the error.

Comments The file must be created with the Save method and can have any extension.

Example `MyData.Load("c:\RPTSMITH\MyData.DSC")`

Recalc

Re-executes the SQL for this DataSet Control Object.

Syntax [object].Recalc

Comments Because this command is associated with a dataset control, any changes to the result are not reflected in associated reports until a report level recalc is performed. This command is generally used with dataset control functions as stand-alone queries that do not have associated reports.

Example `MyData.Recalc`

RemoveGroup

Removes a grouping criteria from a report.

Syntax [object].RemoveGroup Level

Parameters

Level Specifies the grouping level that you want to remove where 0 is the entire report group, 1 is the primary grouping criteria, 2 is the secondary grouping criteria, and so forth.

Returns 0 on success, a non-zero on error.

Comments If an invalid index is specified, a null string is returned and the Error\$ property is set to indicate the error.

Example MyData.RemoveGroup 1

RemoveSort

Removes the sorting criteria at the given level if one exists.

Syntax [object].RemoveSort Level

Parameters

Level Specifies the grouping level that you want to remove where 0 is the entire report group, 1 is the primary grouping criteria, 2 is the secondary grouping criteria, and so forth.

Returns If an invalid index is specified, a 0 on success, a non-zero on error.

Comments Valid values for the level argument are 1 to the number of current sorting criteria.

Example MyData.RemoveSort 1

RemoveSummary

Returns a summary field from a report.

Syntax [object].RemoveSummary Level, Index

Parameters

Level	Specifies the grouping level that you want information about where 0 is the entire report group, 1 is the primary grouping criteria, 2 is the secondary grouping criteria, and so forth.
Index	Matches the order in which the tables were originally added.

Returns If an invalid index is specified a null string is returned and the Error\$ property is set to indicate the error.

Example `MyData.RemoveSummary 1,2`

RemoveTable

Removes the table at the specified index if possible.

Syntax `[object].RemoveTable Index`

Parameters

Index	Matches the order in which the tables were originally added.
-------	--

Returns 0 on success, a non-zero on error.

Comments The table at any given index can be determined using the GetTable function.

Example `MyData.RemoveTable 2`

RemoveTableLink

Removes the table link for the specified index from the dataset control object.

Syntax `[object].RemoveSummary Level, Index`

Parameters

Level	Specifies the grouping level that the summary is operating on where 0 is the entire report group, 1 is the primary grouping criteria, 2 is the secondary grouping criteria, and so forth.
Index	The index of the link for which to retrieve information.

Comments If an invalid index is specified, a null string is returned and the Error\$ property is set to indicate the error.

Example `MyData.RemoveSummary 1,2`

ReplaceTable

Replaces one table in a report with another.

Syntax [object].ReplaceTable Table\$, Database\$, NewTable\$, NewDataBase\$

Parameters

Table\$ Path and file name for local databases.

Database\$ Contains the old table.

NewTable\$ The name of the replacement table.

NewDataBase\$ Contains the replacement table.

Returns 0 on success, a non-zero on error.

Comments Any fields that don't have a direct match in the old table are excluded from the report and those fields on the report surface should be removed or show #ref.

For database servers this method takes the form:

```
Owner.TableName
```

For local databases or servers that don't require that a database be specified, the Database\$ parameter should be set to a null string.

Example MyData.ReplaceTable "dbo.emp", "hr", "dbo.emp2", "new_hr"

Save

Replaces any previous connection information in a DataSet control with information from the file that is specified by the Filename\$ parameter.

Syntax [object].Save Filename\$

Parameters

Filename\$ The file to save to.

Returns 0 on success. If it is not zero, then the Error\$ property contains text that describes the error.

Example MyData.Save "c:\MyData.Dat"

SetColumnAlias

Sets or changes the Alias for a column in a report's table.

Syntax [object].SetColumnAlias Table\$, Databases\$, Column\$, Alias\$

Parameters

Table\$	The path and file name for local databases.
Database\$	The database that contains the table.
Column\$	The name of the field for which to set an alias.
Alias\$	The new alias for the field.

Returns 0 on success, a non-zero on error.

Comments For database servers the Table\$ parameter takes the form:

```
Owner.TableName.
```

For local databases or servers that don't require that a database be specified the Database\$ parameter should be set to a null string.

Example `MyData.SetColumnAlias "dbo.emp", "hr", "DEPT_ID", "Departments"`

SetFromActive

Replaces any previous connection information in a DataSet control with a reference to the data description for the currently active report.

Syntax [object].SetFromActive

Comments Use this method to change a currently active report. It can also be used to save DataSet information for a report that can then be reloaded, changed and used to create other reports.

Example The following example uses the SetFromActive method along with the Selection\$ property to change the selection criteria for the active report.

```
Sub ChangeActiveSelection()
    'Create a DataSet named DS
    dim DS as DataSet
    DS.SetFromActive
    DS.Selection$ = " Department = 'Accounting' "
    'Cause the report to update to reflect the change
    Recalc
End Sub
```

SetFromLoading

Associates the dataset control object with a report that is being loaded (before the SQL is executed for this report).

Syntax [object].SetFromLoading

Returns 0 on success, a non-zero on error.

Comments This function is only valid when used in the “Before report is opened” event.

Example `MyData.SetFromLoading`

SetTableAlias

Sets or changes the alias for a table in a report.

Syntax `[object].SetTableAlias Table$, Database$, Alias$`

Parameters

Table\$ The path and file name for local databases.

Database\$ The database that contains the table.

Alias\$ The new alias for the table.

Returns 0 on success, a non-zero on error.

Comments For database servers the Table\$ parameter takes the form:

`Owner.TableName.`

For local databases or servers that don’t require that a database be specified the Database\$ parameter should be set to a null string.

Example `MyDate.SetTableAlias "dbo.emp", "hr", "Human_Resource"`

SetTableLink

Defines a link between two tables.

Syntax `[object].SetTableLink Table1$,DBase1$,Field1$,Table2$,DBase2$,Field2$,Operation, JoinType`

Parameters

Table1\$	The first table to link.
DBase1\$	The database that contains Table1.
Field1\$	The field to link on from the first table.
Table2\$	The second table to link.
DBase2\$	The database that contains Table2.
Field2\$	The field to link on from the second table.
Operation	The relation between the linked fields. It can have one of the following values: 0 Field 1 = Field 2 1 Field 1 < Field 2 2 Field 1 <= Field 2 3 Field 1 > Field 2 4 Field 1 >= Field 2
JoinType	The type of link. It can have one of the following values: 0 Inner join 1 Left outer join 2 Right outer join 3 Full outer join

Comments Before a table link can be defined, both tables must be added to the DataSet control using the AddTable function.

Example This example links the emp table to the dept table by the department id excluding all unmatched records.

"The following needs to be one line in Basic"

```
SetTableLink "dbo.emp", "Indigo", "Dept_Id", "dbo.dept", "Indigo", "Dept_Id", 0, 0
```

SetUserSQL

Places the DataSet control object into user entered SQL mode with the provided SQL.

Syntax [object].SetUserSQL SQL\$

Parameters

SQL\$ The complete SQL string to be used for this dataset query.

Returns 0 on success, a non-zero on error.

Comments You must have a connection to the appropriate server (or local database) that supports the SQL you generate in order for the SetUserSQL to work properly.

Example

```
MyData.SetUsersSQL"SELECT dbo.emp.First_Name,
dbo.emp.Last_Name FROM dbo.emp"
```

TestSelection\$

Returns a string that tells how many records would be selected or an error message about the selection criteria.

Syntax [object].TestSelection\$

Comments In order to set and test a selection criteria, you must have a connection and at least one table.

Example

```
MyData.Selection$="Salary>40000"
Msgbox MyData.TestSelection$()
```

DataSet Control properties summary

Properties are variables that belong to an object. You can access these variables in the same way as you access object methods by using the object name followed by a period (.) and the property name. You use this just like any other variable. Some properties are read-only while others can be both read and written.

Property	Description
AllDataBases\$	Returns all the databases available under the current connection.
AllOwners\$	Returns all owners available under the current connection.
AllTables\$	Returns a list of all owners for connections that have owners.
DataBases\$	Returns the current database for the current connection.
Error\$	Contains a string describing the error that occurred in the last dataset control method executed.
Id	Used to store an integer value.
Link Macro	Links a macro in an active list to the specified Object, Event, and Item.
Name\$	Returns the current name for the current situation.
Owner\$	Returns the current owner for the current situation.
Record	Returns the current record in the dataset.
RecordCount	Returns the total number of records in the dataset.
Selection\$	Gets or sets the selection criteria for a dataset control object.
Table\$	Returns a list of tables included in a report.

AllDataBases\$

Returns all the databases available under the current connection, separated by commas.

Syntax [object].AllDataBases\$

Comments A connection must be made before the list of all databases can be retrieved from a DataSet object.

Example ListofDataBases\$=AllDataBases\$

AllOwners\$

Returns all owners available under the current connection, separated by commas.

Syntax [object].AllOwners\$

Comments Before the list of all owners can be retrieved from a DataSet object, a connection must be made.

Example OwnerList\$ = MyData.AllOwners\$

AllTables\$

Returns a list of all tables, separated by commas, for a connection.

Syntax [object].Selection\$ [=stringexpression]

Comments The table in the string is separated by commas and access to individual tables can be achieved by using the “GetField\$ function” on page 319.

Example MsgBox " All Tables: " + MyData.AllTables\$

DataBase\$

Returns the current database for the current connection.

Syntax [object].DataBase\$

Comments Before a current database can be retrieved from a DataSet object, a connection that has databases must be made.

Example CurrentDataBase\$=MyData.DataBase\$

Error\$

Contains a string that describes the error that occurred in the last dataset control method executed.

Syntax [object].Selection\$ [=stringexpression]

Comments Use MyData.PS or any other DataSet name. Dataset names must be used consistently.

Example

```
X=DS.ADDTABLE("INVALID.TABLE","BOGUS")
If x<>0 then msgbox DS.ERROR$
End If
```

Id

Stores an integer value.

Syntax [object].Id = [integerexpression]

Comments The Id property function can be used to keep track of some information related to the DataSet control. As with the Name\$ property this value has no meaning to ReportBasic and it is completely up to the programmer how it is to be used. This property is read/write at run time.

Example MyData.Id=127

Name\$

Returns the current name for the current situation.

Syntax [object].Name\$ = [stringexpression]

Comments This property is read/write at run time.

Example MyData.Name\$="Susan's Data"

Owner\$

Returns the current owner for the current situation.

Syntax [object].Owner\$

Comments Before the current owner can be retrieved from a DataSet object, a connection that has owners must be made.

Example CurrentOwner\$=MyData.Owner\$

Record

Returns the current record in the dataset.

Syntax [object].Record

Comments Before the number of records can be retrieved from a DataSet object a connection must be made, links must be set, and a commit or recalc must be performed successfully.

Example `If MyData.Record = 1 then MsgBox "We are at the beginning"`

RecordCount

Returns the total number of records in the dataset.

Syntax `[object].RecordCount$`

Comments Before the number of records can be retrieved from a DataSet object a connection must be made, links must be set, and a commit or recalc must be successfully performed.

Example `TotalRecords=MyData.RecordCount$`

Selection\$

Gets or sets the selection criteria for a DataSet Control Object.

Syntax `[object.]Selection$`

Comments A table must be added to the data set before the selection criteria can be written or read. A change in Selection\$ does not change the data until a commit method or a recalc command is executed. This property is read/write at run time.

You can get the individual tables from the list by using the GetField\$ function.

Example `MyData.Selection$="Salary>40000"`

Table\$

Returns a list of tables included in a report, separated by commas.

Syntax `[object].Table$`

Comments By using the GetField\$ function you can get the individual tables from the list.

Example `SecondTable$=GetField$(MyData.Table$,2,"")`

Report Control Object

Controls reports in the same manner that the DataSet Control Object is used to control queries. At this time, the Report Control has only a limited number of methods and properties. New methods and properties will be added in future versions of ReportSmith.

A report control object is created in the same manner as a dataset control.

Syntax A `Dim MyReport as Report`

Syntax B Global MyGlobalReport as Report**Report Object properties (members)**

Name\$	A string that identifies this object. Null by default
Id	A number that identifies this object. Null by default
Page	The current page of the report
TotalPages	Returns the total number of pages in the report (read only)
Error\$	Last error returned by a method of this object

Report Object methods

SetFromActive	Associates a Report Control Object with the currently active report.
Recalc	Just like the general Recalc command, but only affects the associated report regardless of what report is active.

Basic commands summary

The following table contains a list of available Help topics for standard Basic functions and statements. Index items are arranged in alphabetical order within each major category. You can also view these commands in ReportSmith's online Help.

Note For information on how to use Help, press *F1* or choose Using Help from the Help menu.

Command	Description
Abs	Return the absolute value of a number.
Asc	Return an integer corresponding to a character code.
Assert	Trigger an error if a condition is false.
Atn	Return the arctangent of a number.
Beep	Produce a short beeping tone through the speaker.
Begin Dialog... End Dialog	Begin a dialog box definition.
Button	Define a button dialog box control.
ButtonGroup	Begin definition of a group of button dialog box controls.
Call	Transfer control to a subprogram.
CancelButton	Define a cancel-button dialog box control.
Caption	Define the title of a dialog box.
CDbl	Convert a number to double-precision floating point.
ChDir	Change the default directory for a drive.
ChDrive	Change the default drive.
CheckBox	Define a checkbox dialog box control.
Chr\$	Return a string containing a single character that corresponds to a character code.
CInt	Convert a number to an integer by rounding.

Command	Description
CLng	Convert a number to a long by rounding.
Close	Close a file.
ComboBox	Define a ComboBox dialog box control.
Command\$	Return the command line specified when the INIT or MAIN sub was run.
Const	Declare a symbolic constant.
Cos	Return the cosine of an angle.
CSng	Convert a number to single-precision floating point.
'\$CStrings	Treat backslash in string as escape character in the same way that the 'C' language does.
CurDir\$	Return the current directory for a drive.
Date\$	Return the current date.
Declare	Forward declare a procedure in the same module or in a dynamic link library.
Deftype	Declare the default data type for variables.
Dialog	Display a dialog box.
Dim	Declare variables for use in a method.
Dir\$	Return a filename which matches a pattern.
Do...Loop	Control repetitive actions.
Do...While	Repeat a block of statements while or until a condition is true.
Environ\$	Return a string from the operating system's environment.
Eof	Check for end of file.
Erl	Return the line number where a runtime error occurred.
Err function	Return a runtime error code.
Err statement	Set the runtime error code.
Error	Generate an error condition.
Error\$ function	Returns a string representing an error.
Exit	Cause the current procedure or loop structure to return.
Exp	Return the value of e raised to a power.
FileAttr	Return information about an open file.
Fix	Return the integer part of a number.
For...Next	Loop a fixed number of times.
FreeFile	Return the next unused file number.
Function ... End Function statement	Return a function.
GetCurValues	Retrieve current values for a dialog box.
GetField\$	Return a substring from a delimited source string.
Global	Declare a global variable.
GoTo	Send control to a line label.
GroupBox	Return a GroupBox in a dialog box.
Hex\$	Return the hexadecimal representation of a number, as a string.
If ... Then ... Else	Branch on a conditional value.
'\$Include	Tell the compiler to include statements from another file.
Input\$ function	Return a string of characters from a file.
Input # statement	Reads data from a file or from the keyboard.
InputBox\$ function	Display a dialog box which prompts for input.

Command	Description
InStr	Return the position of one string within another.
Int	Return the integer part of a number.
Kill	Delete files from a disk.
LBound	Return the lower bound of an array's dimension.
LCase\$	Convert a string to lower case.
Left\$	Return the left portion of a string.
Len	Return the length of a string.
Let (assignment statement)	Assign a numeric or string value to a variable.
Line Input #	Read a line from a sequential file.
ListBox statement	Return a Listbox dialog box control.
Lof	Return the length of an open file.
Log	Return the natural logarithm of a value.
LTrim\$	Remove leading spaces in a string.
Mid\$ statement	Replace a portion of a string with another string.
Mid\$ function	Return a portion of a string.
MkDir	Make a directory on a disk.
MsgBox function	Display a Windows message box.
MsgBox statement	Display a Windows message box.
Name	Rename a disk file.
'\$NoCStrings	Tell the compiler to treat a backslash as normal character.
Oct\$	Return the octal representation of a number, as a string.
OkButton	Return an OK button dialog box control.
On Error	Control runtime error handling.
Open	Open a disk file or device for I/O.
Option Base	Declare the default lower bound for array dimensions.
OptionButton	Return a OptionButton dialog box control.
OptionGroup	Begin definition of a group of option button dialog box controls.
Print	Print data to a file or to the screen.
Randomize	Initialize the random-number generator.
ReDim	Declare dynamic arrays and reallocate memory.
Rem	Treat the remainder of the line as a comment.
Reset	Close all open disk files.
Resume	End an error-handling routine.
Right\$	Return the right portion of a string.
Rmdir	Remove a directory from a disk.
Rnd	Return a random number.
RTrim\$	Remove trailing spaces in a string.
Seek	Return the current position for a file.
Seek	Set the current position for a file.
Select Case	Execute one of a series of statement blocks.
SetField\$	Replace a substring within a delimited target string.
Sgn	Return a value indicating the sign of a number.
Shell	Run an executable program.

Command	Description
Sin	Return the sine of an angle.
Space\$	Return a string of spaces.
Sqr	Return the square root of a number.
Stop	Stop program execution.
Str\$	Return the string representation of a number.
String\$	Return a string consisting of a repeated character.
Sub ... End Sub	Return a subprogram.
Tan	Return the tangent of an angle.
Text	Return a line of text in a dialog box.
TextBox	Return a line of TextBox in a dialog box.
Time\$	Return the current time.
Timer	Return the number of seconds since midnight.
Type	Declare a user-defined data type.
UBound	Return the upper bound of an array's dimension.
UCase\$	Convert a string to upper case.
Val	Convert a string to a number.
While ... Wend	Control repetitive actions.
Write	Write data to a sequential file.

Abs function

Returns the absolute value of a specified numeric expression.

Syntax Abs(numeric_expression)

Parameters

numeric_expression A field or variable representing a number.

Returns Matches the type of numeric expression. This includes variant expressions that return a result of the same vartype as input, except vartype 8 (string) is returned as vartype 5 (double) and vartype 0 (empty) is returned as vartype 3 (long).

Example Value1 and Value2 below are integer variables.

```
Difference=ABS(Value1-Value2)
```

Asc function

Converts the first character in a string from character code to an ASCII code number.

Syntax Asc(string_expression\$)

Parameters

`string_expression$` The string from which to get the first character.

Returns An integer corresponding to the ANSI code of the first character in the specified numeric expression. The return value is single-precision for an integer, currency or single-precision numeric expression, double-precision for a long, variant, or double-precision numeric expression.

Comments Generates an error if the argument is null.

Example `MsgBox "The character "%" has the ASCII code:" +Str$(ASC("%"))`

Assert statement

Declares a condition that must be true for continued macro execution.

Syntax `Assert condition`

Parameters

`condition` An expression or condition which you want to assert as true.

Comments Triggers an error if the condition is FALSE. An assertion error cannot be trapped by the ON ERROR statement.

The Assert statement is intended to help ensure that a procedure is performing in the expected manner.

Example `Assert x>0`

Atn function

Calculates the arctangent of a numeric expression that indicates a ratio. This can be used only as a function.

Syntax `Atn( numeric_expression )`

Parameters

`numeric_expression` A field or variable representing a number.

Returns Returns the angle (in radians) corresponding to the arctangent of the specified numeric expression.

Comments The return value is single-precision for an integer, currency or single-precision numeric expression, double-precision for a long, variant or double-precision numeric expression.

Example `x=Atn(0.33)`

Beep statement

The Beep statement produces a short beeping tone that can be used to alert a user.

Syntax `Beep`

Comments The Beep statement produces a single short beeping tone through the computer speaker.

Example

```
If Value>Limit then
  Beep
  MsgBox "The value was greater than allowed"
End If
```

Begin Dialog ... End Dialog statement

Starts the dialog box declaration for a user-defined dialog box.

Syntax

```
Begin Dialog dialogName [x, y,] dx, dy
[dialog box definition statements]
End Dialog
```

Parameters

`[x, y,]` The x and y arguments give the coordinates that position the dialog box. These coordinates designate the position of the upper left corner of the dialog box, relative to the upper left corner of the client area of the parent window. The x argument is measured in units that are 1/4 the average width of the system font. The y argument is measured in units 1/8 the height of the system font. (For example, to position a dialog box 20 characters in, and 15 characters down from the upper left hand corner, enter 80, 120 as the x, y coordinates.) If these arguments are omitted, the dialog box is centered in the client area of the parent window.

`dx, dy` The dx and dy arguments specify the width and height of the dialog box (relative to the x and y coordinates). The dx argument is measured in 1/4 system-font character-width units. The dy argument is measured in 1/8 system-font character-width units (i.e., to create a dialog box 80 characters wide, and 15 characters in height, enter 320, 120 as the dx, dy coordinates).

Comments The Begin Dialog statement assumes that if only two arguments are given, they are the dx (width) and dy (height) arguments. Unless the Begin Dialog statement is

followed by at least one other dialog box definition statement and the End Dialog statement, an error results.

The other definition statement must include an OkButton or CancelButton or Button statement. If this statement is left out, there is no way to close the dialog box, and the procedure is unable to continue executing.

To display the dialog box, you create a dialog record variable with the Dim statement, and then display the dialog box using the Dialog statement. In the Dim statement, dialogName is used to identify the dialog definition.

Example

```
Begin Dialog MyDialog 120,130
Caption "This is My Dialog"
End Dialog
```

Button statement

Defines a custom push button. (This allows the use of push buttons other than OK and CANCEL.) It is used in conjunction with the ButtonGroup statement.

Syntax Button x, y, dx, dy, text\$

Parameters

x, y	The x and y arguments set the position of the button relative to the upper left corner of the dialog box.
dx, dy	dx and dy set the width and height of the button. A dy value of 14 typically accommodates text in the system font.
text\$	Contains a message that is contained in the push button. If the width of this string is greater than dx, trailing characters are truncated.

Comments The Button statement can be used only between a Begin Dialog and an End Dialog statement. A dy value of 14 typically accommodates text in the system font.

Example Button 10,20,14,14, "Hello User!"

ButtonGroup statement

Begins definition of the buttons when custom buttons are to be used.

Syntax ButtonGroup .field

Parameters

.field A variable that contains the number of the button in the group that was selected. Buttons are numbered by the order they are entered in the dialog definition.

Comments ButtonGroup establishes the dialog-record field that contains the user's selection. If ButtonGroup is used, it must appear before any Button statement which creates a pushbutton. Only one ButtonGroup statement is allowed within a dialog box definition.

The ButtonGroup statement can be used only between a Begin Dialog and an End Dialog statement.

Example

```
Begin Dialog MyDialog 360,260
ButtonGroup.Pressed
Button 20, 130, 150, 30, "Load Report"
Button 20, 130, 150, 30, "Print Report"
End Dialog

'Execute dialog in this code

MyDialog.Pressed

'This will have the value of 1 if load report is pressed and 2 if print report if pressed.
```

Call statement

Transfers control to a subprogram procedure or application-defined dialog box.

Syntax A Call subprogram_name [(argumentlist)]

Syntax B Subprogram_name argumentlist

Syntax C Call app_dialog (recordName)

Syntax D App_dialog {recordName | dotList}

Parameters The Call statement parameter consists of a subprogram and its arguments.

Comments Used to call a subprogram written in Basic or to call C procedures in a DLL. These C procedures must be described in a Declare statement or be implicit in the application.

The arguments to the subprogram must match the parameters as specified in the definition of the subprogram. The arguments can be either variables or expressions. Arguments are passed by reference to procedures written in Basic. If you pass a variable to a procedure which modifies its corresponding formal parameter, and you do not wish to have your variable modified, enclose the variable in parentheses in the Call statement. This tells Basic to pass a copy of the variable. This is less efficient and should not be done unless necessary.

When a variable is passed to a procedure which expects its argument by reference, the variable must match the exact type of the formal parameter of the function. (This restriction does not apply to expressions.)

Similarly to subprogram invocation, functions associated with application-defined dialog boxes can be invoked using Call syntaxes listed as C and D above. In Syntax C, the name inside the parentheses must be a variable previously Dimmed as an application-defined dialog record. In Syntax D, the dialog box name can be followed by either a dialog record variable or a comma-separated list of dialog box fields settings, for example:

```
SearchFind .SearchFor="abc", .Forward=1
```

When calling an external DLL procedure, arguments can be passed by value rather than by reference. This is specified either in the Declare statement, the Call itself, or both, using the ByVal keyword. If ByVal is specified in the declaration, then the ByVal keyword is optional in the call; if present, it must precede the value. If ByVal was not specified in the declaration, it is illegal in the call unless the datatype specified in the declaration was Any. Specifying ByVal causes the parameter's value to be placed on the stack, rather than a far reference to it.

Example

```
Sub Callable (Argument$)
:
End Sub
Sub Macro
Call Callable "My Argument"
```

CancelButton statement

The CancelButton statement determines the position and size of a cancel button.

Syntax CancelButton x, y, dx, dy

Parameters

<i>x, y</i>	The x and y arguments set the position of the cancel button relative to the upper left corner of the dialog box.
<i>dx, dy</i>	dx and dy set the width and height of the button. A dy value of 14 can usually accommodate text in the system font.

Comments The CancelButton statement can be used only between a Begin Dialog and an End Dialog statement.

The x and y arguments set the position of the cancel button relative to the upper left corner of the dialog box. dx and dy set the width and height of the button. A dy value of 14 can usually accommodate text in the system font.

If the CancelButton is pushed at runtime, the dialog box is removed from the screen and an Error 102 is triggered.

Example `CancelButton 10,20,14,14`

Caption statement

Defines the text to be used as the title of a dialog box.

Syntax `Caption text$`

Parameters

`text$` The text you would like to see appear in your caption.

Comments The Caption statement can be used only between a Begin Dialog and an End Dialog statement.

If no Caption statement is specified for the dialog box, a default caption is used.

Example `Caption "Dialog Title"`

CDBl function

The CDBl function converts an expression to double-precision floating point. This can be used only as a function.

Syntax `CDBl ( numeric_expression )`

Parameters

`numeric_expression` A field or variable representing a number.

Returns Converts an expression to a double-precision floating point.

Comments CDBl accepts any type of expression. Strings that cannot be converted to a currency results in a "Type Mismatch" error. Variants containing nulls result in an "Illegal Use of Null" error.

To convert an expression to a different data type, see CCur, CInt, CLng, CSng, CStr, CDate and CVar.

Example `Cdbl ("Table.Numberfield1"+"Table.Numberfield2")`

ChDir statement

Changes the default directory for the specified drive.

Syntax `ChDir pathname$`

Parameters

pathname\$ Pathname\$ is a string expression identifying the new default directory.

Comments The ChDir statement changes the default directory for the specified drive. It does not change the default drive. (To change the default drive, use ChDrive).

The syntax for pathname\$ is:

```
[drive:] [\] directory [\directory]
```

The drive argument is optional. If omitted, ChDir changes the default directory on the current drive.

Example ChDir "c:\rptsmith\macros"

ChDrive statement

Changes the default drive.

Syntax ChDrive drivename\$

Parameters

drivename\$ Drivename\$ is a string expression designating the new default drive. This drive must exist, and must be within the range specified in the CONFIG.SYS file.

Comments If a null argument ("") is supplied, the default drive remains the same. If the drivename\$ argument is a string, ChDrive uses the first letter only. If the argument is omitted, an error message is produced. (To change the current directory on a drive, use ChDir.)

Example ChDrive "g:"

CheckBox statement

The CheckBox statement places a checkbox within a dialog box.

Syntax CheckBox x, y, dx, dy, text\$, .field

Parameters

<code>x, y</code>	The x and y arguments give the coordinates that position the check box. These coordinates designate the position of the upper left corner of the check box, relative to the upper left corner of the dialog box. The x argument is measured in 1/4 system-font character-width units. The y argument is measured in 1/8 system-font character-height units. (See “Begin Dialog ... End Dialog statement” on page 293.)
<code>dx</code>	The dx argument is the combined width of the check box and the text\$ field. Because proportional spacing is used, the width varies with the characters used. To approximate the width, multiply the number of characters in the text\$ field (including blanks and punctuation) by 4 and add 12 for the checkbox.
<code>dy</code>	The dy argument is the height of the text\$ field. A dy value of 12 is standard, and should cover typical default fonts. If larger fonts are used, the value should be increased. As the dy number grows, the checkbox and the accompanying text moves downward within the dialog box.
<code>text\$</code>	The text\$ field contains the title shown to the right of the check box. If the width of this string is greater than dx, trailing characters are truncated. If you wish to include underlined characters so that the check box selection can be made from the keyboard, the character must be preceded with an ampersand (&).
<code>.field</code>	The .field argument is the name of the dialog-record field that holds the current check box setting. If its value is 0, the box is unchecked; if its value is -1 the box appears grayed; if its value is 1, the box is checked. The macro language treats any other value of .field the same as a 1.

Comments The CheckBox statement can be used only between a Begin Dialog and an End Dialog statement.

Example `CheckBox 10,10,14,14,"My statement", .check_value-holder`

CInt function

Converts the value of expression to an integer by rounding.

Syntax `CInt ( numeric_expression )`

Parameters

(numeric_expression) The argument given is any numeric-expression.

Comments CInt accepts any type of expression. After rounding, the resulting number must be within the range of -32767 to 32767, or an error occurs.

Strings that cannot be converted to an integer results in a “Type Mismatch” error. Variants containing nulls results in an “Illegal Use of Null” error. To convert a numeric

expression to a different data type, see the “CDBl function” on page 297, “CLng function” on page 300 and “CSng function” on page 302.

Example `CLnt("table.Decimalfield")`

CLng function

The CLng function converts the value of expression to a long by rounding.

Syntax `CLng (numeric_expression)`

Parameters

(numeric_expression) The argument given is any numeric-expression.

Returns A round long number.

Comments After rounding, the resulting number must be within the range of: -2,147,483,648 to 2,147,483,647, or an error occurs.

Strings that cannot be converted to a long result in a “Type Mismatch” error. Variants containing nulls result in an “Illegal Use of Null” error.

CLng generates the same result as you would get by assigning the numeric-expression to a Long variable. To convert a value to a different data type, see the “CDBl function” on page 297, “CInt function” on page 299 and “CSng function” on page 302.

Example `CLng("Table.Numfield"*10000)`

Close statement

Closes a file, concluding input/output to that file.

Syntax `Close [ [#] filename% [ , [ # ] filename% ... ]]`

Parameters

filename% Filename% is an integer expression identifying the file to close. It is the number used in the Open statement for the file. If this argument is omitted, all open files are closed.

Comments Once a Close statement is executed, the association of a file with filename% is ended, and the file can be reopened with the same or different file number.

When the Close statement is used, the final output buffer is written to the operating system buffer for that file. Close frees all buffer space associated with the closed file. Use the Reset statement so that the operating system flushes its buffers to disk.

Example

```
'Close file number 1
close number 1
```

ComboBox statement

The ComboBox statement is used to create a combination text box and list box.

Syntax `ComboBox x, y, dx, dy, text$, .field`

Parameters

<code>x,y</code>	The x and y arguments give the coordinates that position the upper left corner of the list box, relative to the upper left corner of the dialog box. The x argument is measured in 1/4 system-font character-width units. The y argument is measured in 1/8 system-font character-width units. (See “Begin Dialog ... End Dialog statement” on page 293.)
<code>dx,dy</code>	The dx and dy arguments specify the width and height of the combo box in which the user enters or selects text.
<code>text\$</code>	The text\$ field specifies the name of the string containing the list variables.
<code>.field</code>	The .field argument is the name of the dialog-record field that holds the text string entered in the text box or chosen from the list box. The string in the text box is recorded in the field designated by the .field argument when the OK button (or any pushbutton other than CANCEL) is pushed.

Comments The ComboBox statement can be used only between a Begin Dialog and an End Dialog statement.

Example `ComboBox 10,10,14,14,"My Statement",.text_entered_holder`

Const statement

You use the Const statement to declare symbolic constants for use in a Basic program.

Syntax `[Global] Const constantName = expression [,constantName = expression ]...`

Parameters

<code>constantName</code>	The name of the constant being defined.
<code>expression</code>	The value to assign to the constant.

Comments Basic is a strongly typed language. The available data types for constants are numbers and strings.

The type of the constant can be specified by using a type character as a suffix to the constantName. If no type character is specified, the type of the constantName is derived from the type of the expression.

If Global is specified, the constant is validated at module load time; if the constant has already been added to the runtime global area, the constant's type and value are compared to the previous definition, and the load fails if a mismatch is found. This is useful as a mechanism for detecting version mismatches between modules.

Example `Const True=1, False=0, User$="John Doe", Pi=3.1415`

Cos function

Declares symbolic constants for use in a Basic program.

Syntax `Cos( angle )`

Parameters

`angle` A variable, field, or number representing an angle.

Returns The Cos function returns the cosine of an angle. The return value is between -1 and 1. The return value is single-precision if the angle is an integer or single-precision value, double precision for a long or double-precision value.

Comments The angle is specified in radians, and can be either positive or negative.

Example

```
'Calculate the cos of 450
Pi radians = 1800
The_cos = cos(45x3.1415/180)
```

Note The mathematical equation in parentheses represents the conversion factor.

CSng function

The CSng function converts the value of expression to a single-precision floating point.

Syntax `CSng ( numeric_expression )`

Parameters

`numeric_expression` The argument given is any numeric expression.

Returns A numeric expression.

Comments Accepts any type of expression. The numeric-expression must have a value within the range allowed for the Single data type, or an error occurs.

Strings that cannot be converted to an integer results in a “Type Mismatch” error. Variants containing nulls results in an “Illegal Use of Null” error.

To convert a numeric expression to a different data type, see the “CDbl function” on page 297, “CInt function” on page 299 and “CLng function” on page 300.

Example `S=Csng(1/3)`

'\$CStrings metacommand

The '\$CStrings metacommand tells the compiler to treat a backslash character inside a string (\) as an escape character. This treatment is based on the 'C' language.

Syntax '\$CSTRINGS

Comments The supported special characters are:

- | | |
|------------------------------|-------------------------------|
| • Newline (Linefeed) \n | • Formfeed \f |
| • Horizontal Tab \t | • Backslash \\ |
| • Vertical Tab \v | • Single Quote \' |
| • Backspace \b | • Double Quote \" |
| • Carriage Return \r | • Null Character \0 |

The instruction “Hello\r World” is the equivalent of “Hello” + Chr\$(13)+“World”.

In addition, any character can be represented as a 3 digit octal code or a 3 digit hexadecimal code:

```
Octal Code\ddd
Hexadecimal Code\xddd
```

For both hexadecimal and octal, fewer than 3 characters can be used to specify the code as long as the subsequent character is not a valid (hex or octal) character.

To tell the compiler to return to the default string processing mode, where the backslash character has no special meaning, use the '\$NoCStrings metacommand.

Example '\$CStrings

CurDir\$ function

This function determines the current directory of the specified drive.

Syntax `CurDir$ [ (drivename$) ]`

Parameters

[(drivename\$)] Drivename\$ is a string expression identifying the drive to return the default directory of. This drive must exist, and must be within the range specified in the CONFIG.SYS file. If a null argument ("") is supplied, or if no drivename is indicated, the path for the default drive is returned.

Returns The path (including the drive letter) that is current default directory for the specified drive. The dollar sign, "\$", in the function name is optional. If specified the return type is string. If omitted the function returns a variant of vartype 8 (string).

Comments To change the current drive, use ChDrive; to change the current directory, use ChDir. command from the list box.

Example

```
'Get the current directory of Drive C
CurrentDir$=CurDir$("C")
```

Date\$ function

Retrieves the system date as a string.

Syntax Date\$

Returns A string representing the current date. The dollar sign, "\$", in the function name is optional. If specified the return type is string. If omitted the function returns a variant of vartype 8 (string).

Comments The Date\$ function returns a ten character string.

Example Today\$=Date\$Declare Statement

Declare statement

The Declare statement has two uses—forward declaration of a procedure whose definition is to be found later in this module, and declaration of a procedure which is to be found in an external Windows DLL or external Basic module.

Syntax A Declare Sub name [libSpecification] [(parameter [As type])]

Syntax B Declare Function name [libSpecification] [(parameter [As type])]

Parameters The parameters are specified as a comma-separated list of parameter names. The data type of a parameter can be specified by using a type character or by using the As clause. Record parameters are declared by using an As clause and a type which has previously been defined using the Type statement.

A forward declaration is needed only when a procedure in the current module is referenced before it is used. In this case, the BasicLib, Lib and Alias clauses are not used.

Array parameters are indicated by using empty parentheses after the parameter. Array dimensions are not specified in the Declare statement.

External DLL procedures are called with the PASCAL calling convention (the actual arguments are pushed on the stack from left to right). By default, the actual arguments are passed by far reference. For external DLL procedures, there are two additional keywords, `ByVal` and `Any`, that can be used in the parameter list.

When `ByVal` is used, it must be specified before the parameter it modifies. When applied to numeric data types, `ByVal` indicates that the parameter is passed by value, not by reference. When applied to string parameters, `ByVal` indicates that the string is passed by far pointer to the string data. By default, strings are passed by far pointer to a string descriptor.

`Any` can be used as a type specification, and permits a call to the procedure to pass a value of any datatype. When `Any` is used, type checking on the actual argument used in calls to the procedure is disabled (although other arguments not declared as type `Any` are fully type-safe). The actual argument is passed by far reference, unless `ByVal` is specified, in which case the actual value is placed on the stack (or a pointer to the string in the case of string data). `ByVal` can also be used in the call. It is the external DLL procedure's responsibility to determine the type and size of the passed-in value.

Returns A Sub procedure does not return a value. Function returns a value, and can be used in an expression. Function names must end with a type character.

This specifies the return value of the function. The name argument names the Sub or Function being declared.

Comments If the `libSpecification` is of the format:

```
BasicLib libName
```

the procedure is to be found in another Basic module named `libName`. In this case, the other module is loaded on demand whenever the procedure is called. The macro language will not automatically unload modules that are loaded in this fashion. The macro language detects errors of mis-declaration with very high (but not perfect) reliability.

If the `libSpecification` is of the format:

```
Lib libName [ Alias ordinal ]
```

the procedure is to be found in a Dynamic Link Library (DLL) named `libName`. The ordinal argument specifies the ordinal number of the procedure within the external DLL. If the ordinal is not specified, the DLL function is accessed by name, which can cause the module to load more slowly. It is recommended that the ordinal be used whenever possible.

The Softbridge language supports two different behaviors when an empty string ("") is passed `ByVal` to an external procedure. The implementor of the macro language can specify which behavior by using the macro language API function `SetInstanceFlags`. In any specific implementation which uses the macro language, one of these two behaviors should be used consistently. We recommend the second behavior, which is compatible with Microsoft's VB Language. The following two paragraphs describe the two possible behaviors.

When an empty string ("") is passed ByVal to an external procedure, the external procedure receives a NULL pointer. If you wish to send a valid pointer to an empty string, use Chr\$(0).

When an empty string ("") is passed ByVal to an external procedure, the external procedure receives a valid (non-NULL) pointer to a character of 0. To send a NULL pointer, Declare the procedure argument as ByVal As Any, and call the procedure with an argument of 0&.

Example

```
Declare Function ShowWindow Lib
"User" (ByVal hwnd as Integer,
ByVal hwnd CardShow as Integer)as Integer
```

Deftype statement

Specifies the default data type of a variable specified in varTypeLetters.

Syntax

```
DefInt varTypeLetters
DefLng varTypeLetters
DefSng varTypeLetters
DefDbt varTypeLetters
DefStr varTypeLetters
```

Parameters

varTypeLetters The name of the variable in which you would like to define the data type.

Comments The varTypeLetters are specified as a comma-separated list of letters. A range of letters can also be specified. For example, a–d indicates the letters a, b, c and d.

The case of the letters is not important, even in a letter range. The letter range a–z is treated as a special case. It denotes all alpha characters, including the international characters.

The Deftype statement only affects the module in which it is specified. It must precede any variable definition within the module.

Variables defined using the Global or Dim can override the Deftype statement by using an As clause or a type character.

Example

```
DefInt MyInt1, Another Int, LastInt
DefStr Name, Caption
```

Dialog

Displays a dialog box.

Syntax Dialog DialogName\$

Parameters

DialogName\$ The name of a user defined dialog box created with the dim statement.

Comments The data for the controls of the dialog box comes from the dialog box record recordName.

The dialog box recordName must have been declared using the Dim statement. If the user exits the dialog box by pushing the Cancel button, a runtime error is triggered which can be trapped using On Error.

The Dialog statement does not return until the dialog box is closed.

Example

```
Dim The_Diag as UserDiag
Dialog The_Diag
```

Dim statement

Declares variables for use in a Basic program.

Syntax Dim [Shared] variableName [As type] [,variableName [As type]] ...

Comments Basic is a strongly typed language. The available data types are: numbers, strings, records, arrays, dialog boxes and Application Data Types (ADTs).

If the As clause is not used, the type of the variable can be specified by using a type character as a suffix to the variableName. The two different type-specification methods can be intermixed in a single Dim statement (although not on the same variable).

Names

Variable names must begin with a letter and contain only letters, numbers and underscores. Variable names can also be delimited by brackets, except for other brackets, any character can be used inside the brackets.

```
Dim my_1st_variable As String
Dim [one long and strange! variable name] As String
```

Numbers

Numeric variables can be declared using the As clause and one of the following numeric types: Currency, Integer, Long, Single, Double. Numeric variables can also be declared by including a type character as a suffix to the name.

Strings

Basic supports two types of strings, fixed-length and dynamic. Fixed-length strings are declared with a specific length (between 1 and 32767) and cannot be changed later. Use the following syntax to declare a fixed-length string:

```
Dim variableName As String* length
```

Dynamic strings have no declared length, and can vary in length from 0 to 32767. The initial length for a dynamic string is 0. Use the following syntax to declare a dynamic string:

```
Dim variableName$
```

Or,

```
Dim variableName As String
```

Records

Record variables are declared by using an As clause and a typeName which has previously been defined using the Type statement. The syntax to use is:

```
Dim variableName As typeName
```

Records are made up of a collection of data elements called fields. These fields can be of any numeric, string, variant, or previously-defined record type. See Type for details on accessing fields within a record.

You can also use the Dim statement to declare a dialog record. In this case type is specified as:

```
[ Dialog ] dialogName
```

where dialogName matches a dialog box name previously defined using Begin Dialog. The dialog record variable can then be used in a Dialog statement.

Dialog records have the same behavior as regular records—they differ only in the way they are defined. Some applications may provide a number of pre-defined dialog boxes.

Objects

Object variables are declared by using an As clause and a typeName of a class. Object variables can be Set to refer to an object, and then used to access members and methods of the object using dot notation.

```
Dim Ole2 As Object
Set Ole2 = CreateObject("spoly.cpoly")
Ole2.reset
```

An object may be declared as new for some classes. In such instances, the object variable does not need to be Set , a new object is allocated when the variable is used.

Note The class object does not support the new operator.

```
Dim variableName As New className
variableName.methodName
```

Arrays

The available data types for arrays are: numbers, strings, variants, objects and records. Arrays of arrays, dialog box records, and ADTs are not supported.

Array variables are declared by including a subscript list as part of the variableName. The syntax to use for variableName is:

```
Dim variable( [ subscriptRange, ... ] ) As typeName
```

Or,

```
Dim variable_with_suffix( [ subscriptRange, ... ] )
```

where subscriptRange is of the format:

```
[ startSubscript To ] endSubscript
```

If startSubscript is not specified, 0 is used as the default. The Option Base statement can be used to change the default.

Both the startSubscript and the endSubscript are valid subscripts for the array. The maximum number of subscripts which can be specified in an array definition is 60. The maximum total size for an array is only limited by the amount of memory available.

If no subscriptRange is specified for an array, the array is declared as a dynamic array. In this case, the ReDim statement must be used to specify the dimensions of the array before the array can be used. A variable declared inside of a procedure has scope local to that procedure. A variable declared outside of a procedure has scope local to the module. It is permissible for a procedure to declare a variable with a name that matches a module variable. When this happens, the module variable is not accessible by the procedure.

Variables can be shared across modules. See the Global statement for details.

The Shared keyword is included for backward compatibility with older versions of Basic. It is not allowed in Dim statements inside of a procedure. It has no effect. Basic allows a variable to be automatically declared, without the use of a Dim statement. If a variable is first used with a type character as a suffix to its name, the variable is automatically declared to be a local variable of the specified type. If no type character is specified, the variable is automatically declared to be a local variable of type Variant. It is considered good programming practice to declare all variables, and not make use of this feature. To force all variables to be explicitly declared use the Option Explicit statement. It is also recommended that you place all procedure-level Dim statements at the beginning of the procedure. Regardless of what mechanism you used to declare a variable, you can choose to use or omit the type character when referring to the variable in the rest of your program. The type suffix is not considered part of the variable name.

Example `Dim My_var as Array`

Dir\$ function

Returns a file that matches the given directory and/or wild card. Can be used only as a function.

Syntax Dir\$ [(filespec\$)]

Parameters

- filespec\$** A string expression identifying a path or filename. This argument can also include a drive specification. It may also include “wildcard” characters (?) and (*).
- Attrib%** An integer expression specifying the file names that need to be added to the list. The default value for attrib% is 0.

Returns The Dir\$ function returns a filename that matches the specified pattern. The dollar sign (\$) in the functions name is optional. If specified the return type is string. If omitted the function returns a variant of vartype 8 (string).

Comments This argument can include a drive specification. It can also include the “wildcard” characters '?' and '*'. Dir\$ returns the first filename that matches the filespec\$ argument. To retrieve additional filenames that match the filespec\$, call the Dir\$ function again, omitting the filespec\$ argument. If no file is found, an empty string ("") is returned. For Attrib%, Dir\$ returns only files without directory, hidden, system, or volume label attributes set.

Here are the possible values for attrib%:

Value	Meaning
0	Return normal files
2	Add hidden files
4	Add system files
8	Return volume label
16	Add directories

Example

```
'Count .rpt files in "c:s"
Count=1
A$=Dir$("C:\.rpt")
While Dir$<>""
Count=Count+1
Wend
```

Do...While statement

Repeats a block of statements while a condition is true or until a condition becomes true.

Syntax A

```
Do [ { While | Until } condition]
[ statementblock ]
[ Exit Do ]
```

```
[ statementblock ]
Loop
```

Syntax B

```
Do
[ statementblock ]
[ Exit Do ]
[ statementblock ]
Loop [ { While | Until } condition]
```

Parameters

condition Condition is any expression that Basic can determine to be TRUE (nonzero) or FALSE (0).

Comments Basic repeats the program lines contained in the statementblock(s) as long as a While condition is true or until an Until condition is FALSE.

When an Exit Do statement is executed, control is transferred to the statement which follows the loop statement. When used within a nested loop, an Exit Do statement moves control out of the immediately enclosing loop.

Example

```
'search to the end of the R&D Group
do GetNext
Loop While Field$("Dept") = "R&D"
```

Environ\$ function

Retrieves strings from the operating system's environment table.

Syntax A Environ\$(environment-string\$)

Syntax B Environ\$(n%)

Parameters

environment-string\$ The name of a keyword in the operating system environment. If this argument is given, it must be entered in uppercase, or it returns a null string. The value associated with the keyword is returned.

n% One of the strings from the operating system environment.

This can be any numeric expression, but it is rounded to a whole number by Environ\$. If this argument is used, Environ\$ returns the *n*th string from the environment table. This string is in the form "keyword = value."

Returns A string from the operating system's environment table.

Comments The argument of the Environ\$ function can be either a string (environment-string\$) or an integer (n%). A null string is returned if the specified argument cannot be found.

Example

```
'Get the Users Path
My Path = Environ$("Path")
```

Eof function

Indicates if the end of a file has been reached.

Syntax Eof (filenumber%)

Parameters

filenumber% The filenumber% is the number used in the Open statement of the file.

Returns A value indicating whether the end of a file has been reached.

Comments The Eof Function returns a (–1) if the end-of-file condition is true for the specified file.

Example The following fills a string-array from a file. (!= means not equal.)

```
While Eof(2) != -1
Input#2, A$(x)
x=x+1
Wend
```

Erl function

Gets the line number of the last trapped error. Can be used only as a function.

Syntax Erl

Returns The line number where an error was trapped.

Comments Using the Resume or On Error statements resets the Erl value to 0. If you want to maintain the value of the line number returned by Erl, you should assign it to a variable.

The value of the Erl function can be set indirectly through the Error statement.

Example MsgBox "Error on Line:" + str\$(Erl)

Err function

Determines the last run time error code. It can be used only as a function.

Syntax Err

Returns The runtime error code for the last error that was trapped.

Comments Using the Resume or On Error statements resets the Err value to 0. If you want to maintain the value of the error code returned by Err, you should assign it to a variable.

The value of the Err function can be set directly through the Err statement, and indirectly through the Error statement. See “Trappable errors” on page 225.

Example MsgBox "Error #" + Str\$(Err)

Err statement

Sends error information between procedures.

Syntax Err = n%

Parameters

n% An integer that contains the code of the error.

Comments The argument n% must be a 0 (indicating that no runtime error has been trapped) or an integer expression indicating a runtime error code (having a value between 1 and 32,767).

This statement can be used with Error\$ to generate application specific errors.

Example

```
'Set to user error number 276
Err=276
```

Error statement

Error errorcode% simulates the occurrence of a macro language or user-defined error.

Syntax Error errorcode%

Parameters

`errorcode%` Represents the error code, must be an integer between 1 and 32,767.

Comments If an `errorcode%` is one which the macro language already uses, the Error statement simulates an occurrence of that error.

User-defined error codes should employ values greater than those used for standard macro language error codes. To help ensure that non-macro language error codes are chosen, user-defined codes should work down from 32,767.

If an Error statement is executed and there is no error-handling routine enabled, the macro language produces an error message and halts program execution. If an Error statement specified an error code not used by the macro language, the message "User-defined error" is displayed.

Example Generate an error using a code of 7.

```
Error 7
```

Error\$ function

Returns an error message for the given error code. Can be used only as a function.

Syntax `Error$ [(errorcode%)]`

Parameters

`errorcode%` A number from 1–32,767

Returns The error message that corresponds to the specified error code.

Comments If the `errorcode%` is omitted, Basic returns the error message for the runtime error which has occurred most recently.

If no error message is found to match the `errorcode` null string("") is returned. See "Trappable errors" on page 225.

Example `MsgBox "Error #75 is" + Error$(75)`

Exit statement

Allows the program flow to escape from a loop function or subroutine.

Syntax `Exit {Do | For | Function | Sub}`

Parameters

Do For	Terminate loop statements.
Function Sub	Transfer control from the current procedure back to the original calling procedure.

Comments Exit Do can be used only within a Do...Loop statement. Exit For can be used only within a For...Next statement. In both cases, control is transferred to the statement which follows the loop statement. When used within a nested loop, an Exit statement moves control out of the immediately enclosing loop.

The Exit Function and Exit Sub statements transfer control from the current procedure back to the original calling procedure. Exit Function must be used in a function procedure. Exit Sub can be used only to exit from a Sub procedure.

Example

```
'Wait until 11am
Do
If time$="11:00" then Exit Do
Loop
```

Exp function

Returns the value e raised to the numeric-expression power.

Syntax Exp(numeric-expression)

Parameters

numeric-expression The numeric-expression is the value to raise Euler's number to.

Returns The value e raised to the numeric-expression power.

Comments The return value is single-precision for an integer or single-precision numeric expression. It is double precision for a long or double-precision numeric expression.

Example The following example finds the value of $e^{2.333}$.

```
value = Exp(2.333)
```

FileAttr function

Returns information about an open file.

Syntax FileAttr (filenumber%, attribute%)

Parameters

`filenumber%` The number used in the `Open` statement to open the file.

`attribute%` Either a 1 or 2. If `attribute%` is 2, `FileAttr` returns the operating system handle for the file.

Returns The `FileAttr` function returns information about an open file. Depending on the attribute chosen, this information is either the file mode or the operating system handle. The following table lists the return values and corresponding file modes if `attribute%` is 1. See the following table.

Value	Mode
1	Input
2	Output
8	Append

Example The following example gets the DOS file handle of the second open tile report.

```
handle=FileAttr(2,2)
```

Fix function

Returns the integer part of a numeric expression.

Syntax `Fix (numeric-expression)`

Parameters

`numeric-expression` The argument given is any numeric-expression.

Returns The integer part of a numeric-expression. The return type matches the type of numeric expression. This includes variant expressions which return a result of the same vartype as input. Except vartype 8 (string) is returned as vartype 5 (double) and vartype 0 (empty) is returned as vartype 3 (long).

Comments The argument given is any numeric expression. `Fix` removes the fractional part of the expression and returns the integer part only for both positive and negative numeric expressions. See the “`CInt` function” on page 299 and “`Int` function” on page 326.

Examples The following returns 6:

```
Fix (6.2)
```

The following returns -6:

```
Fix (-6.2)
```

For...Next statement

Repeats the statement block a fixed number of times, determined by the values of start, end, and step.

Syntax

```
For counter = start TO end [STEP increment]
    [ statementblock ]
    [ Exit For ]
    [ statementblock ]
Next [ counter ]
```

Parameters

start	The initial value.
end	The maximum value of the count before the loop is executed.
increment	The amount to add to the counter each time through the loop.
counter	A variable to hold the count.
statementblock	Basic functions, statements, or methods to be executed in the loop.

Comments In order for a For...Next loop to execute, the start and end values must be consistent with increment. If end is greater than start, increment must be positive. If end is less than start, increment must be negative. Basic compares the sign of (end-start) with the sign of Step. If the signs are the same, and end does not equal start, the For...Next loop is entered. If not, the loop is omitted in its entirety.

With a For...Next loop, the program lines following the For statement are executed until the Next statement is encountered. At this point, the Step amount is added to the counter and compared with the final value, end. If the beginning and ending values are the same, the loop executes once, regardless of the Step value. Otherwise, the Step value controls the loop as follows:

Step Value	Loop Execution
Positive	If counter is less than or equal to end, the Step value is added to counter. Control returns to the statement after the For statement and the process repeats. If counter is greater than end, the loop is exited; execution resumes with the statement following the Next statement.
Negative	The loop repeats until counter is less than end.
Zero	The loop repeats indefinitely.

Within the loop, the value of the counter should not be changed, as changing the counter makes programs more difficult to edit and debug.

For...Next loops can be nested within one another. Each nested loop should be given a unique variable name as its counter. The Next statement for the inside loop must appear before the Next statement for the outside loop. The Exit For statement can be used as an alternative exit from For...Next loops.

If the variable is left out of a Next statement, the Next statement matches the most recent For statement. If a Next statement occurs prior to its corresponding For statement, Basic returns an error message.

Multiple consecutive Next statements can be merged together. If this is done, the counters must appear with the innermost counter first and the outermost counter last.

Example

```
For i = 1 To 10
  [ statementblock ]
  For j = 1 To 5
    [ statementblock ]
  Next j, i
```

FreeFile function

Used when you need to supply a file number, and want to make sure that you are not choosing a file number which is already being used.

Syntax FreeFile

Returns The lowest unused file number.

Comments The value returned can be used in a subsequent Open statement.

Example

```
FileNumber=FreeFile
Open for output as filename "Temp.txt"
```

Function ... End Function statement

Defines a function procedure. It can be used only as a function. The purpose of a function is to produce and return a single value of a specified type.

Syntax

```
Function name [ ( parameter [ As type ] ... ) ]
  name = expression
End Function
```

Parameters The parameters are specified as a comma-separated list of parameter names. The data type of a parameter can be specified by using a type character or by using the As clause. Record parameters are declared by using an As clause and a type which has previously been defined using the Type statement.

Array parameters are indicated by using empty parentheses after the parameter. The array dimensions are not be specified in the Function statement. All references to an array parameter within the body of the function must have a consistent number of dimensions.

Returns You specify the return value by assigning it to the function name as if it were a variable or parameter. If no such assignment occurs, the value returned is 0 for numeric functions and the empty string ("") for string functions. The function returns to the caller when the End Function statement is reached or when an Exit Function statement is executed.

Comments Recursion is supported.

In the Function statement, the name of the function can end with a type character, which specifies the type that the function returns. When calling the function, you need not specify the type character.

Basic procedures use the call-by-reference convention. This means that if a procedure assigns a value to a parameter, it modifies the variable passed by the caller. This feature should be used with great care.

Use Sub to define a procedure which has no return value.

Example

```
Function = triangle(leg1 as double,leg2 as double)
Triangle = (leg1^2+leg2^2)^.5
End Function
```

GetField\$ function

Returns a substring from a delimited string. Can be used only as a function.

Syntax GetField\$(string\$, field_number%, separator_chars\$

Parameters

string\$	The source string is considered to be divided into fields by separator characters.
field_number%	The number of the substring to fetch.
separator_chars\$	The character that is used to separate the fields. E.g., in a comma delimited string it would be “,”.

Returns A substring from a source string.

Comments Multiple separator characters can be specified. The fields are numbered starting with one.

If field_number is greater than the number of fields in the string, the empty string is returned.

Example

```
GetField$("value1, value2, value3","2,","")
would return "value2"
```

Global statement

Declares global variables for use in a Basic program.

Syntax Global variableName [As type] [,variableName [As type]]

Parameters

variableName The name of your variable.

type A valid data type to assign your variable. (See “Variable data types” on page 222 for a list of valid data types.)

Comments Basic is a strongly typed language. The available data types are: numbers, strings, records, arrays, dialog boxes, and Application Data Types (ADTs).

Global data is shared across all loaded modules. If an attempt is made to load a module which has a global variable declared that has a different data type than an existing global variable of the same name, the module load fails.

If the As clause is not used, the type of the global variable can be specified by using a type character as a suffix to the variableName. The two different type-specification methods can be intermixed in a single Global statement (although not on the same variable).

Regardless of which mechanism you use to declare a global variable, you can choose to use or omit the type character when referring to the variable in the rest of your program. The type suffix is not considered part of the variable name.

Example Global A as string

GoTo statement

Changes a program flow by branching to a label.

Syntax GoTo Label

Parameters

Label The name of a label to navigate to.

Comments GoTo sends control to a label. The macro language does not support the use of line numbers.

A label has the same format as any other Basic name. To be recognized as a label, a name must begin in the first column and be followed immediately by a colon (":"). Reserved words are not valid labels.

GoTo cannot be used to transfer control out of the current function or sub.

GoTo is not recommended. While_Loops, Do_Loops, Select_Case statements for loops and subroutines can be used instead.

Example

```
Start:
If Value>Limit then GoTo Done
value=value+someore.
GoTo Start
Done:
```

GroupBox statement

Sets up a box that encloses sets of items, such as option boxes and check boxes that you wish to group together in a dialog box.

Syntax GroupBox x, y, dx, dy, text\$

Parameters

x, y	The x and y arguments set the position of the group box relative to the upper left corner of the dialog box.
dx, dy	dx and dy set the width and height of the box.
text\$	The text\$ field contains a title that is embedded in the top border of the group box.

Comments The GroupBox statement can be used only between a Begin Dialog and an End Dialog statement.

In the text\$ field, trailing characters are truncated if text\$ is wider than dx. If the text\$ argument is an empty string (""), the top border of the group box will be a solid line.

Example GroupBox 10,10,80,50, "My Group"

Hex\$ function

Converts from a decimal to a hex string.

Syntax Hex\$(numeric-expression)

Parameters

numeric-expression A decimal or integer to convert.

Returns A hexadecimal representation of a numeric-expression, as a string.

Comments If the numeric expression is an integer, the string contains up to four hexadecimal digits; otherwise, the expression is converted to a long integer, and the string can contain up to 8 hexadecimal digits.

Example `MsgBox "The hexadecimal representation of 175 is:" + Hex$(175)`

If ... Then ... Else

Organizes alternative actions into separate blocks of code.

Syntax A

```
If condition Then
    then_statement [Else else_statement ]
```

Syntax B

```
If condition Then
    statement_block
[ ElseIf expression Then
    statement_block]...
[ Else
    statement_block ]
End If
```

Note The syntax in both formats above, including the placement and organization of items on each line, must be followed exactly.

Comments The resulting action depends on the logical value of one or more conditions expressed in the structure.

The condition can be any expression which is evaluated as TRUE (non-zero) or FALSE (zero).

In the single-line version of the If statement, the then_statement and else_statement can be any valid single statement. Multiple statements separated by colons (:) are not allowed. When multiple statements are required in either the Then or Else clauses, use the block version of the If statement.

In the block version of the If statement, the statement_blocks can be made up of zero or more statements, separated by colons (:) or on different lines.

Example

```
If total>limit then
MsgBox "over limit"
Else
MsgBox "under limit"
```

'\$Include metacommand

Tells the compiler to include statements from another file.

Syntax '\$Include: "filename"

Parameters

filename A file with Basic source code to compile along with the current source.

Comments Comments which include metacommands are only recognized at the beginning of a line. For compatibility with other versions of Basic, you can use single quotes (") to enclose the filename.

A file extension of .SBH is suggested for the macro language include files. This is only a recommendation, and any other valid file extension can be used. If no directory or drive is specified, the compiler searches for filename on the source file search path.

Example I\$ Include "MYLIB.BAS"

Input\$ function

Reads a specified number of characters from a file. Can be used only as a function.

Syntax Input\$ (numchars%, [#]filenumber%)

Parameters

numchars% The numchars% argument contains the number of characters (bytes) to read from the file.

filenumber% The filenumber% argument is an integer expression identifying the open file to read from.

Returns A string containing the characters read.

Comments The file pointer is advanced the number of characters read. Unlike the Input # statement, Input\$ returns all characters it reads, including carriage returns, line feeds, and leading spaces.

This function is useful when you want to read in characters like carriage returns, commas, or other characters that are used as delimiters for the Input\$ statement.

Example The following example reads 20 characters from File2.

```
Input$ (20,#2)
```

Input # statement

Reads data from a sequential file and assigns the data to variables.

Syntax

Input [#] filename%, variable, [variable]

Input [prompt\$,] variable, [variable]

Parameters

`filename%` An integer expression identifying the open file to read from. This is the number used in the Open statement to open the file.

`prompt$` An optional string that can be used to prompt for keyboard input.

`variable` Lists the variables that are assigned the values read from the file. The list of variables is separated by commas.

Comments If `filename` is not specified, the user is prompted for keyboard input with a question mark "?", unless `prompt$` is specified.

Examples Open "c:\MyFile.Txt" for input as 1

Or,

```
input #2, Name$, age, address$
```

InputBox\$ function

Displays a dialog box containing a prompt.

Syntax InputBox\$(prompt\$[,title\$ [,default\$ [,xpos%,ypos%]]])

Parameters

<code>prompt\$</code>	A string expression containing the text to be shown in the dialog box. The length of <code>prompt\$</code> is restricted to 255 characters. This figure is approximate and depends on the width of the characters used. Note that a carriage return and a line-feed character must be included in <code>prompt\$</code> if a multiple-line prompt is used.
<code>title\$,default\$</code>	The caption that appears in the dialog box's title bar. <code>Default\$</code> is the string expression that is shown in the edit box as the default response. If either of these arguments is omitted, nothing is displayed.
<code>xpos%, ypos%</code>	Numeric expressions, specified in dialog box units, that determine the position of the dialog box. <code>Xpos%</code> determines the horizontal distance between the left edge of the screen and the left border of the dialog box. <code>Ypos%</code> determines the horizontal distance from the top of the screen to the dialog box's upper edge. If these arguments are not entered, the dialog box is centered roughly one third of the way down the screen. A horizontal dialog box unit is 1/4 of the average character width in the system font; a vertical dialog box unit is 1/8 of the height of a character in the system font. If you wish to specify the dialog box's position, you must enter both of these arguments. If you enter one without the other, the default positioning is set.

Returns After the user presses Enter, or selects the OK button, `InputBox$` returns the text contained in the input box. If the user selects Cancel, the `InputBox$` function returns a null string.

Comments Once the user has entered text, or made the button choice being prompted for, the contents of the box are returned.

Example The following example should appear in your code on one line.

```
Street$=InputBox$("Enter Street Name","MyTitle","Default Exp",15,15)
```

InStr function

Finds the occurrence of a substring in a string. The `InStr` function can be used only as a function.

Syntax `InStr ( [position%], string$, substring$ )`

Parameters

<code>position%</code>	Indicates the index of the character within <code>string\$</code> where the search should start. If not specified, the search starts at the beginning of the string (equivalent to a <code>position%</code> of 1).
<code>string\$</code>	The string being searched.
<code>substring\$</code>	The string being looked for.

Returns An integer representing the position of the first occurrence of a substring within another string.

Comments These arguments can be string variables, string expressions, or string literals.

If the `position%` argument is greater than the length of the substring, if the `string$` argument is a null string, or if the `substring$` cannot be located, `InStr` returns a zero. If the `substring$` argument is a null string, then the `position%` argument is returned.

The index of the first character in a string is 1.

Example The following starts at the first character, searches for JR. in the field `Last_Name` from the Employee table.

```
Instr(1, "Employee.Last_Name", "JR.")
```

Int function

Returns the integer portion of a number. Can be used only as a function.

Syntax `Int ( numeric-expression )`

Parameters

`numeric-expression` The argument given is any numeric-expression.

Returns The integer part of a numeric-expression.

Comments For positive numeric-expressions, `Int` removes the fractional part of the expression and returns the integer part only. For negative numeric-expressions, `Int` returns the largest integer less than or equal to the expression.

Example For example, `Int (6.2)` returns 6; `Int(-6.2)` returns -7. See the “CInt function” on page 299 and “Fix function” on page 316.

Kill statement

Deletes files from disk.

Syntax `Kill filespec$`

Parameters

`filespec$` A string expression that specifies a valid DOS file specification. This specification can contain paths and wildcards.

Comments Kill deletes files only, not directories. Use the `Rmdir` function to delete directories.

Example Kill "c:\Temp.Txt"

LBound function

Declares the number that represents the first element of an array. Can be used only as a function.

Syntax LBound (arrayVariable [, dimension])

Parameters

`arrayVariable` The name of the array to declare a lower bound for.

`dimension` The number that represents the first element of an array.

Returns The lower bound of the subscript range for the specified dimension of the `arrayVariable`.

Comments The dimensions of an array are numbered starting with 1. If the dimension is not specified, 1 is used as a default.

LBound can be used with UBound to determine the length of an array.

Example

```
'Start employee array at element 5
LBound (Employee, 5)
```

LCase\$ function

Converts the characters in a string to lower case. Can be used only as a function.

Syntax LCase\$ (string\$)

Parameters

`string$` The string to convert.

Returns A copy of the source string, with all upper-case letters converted to lower case.

Comments The translation is based on the country specified in the Windows Control Panel.

Example The following example changes “First Street” to “first street.”

```
Lower_Case = LCase$("First Street")
```

Left\$ function

Returns the left n characters of a string. It can be used only as a function.

Syntax Left\$ (string\$, length%)

Parameters

string\$ The string to characters from.

length% The number of characters to return.

Returns A string of a specified length copied from the beginning of the source string. The dollar sign (\$) in the function name is optional. If specified the return type is string. If omitted the function typically returns a variant of vartype 8 (string). If the value of expression is null a variant of vartype 1 (null) is returned.

Comments If the length of string\$ is less than length%, Left\$ returns the whole string. Left\$ accepts expressions of type string and any type of expression including numeric values and converts the input value to a string.

Example The following will return “C:\”

```
left$ ("c:\RPTSMITH",3)
```

Len function

Returns the length of a string or the number of bytes used to store a non-string value. Can be used only as a function.

Syntax

```
Len ( string$ )      Len ( non-string )
```

Parameters

string\$,non-string If the argument is a string, the number of characters in the string is returned; otherwise, the length of the built-in datatype or user-defined type is returned.

Returns The length of the argument.

Comments The argument can be of any type. If the argument is a string, the number of characters in the string is returned. If the argument is a variant variable the number of bytes required to represent its value as a string is returned. If not, the length of the built-in datatype or user-defined type is returned.

Example The following example returns a value of 8.

```
Length_Var = Len ("Thursday")
```

Let (assignment statement)

Stores a value in a Basic variable. The keyword Let is optional.

Syntax [Let] variable = expression

Parameters

Let Used to assign to a numeric, string or record variable. You can also use the Let statement to assign to a record field or to an element of an array.

Comments You can also use this statement to assign to a record field or to an element of an array. When assigning a value to a numeric or string variable, standard conversion rules apply.

Example

```
Let A =1
Let Name$ = "John"
```

Line Input # statement

Reads a line from a sequential file into a string variable or brings up a dialog box with an edit control that allows you to enter a string.

Syntax

Line Input [#] filename%, variable\$
 Line Input [prompt\$,] variables

Parameters

Line Input [#]	The Line Input # statement reads a line from a sequential file into a string variable.
filenumber%	An integer expression identifying the open file from which to read. This is the number used in the Open statement to open the file.
prompt\$	An optional string that can be used to prompt for keyboard input.
variable\$	A string variables into which the line from the input file is read.

Comments If filenumber is not specified, the user is prompted for keyboard input with a question mark (?) unless prompt\$ is specified.

Example Line Input #1, A\$

ListBox statement

Used to create a list of choices.

Syntax A ListBox x, y, dx, dy, text\$, .field

Syntax B ListBox x, y, dx, dy, stringarray\$(), .field

Parameters

x, y	Give the coordinates that the upper left corner of the list box, relative to the upper left corner of the dialog box. The x argument is measured in 1/4 system-font character-width units. The y argument is measured in 1/8 system-font character-width units. (See “Begin Dialog ... End Dialog statement” on page 293.)
dx, dy	Specify the width and height of the list box.
text\$	A string containing the selections for the list box. This string must be defined, using a Dim statement, before the Begin Dialog statement is executed.
.field	The name of the dialog-record field that holds the selection made from the list box. When the user selects OK (or selects the customized button created using the Button statement), a number representing the selection’s position in the text\$ string is recorded in the field designated by the .field argument. The numbers begin at 0. If no item is selected, it is -1.

Comments The ListBox statement can be used only between a Begin Dialog and an End Dialog statement.

When syntax A is used, the text\$ argument is a string containing the selections for the list box. This string must be defined, using a Dim statement, before the Begin Dialog statement is executed. Where dimname is the name of a String variable defined in a Dim statement, listchoice is the text that appears as a selection in the list box, and Chr\$(9) is the function call that produces a tab character. Note that multiple selections can be specified in text\$ by separating the list choices with tab characters, as shown above.

Example The arguments in the text\$ string are entered as shown in the following example.

```
dimname =
"listchoice"+Chr$(9)+"listchoice"+Chr$(9)
+"listchoice"+Chr$(9)... ,.Selection_Holder
```

Lof function

Determines the length of a file in bytes. It can be used only as a function.

Syntax Lof (filename%)

Parameters

filename% An integer expression identifying the open file from which the file length is read.

Returns The length in bytes of the file specified by filename%.

Comments The filename% is the number used in the Open statement of the file.

Example MsgBox = "c:\Sample.txt is"; Lof (7); "Bytes Long"

Log function

Determines the natural logarithm of an expression. Can be used only as a function.

Syntax Log(numeric-expression)

Parameters

numeric-expression Determines the natural log of an expression.

Returns The value natural logarithm of numeric-expression.

Comments The return value is single-precision for an integer or single-precision numeric expression, or double precision for a long or double-precision numeric expression.

Example MsgBox "The natural log of 12,000 is:" + Str\$ (log(12,000))

LTrim\$ function

Returns a copy of the source string with all leading characters removed.

Syntax LTrim\$ (string\$)

Parameters

`string$` The string to strip leading spaces from.

Returns A copy of the source string, with all leading space characters removed. The dollar sign (\$) in the function name is optional. If specified the return type is string. If omitted the function typically returns a variant of vartype 8 (string). If the value of expression is null a variant of vartype 1 (null) is returned.

Comments `LTrim$` accepts expressions of type string and any type of expression including numeric values and converts the input value to a string.

Example The following example results in "October 7."

```
LTrim$(" October 7")
```

Mid statement

Replaces part of a string with another string.

Syntax `Mid (string$, position%[, length%] ) = subst-string$`

Parameters

`string$` The string expression into which the substring is placed.
`position%` The character in a string where substring is inserted.
`length%` The number of characters for the substring to insert at the string.
`subst-string$` A string to insert in another string.

Returns The specified substring in `string$` with `substring$`.

Comments If the `length%` argument is left out, or if there are fewer characters in a string than specified in `length%`, then `Mid$` replaces all the characters from the `position%` to the end of the string. If `position%` is larger than the number of characters in the indicated `string$`, then `Mid$` appends `subst-string%` to `string$`.

The index of the first character in a string is 1.

Example

```
A$ = "One rotten day"
Mid$ (A$, 5, 6) = "Happy"
Will leave A$ equal to "One happy day"
```

Mid\$ function

Returns a substring of a specified `length%` from a source expression, starting with the character at the specified `position%`. Can be used only as a function.

Syntax Mid\$ (string\$, position%[, length%])

Parameters

string\$ The string expression from which a sub-string is taken.
 position% The start of the sub-string.
 length% The length of the sub-string.

Returns A substring of a specified length% from a source string\$, starting with the character at the specified position%.

The dollar sign (\$) in the function name is optional. If specified the return type is string. If omitted the function typically returns a variant of vartype 8 (string). If the value of expression is null a variant of vartype 1 (null) is returned.

Comments If the length% argument is left out, or if there are fewer characters in a string than specified in length%, then Mid\$ returns all the characters from the position% to the end of the string. If position% is larger than the number of characters in the indicated string\$, then Mid\$ returns a null string.

The index of the first character in a string is 1.

To modify a portion of a string value, see “Mid statement” on page 332.

Example The following example returns “press.”

```
Return_value = Mid$ ("expression", 3,4)
```

MkDir

Makes a new directory.

Syntax MkDir pathname\$

Parameters

pathname\$ A string expression identifying the new default directory.

Comments The syntax for pathname\$ is:

```
[drive:] [\] directory [\directory]
```

The drive argument is optional. If omitted, MkDir makes a new directory on the current drive. The directory argument is a directory name.

Example The following example creates a directory called “2ndQrt” under the “RptSmith” directory.

```
MkDir "c:\RptSmith\2ndQrt"
```

Msgbox function

Displays a message in a dialog box. It can be used only as a function.

Syntax `Msgbox( message$,type%[, caption$] )`

Parameters

`message$` A string to display to the user.

`type%` Governs the icons and buttons that are displayed in the dialog box. This argument is the sum of values describing the number and type of radio buttons that appear, the icon style, and the default button. One selection can be made from each group.

Note: If `type%` is omitted, a single OK button appears.

Group 1: Buttons

- 0 OK only
- 1 OK, Cancel
- 2 Abort, Retry, Ignore
- 3 Yes, No, Cancel
- 4 Yes, No
- 5 Retry, Cancel

Group 2: Icons

- 16 Critical Message (STOP)
- 32 Warning Query (?)A
- 64 Information Message (i)

Group 3: Defaults

- 0 First button
- 256 Second button
- 512 Third button

`caption$` Appears on the message dialog box. `Caption%` is a string expression that appears in the dialog box's title bar.

Returns An integer value indicating which button the user selected.

The return values for the Msgbox function are:

- OK
- Cancel
- Abort
- Retry
- Ignore
- Yes

Comments The message displayed must be no more than 1024 characters long. A message string greater than 255 characters without intervening spaces is truncated after the 255th character. Once the user has selected a pushbutton, MsgBox returns a value indicating the user's choice.

Example `Button_Pressed=Msgbox("Please pick Yes or No",4,"Title")`

Msgbox statement

Displays a message in a dialog box.

Syntax `MsgBox message$[ ,type%[ , caption$] ]`

Parameters

`type%` Governs the icons and buttons that are displayed in the dialog box. This argument is the sum of values describing the number and type of radio buttons that will appear, the icon style, and the default button. One selection should be made from each group. If `type%` is omitted, a single OK button appears.

Group 1: Buttons

- 0 OK only
- 1 OK, Cancel
- 2 Abort, Retry, Ignore
- 3 Yes, No, Cancel
- 4 Yes, No
- 5 Retry, Cancel

Group 2: Icons

- 16 Critical Message (STOP)
- 32 Warning Query (?)
- 48 Warning Message (!)
- 64 Information Message (i)

Group 3: Defaults

- 0 First button
- 256 Second button
- 512 Third button

`caption$` A string expression that appears in the dialog box's title bar.

Comments A message can be no longer than 1024 characters in length. Messages longer than 255 characters which contain no intervening spaces are truncated after the 255th character.

Examples `Msgbox "Do you want to Retry or Cancel?",5`

Or,

`Msgbox "Have a nice day!"`

Name statement

Renames a file. It can also be used to move a file from one directory to another.

Syntax `Name oldfilename$ As newfilename$`

Parameters

`oldfilename$,newfilename$` String expressions that designate, respectively, the file to rename and the new name for that file.

Comments A path can be part of the `filename$`. If the paths are different, the file is moved to the new directory.

If the file `oldfilename$` is open, Basic generates an error message. A file must be closed in order to be renamed. If the file `newfilename$` already exists, Basic generates an error message.

Example `rename "win.ini" as "win.bak"`

'\$NoCStrings metacommand

Tells the compiler to treat a backslash inside a string as a normal character. This is the default.

Syntax `'$NoCStrings`

Comments You can use the `'$CStrings` metacommand to tell the compiler to treat a backslash character inside of a string as an escape character.

Example `'$NoCStrings`

Oct\$ function

Returns an octal representation of a number. Can be used only as a function.

Syntax `Oct$( numeric-expression )`

Parameters

`numeric-expression` A decimal number to convert to octal.

Returns An octal representation of a `numeric-expression`, as a string. If the numeric expression is an integer, the string contains up to six octal digits; otherwise, the expression is converted to a long integer, and the string can contain up to 11 octal digits.

The dollar sign (\$) in the function name is optional. If specified the return type is string. If omitted the function typically returns a variant of vartype 8 (string).

Comments The octal value is returned as a string of characters.

Example `MsgBox "The octal representation of 175 is " + oct$(175)`

OkButton statement

Determines the position and size of an OK button.

Syntax `OkButton x, y, dx, dy [,id]`

Parameters

<code>x, y</code>	Set the position of the OK button relative to the upper left corner of the dialog box.
<code>dx, dy</code>	Set the width and height of the button.
<code>id</code>	An optional identifier used by the dialog statements that act on this. A <code>dy</code> value of 14 typically accommodates text in the system font.

Comments The `OkButton` statement can be used only between a `Begin Dialog` and an `End Dialog` statement.

Example `OkButton 10,10,90,20`

On Error statement

Enables an error-handling routine, specifying the location of that routine within procedure.

Syntax `ON [Local] Error {GoTo label [ Resume Next ] GoTo 0}`

Parameters

<code>Local</code>	Keyword allowed in error-handling routines at the procedure level. Used to ensure compatibility with other variants of Basic
<code>GoTo label</code>	Enables the error-handling routine that starts at label. If the designated label is not in the same procedure as the <code>On Error</code> statement, Basic generates an error message.
<code>Resume Next</code>	Establishes that when a runtime error occurs, control is passed to the statement which immediately follows the statement in which the error occurred. At this point, the <code>Err</code> function can be used to retrieve the error-code of the runtime error.
<code>GoTo 0</code>	Disables any error handler that has been enabled.

Comments `On Error` can also be used to disable an error-handling routine. Unless an `On Error` statement is used, any runtime error will be fatal. The Softbridge language (SBL) will terminate the execution of the program.

When it is referenced by an On Error GoTo label statement, an error-handler is enabled. After this enabling occurs, a runtime error results in program control switching to the error-handling routine and “activating” the error handler. The error handler remains active from the time the runtime error has been trapped until a Resume statement is executed in the error handler.

If another error occurs while the error handler is active, the SBL searches for an error handler in the procedure which called the current procedure (if this fails, the macro language looks for a handler belonging to the caller’s caller, and so on). If a handler is found, the current procedure terminates, and the error handler in the calling procedure is activated.

It is an error (No Resume) to execute an End Sub or End Function statement while an error handler is active. The Exit Sub or Exit Function statement can be used to end the error condition and exit the current procedure.

Example On Error Goto ErrorProc

Open statement

Enables I/O to a file or a device.

Syntax Open filename\$ For mode As [#] filenumber% [Len=reclen]

Parameters

filename\$	The argument filename\$ is a string expression specifying the file to open.
mode	Mode is a keyword that specifies one of the following: Input (sequential input mode), Output (sequential output mode), or Append (sequential output mode).
filenumber%	Filenumber% is an integer expression with a value between 1 and 255.
reclen	Specifies the record length for files opened in random mode. It is ignored for all other modes.

Comments A file must be opened before any input/output operation can be performed on it.

The FreeFile function can be used to return the next available filenumber.

Example Open "Data.txt" for input as 1

Option Base statement

Specifies the default lower bound to be used for array subscripts.

Syntax Option Base lowerBound%

Comments The lowerBound must be either 0 or 1. If no Option Base statement is specified, the default lower bound for array subscripts is 0.

The Option Base statement is not allowed inside a procedure, and must precede any use of arrays in the module. Only one Option Base statement is allowed per module.

Example

```
'All arrays start at element
Option Base 5
```

OptionButton statement

Used to define the position and text associated with an option button. There must be at least two Option Button statements. They are used in conjunction with the OptionGroup statement.

Syntax OptionButton x, y, dx, dy, text\$

Parameters

x, y	Set the position of the button relative to the upper left corner of the dialog box.
dx, dy	Set the width and height of the button. A dy value of 12 typically accommodates text in the system font.
text\$	Contains the caption that appears to the right of the option button icon. If the width of this string is greater than dx, trailing characters are truncated.

Comments The OptionButton statements can be used only between a Begin Dialog and an End Dialog statement.

If you want to include underlined characters so that the option selection can be made from the keyboard, the character must be preceded with an ampersand (&).

Example OptionButton 10,10,110,20,"Sales Reports"

OptionGroup statement

Used in conjunction with OptionButton statements to set up a series of related options.

Syntax OptionGroup .field

Parameters

.field	The name of the dialog box field that contains the index of the selected options button.
--------	--

Comments The OptionGroup statement begins definition of the option buttons and establishes the dialog-record field that contains the current option selection. .Field contains a value 0 when the choice associated with the first OptionButton statement is selected, a value of 1 when the choice associated with the second OptionButton statement is chosen, and so on.

The `OptionGroup` statement can be used only between a `Begin Dialog` and an `End Dialog` statement.

Example `OptionGroup .selected_option`

Print statement

Outputs data to the specified `filenumber%`.

Syntax `Print [ # filenumber%, ] expressionlist [ { ; | , } ]`

Parameters

<code>filenumber%</code>	An integer expression identifying the print destination.
<code>expressionlist</code>	The values that are printed. The argument can contain numeric or string expressions.

Comments If the `expressionlist` is omitted, a blank line is written to the file.

`Filenumber%` is optional. If this argument is omitted, the `Print` statement outputs data to the screen.

Expressions are separated by either a semicolon (;) or a comma (,). A semicolon indicates that the next value should appear immediately after the preceding one without intervening white space. A comma indicates that the next value should be positioned at the next print zone. Print zones begin every 14 spaces.

The optional `[[;|,]]` argument at the end of the `Print` statement determines where output for the next `Print` statement to the same output file should begin. A semicolon places output immediately after the output from this `Print` statement on the current line; a comma starts output at the next print zone on the current line. If neither separator is specified, a CR-LF pair is generated and the next `Print` statement prints to the next line.

The `Print` statement supports only elementary Basic data types.

Example `Print #1, MyData$, Index, Amount`

Randomize statement

Seeds the random number generator.

Syntax `Randomize [ numeric-expression% ]`

Parameters

<code>numeric-expression%</code>	An integer value between -32768 and 32767.
----------------------------------	--

Comments If no `numeric-expression%` argument is given, Basic uses the `Timer` function to initialize the random number generator.

Example Randomize Timer

ReDim statement

Changes the upper and lower bounds of a dynamic array's dimensions.

Syntax ReDim [Preserve] variableName (subscriptRange, ...) [As[New] type] ,...

Parameters

variableName An array to re-dimension.

subscriptRange New dimensions.

Comments Memory for the dynamic array is reallocated to support the specified dimensions, and the array elements are reinitialized. ReDim can not be used at the module level—it must be used inside of a procedure.

The Preserve option is used to change the last dimension in the array while maintaining its contents. If Preserve is not specified the contents of the array will be reinitialized. Numbers are set to zero. String variants are set to empty.

A dynamic array is normally created by using Dim to declare an array without a specified subscriptRange. The maximum number of dimensions for a dynamic array created in this fashion is 8. If you need more than 8 dimensions, you can use the ReDim statement inside of a procedure to declare an array which has not previously been declared using Dim or Global. In this case, the maximum number of dimensions allowed is 60.

The available data types for arrays are: numbers, strings, and records. Arrays of arrays, dialog box records, and ADTs are not supported.

If the As clause is not used, the type of the variable can be specified by using a type character as a suffix to the name. The two different type-specification methods can be intermixed in a single ReDim statement (although not on the same variable).

The ReDim statement cannot be used to change the number of dimensions of a dynamic array once the array has been given dimensions. It can change only the upper and lower bounds of the dimensions of the array. The LBound and UBound functions can be used to query the current bounds of an array variable's dimensions.

Do not use the ReDim statement on an array in a procedure that has received a reference to an element in the array in an argument. The result is unpredictable.

The subscriptRange is of the format:

```
[ startSubscript To ] endSubscript
```

If startSubscript is not specified, 0 is used as the default. The Option Base statement can be used to change the default.

Example

```
'dimension to a 2x13 array  
Redim DynaList (3 to 5, 7 to 20)
```

Rem statement

Inserts a comment in a Basic program. Everything from Rem to the end of the line is ignored.

Syntax Rem arbitrary text

Comments The single quote (') can also be used to initiate a comment. Metacommands (e.g., CSTRINGS) must be preceded by the single quote comment form.

Example

```
Rem this comment line is not compiled.  
'This statement is also not compiled. It is a comment.
```

Reset statement

Closes all disk files that are open, and writes any data still remaining in the operating system buffers to disk.

Syntax Reset

Example Reset

Resume statement

Halts an error-handling routine.

Syntax

```
Resume Next  
Resume label  
Resume [ 0 ]
```

Parameters

- Resume Next** When used, control is passed to the statement which immediately follows the statement in which the error occurred.
- Resume label** When used, control is passed to the statement which immediately follows the specified label.
- Resume [0]** When used, control is passed to the statement in which the error occurred.

Comments The location of the error handler which has caught the error determines where execution resumes. If an error is trapped in the same procedure as the error handler, program execution resumes with the statement that caused the error. If an error is located in a different procedure from the error handler, program control reverts to the statement that last called out the procedure containing the error handler.

Example The following example goes to the Reset Procedure after an error.

```
Resume Reset_Proc
```

Right\$ function

Returns a string of a specified length.

Syntax Right\$ (string\$, length%)

Parameters

- string\$** The string from which to copy characters.
- length%** The number of characters to copy.

Returns A string of a specified length copied from the right-most length characters from the string\$.

Comments If the length of string\$ is less than length%, Right\$ returns the whole string.

Example The following example returns "RptSmith."

```
Return_Var$ = Right$("c:\RptSmith",8)
```

RmDir statement

Removes a directory.

Syntax RmDir pathname\$

Parameters

pathname\$ A string expression identifying the directory to remove.

Comments The syntax for pathname\$ is:

```
[drive:] [\] directory [\directory]
```

The drive argument is optional. The directory argument is a directory name. The directory to be removed must be empty, except for the working (.) and parent (...) directories.

Example

```
'Remove a temporary directory under ReportSmith
Rmdir "c:\RptSmith\Temp"
```

Rnd function

Generates a random number between 0 and 1.

Syntax Rnd [(number!)]

Parameters The following table summarizes the valid values for number.

number	<0	Same random number for a given number
	>0	Next random number
	= 0	Last random number
	Omitted	Next random number

Returns A single precision random number between 0 and 1.

Comments The same sequence of random numbers is generated whenever the program is run, unless the random number generated is re-initialized by the Randomize statement.

Example The following example generates a random number between 1 and 100

```
x = (RND*100) +1
```

RTrim\$ function

Removes trailing spaces.

Syntax RTrim\$ (string\$)

Parameters

`string$` A string from which to remove spaces.

Returns A copy of the source string, with all trailing space characters removed.

Example `RTrim$ ("John ") + RTrim$ (" Doe") = "John Doe"`

Seek function

Determines the current position in a file.

Syntax `Seek (filenumber%)`

Parameters

`filenumber%` `Filenumber%` is an integer expression identifying the open file from which to read the file position.

Returns The current file position for the file specified by `filenumber%`. For files opened in Random mode, Seek returns the number of the next record to be read or written. For all other modes, Seek returns the file offset for the next operation. The first byte in the file is at offset 1, the second byte is at offset 2, and so on. The return value is a Long.

Comments See “Open statement” on page 338 for more details.

Example `Position1 = Seek(1)`

Seek statement

Sets the position within a file for the next read or write.

Syntax `Seek [#] filenumber%, position&`

Parameters

`filenumber%` `Filenumber%` is an integer expression identifying the open file to Seek in.

`position&` `Position&` is a numeric expression that indicates where the next write or read occurs. Value must be between 1 and 2,146,483,647.

Comments If you write to a file after seeking beyond the end of the file, the file length is extended. Basic returns an error message if a Seek operation is attempted which specifies a negative or zero position.

Example The following example positions the cursor to Record 50.

```
Seek #1, 50
```

Select Case statement

Executes one of a series of statement blocks, depending on the value of an expression.

Syntax

```
Select Case testexpression
[Case expressionlist
    [statement_block] ]
[Case expressionlist
    [statement_block] ]
.
.
.
[Case Else
    [statement_block] ]
End Select
```

Parameters

testexpression Any numeric or string expression for which you test. Each **statement_block** can contain any number of statements on any number of lines.

Comments The **expressionlist(s)** can be a comma-separated list of expressions of the following forms:

- **expression**
- **expression To expression**
- **Is comparison_operator expression**

The type of each expression must be compatible with the type of **testexpression**.

When there is a match between **testexpression** and one of the **Case** expressions, the statement block following the **Case** clause is executed. When the next **Case** clause is reached, execution control passes to the statement which follows the **End Select** statement.

Note that when the **To** keyword is used to specify a range of values, the smaller value must appear first. The **comparison_operator** used with the **Is** keyword is one of: **<**, **>**, **=**, **<=**, **>=**, **<>**.

Example

```
Select Case price
Case is > 100
    MsgBox "Too expensive"
Case 50 to 99
    MsgBox "Good price"
Case is < 50
    MsgBox "Sale!"
End Select
```

SetField\$ function

Sets a value in a list of values whose fields are all separated by the same character.

Syntax SetField\$(string\$, field_number%, field\$, separator_chars\$)

Parameters

string\$	A string that is the list of items to update.
field_number%	The number if the item in the list to update.
field\$	The new item to be placed in the list.
separator_chars\$	The character that is used to separate individual items in the list.

Returns A string created from a copy of the source string with a substring replaced.

Comments The source string is considered to be divided into fields by separator characters. Multiple separator characters can be specified. The fields are numbered starting with one.

If field_number is greater than the number of fields in the string, the returned string is extended with separator characters to produce a string with the proper number of fields. If more than one separator character was specified, the first one is used as the separator character.

It is legal for the new field value to be a different size than the old field value.

Example The following returns "one\2\three."

```
SetField ("one\two\three",2,"2","\")
```

Sgn function

Determines the sign of a number. Can be used only as a function.

Syntax Sgn (numeric-expression)

Parameters

numeric-expression The number for which to get the sign.

Returns A value indicating the sign of the numeric expression. The value that the Sgn function returns depends on the sign of the expression:

- For numeric-expressions > 0, Sgn (numeric-expression) returns 1.
- For numeric-expressions = 0, Sgn (numeric-expression) returns 0.
- For numeric-expressions < 0, Sgn (numeric-expression) returns -1.

Example The following example makes a number positive.

```
number_sign = number * sgn(number)
```

Shell function

Runs an executable program.

Syntax Shell (commandstring\$, [windowstyle%])

Parameters

`commandstring$` The name of the program to execute. It can be the name of any valid .COM, .EXE, .BAT, or .PIF file. Arguments or command line switches can also be included.

`windowstyle%` The style of the window in which the program is to be executed. It can be one of the following:

- 1Normal window with focus
- 2Minimized with focus
- 3Maximized with focus
- 4Normal window without focus
- 5Minimized without focus

If `windowstyle%` is not specified, the default of `windowstyle% = 1` is assumed (normal window with focus).

Returns A unique number that identifies the running program.

Comments If `commandstring$` is not a valid executable file name or if Shell cannot start the program, an error message is generated.

Example The following command launches Excel.

```
Task_id = Shell("c:\Excel\Excel.Exe",1)
```

Sin function

Calculates the sine of an angle specified in radians. Can be used only as a function.

Syntax Sin(angle)

Parameters

`angle` The angle in radians to compute the sin for.

Returns The sine of an angle. The return value is between -1 and 1. The return value is single-precision if the angle is an integer or single-precision value, double precision for a long or double-precision value.

Comments The angle is specified in radians, and can be either positive or negative.

Example The following calculates the sin of 60 degrees.

```
Value = sin ((60x3.1415)/180)
```

Space\$ function

Generates a string with the given number of spaces. Can be used only as a function.

Syntax Space\$(numeric-expression)

Parameters

numeric-expression Indicates the number of spaces which the returned string contains.

Returns A string of spaces.

Comments Any numeric data type can be used, but the number is rounded to an integer. The numeric expression must be between 0 and 32,767.

Example The following example evaluates to "One_ _ _ _ _Two_ _ _ _ _Three."

```
"One" + Space$(5) + "Two" + Space$(5) + "Three"
```

Sqr function

Calculates the square root of a number. Can be used only as a function.

Syntax Sqr(numeric-expression)

Parameters

numeric-expression Number for which to get the square root.

Returns The square root of numeric-expression.

Comments The return value is single-precision for an integer or single-precision numeric expression, or double precision for a long or double-precision numeric expression. A negative value causes a runtime illegal function call error.

Example $Z = \text{Sqr} (x^2 + y^2)$

Stop statement

Halts program execution.

Syntax Stop

Comments Stop statements can be placed anywhere in a program to suspend its execution. While the Stop statement halts program execution, it does not close files or clear variables.

Example If Resume Code = 0 Then Stop

Str\$ function

Converts a number to a string. Can be used only as a function.

Syntax Str\$(numeric-expression)

Parameters

numeric-expression A number to be converted to a string.

Returns The Str\$ function returns a string representation of a numeric-expression.

Comments The returned string is single-precision for an integer or single-precision numeric expression, double precision for a long or double-precision numeric expression.

Example MsgBox "The value is:" + Str\$ (value)

String\$ function

Creates a string of the given character repeated the given number of times. Can be used only as a function.

Syntax

String\$(numeric-expression, charcode%)

String\$ (numeric-expression, string-expression\$)

Parameters

numeric-expression Specifies the length of the string to be returned. This number must be between 0 and 32,767.

charcode% A decimal ANSI code of the character that is used to create the string. It is a numeric expression that Basic will evaluate as an integer between 0 and 255.

string-expression\$ A string argument, the first character of which becomes the repeated character.

Returns The String\$ function returns a string consisting of a repeated character.

Example DerivedField String\$ (10,"=")

Sub ... End Sub statement

Defines a subprogram procedure.

Syntax Sub name [(parameter [As type] , ...)] End Sub

Parameters The parameters are specified as a comma-separated list of parameter names. A parameter's data type can be specified either by using a type character or by using the As clause.

Record parameters are declared by using an As clause and a type which has previously been defined using the Type statement.

Array Parameters are indicated by using empty parentheses after the parameter. The array dimensions are not specified in the Sub statement. All references to an array parameter within the body of the subprogram must have a consistent number of dimensions.

Returns Returns to the caller when the End Sub statement is reached or when an Exit Sub statement is executed.

Comments A call to a subprogram stands alone as a separate statement. (See "Call statement" on page 295.)

Recursion is supported.

Basic procedures use the call-by-reference convention. This means that if a procedure assigns a value to a parameter, it modifies the variable passed by the caller. This feature should be used with great care.

The MAIN subprogram has a special meaning. In many implementations of Basic, MAIN is called when the module is "run". The MAIN subprogram is not allowed to take arguments.

Example

```
Sub Hello()  
MsgBox "Hello"  
End Sub
```

Tan function

Returns the tangent of an angle. Can be used only as a function.

Syntax Tan(angle)

Parameters

angle The angle, in radians, for which to calculate the tangent.

Returns The tangent of an angle. The return value is single-precision if the angle is an integer or single-precision value, or double precision for a long or double-precision value. The angle is specified in radians, and can be either positive or negative.

Example Find the tangent of $\pi/6$.

```
value=tan (3.14159265359/6)
```

Text statement

Sets up line(s) of text in a dialog box.

Syntax Text x, y, dx, dy, text\$

Parameters

x, y Set the position of the upper left hand corner of the text area relative to the upper left corner of the dialog box.

dx, dy Set the width and height of the text area.

text\$ Contains the text that appears to the right of the position designated by the x/y coordinates. If the width of this string is greater than dx, the spillover characters wrap to the next line. This continues as long as the height of the text area established by dy is not exceeded. Excess characters are truncated.

Comments The Text statement can be used only between a Begin Dialog and an End Dialog statement.

By preceding an underlined character in text\$ with an ampersand (&), you enable a user to press the underlined character on the keyboard and position the cursor in the combo or text box defined in the statement immediately following the Text statement.

Example Text 10,10,180,20 "This is my text"

TextBox statement

Creates a box, used within a dialog box, in which the user can enter and edit text.

Syntax TextBox x, y, dx, dy, .field

Parameters

<code>x, y</code>	Set the position of the upper left hand corner of the text box relative to the upper left corner of the dialog box.
<code>dx, dy</code>	Set the width and height of the text area. A dy value of 12 will usually accommodate text in the system font.
<code>.field</code>	The name of the dialog record field that will hold the any text entered in the text box. When the user selects the OK button, or any pushbutton other than cancel, the text string entered in the text box is recorded in the field.

Comments The TextBox statement can only be used between a Begin Dialog and an End Dialog statement.

Example `Textbox 10,10,130,20 .MyData`

Time\$ function

Gets the current system time as a string. Can be used only as a function.

Syntax `Time$`

Returns A string representing the current time.

Comments The Time\$ function returns an eight-character string. The format of the string is

`"hh:mm:ss"`

where hh is the hour, mm is the minutes, and ss is the seconds. The hour is specified in military style and ranges from 0 to 23.

Example `CurrentTime$ = Time$`

Timer function

Returns the number of seconds that have elapsed since midnight. Can be used only as a function.

Syntax `Timer`

Returns The number of seconds that have elapsed since midnight.

Comments The Timer function can be used in conjunction with the Randomize statement to seed the random number generator.

Example Use the number of seconds since midnight to seed the random number generator.

`Randomize Timer`

Type statement

Declares a user-defined type which can then be used in the DIM statement to declare a record variable. Such a user-defined type is also sometimes referred to as a record type or a structure type.

Syntax

```
Type userType
    field1 As type1
    field2 As type2
End Type
```

Parameters

Fieldn	Each field for which to declare a type.
Type n	A valid data type. (See “Variable data types” on page 222 for a list of data types.)

Comments Between the Type and Type End you may define a number of elements known as fields. Each field can be a string (either dynamic or fixed), number (integer, long, single or double), or a previously-defined record type; it cannot be an array. However, arrays of records are allowed.

The Type statement is not valid inside a procedure definition.

To access the fields of a record, use notation of the form:

```
recordName.fieldName.
```

To access the fields of an array of records, use notation of the form:

```
arrayName( index ).fieldName
```

Example

```
Type car
engine size as integer
number of cylinders as integer
color as string
make as string
model as string
End Type
Dim Honda as car
Honda.Color = "red"
```

UBound function

Determines the largest valid index for a particular dimension of an array. Can be used only as a function.

Syntax `UBound ( arrayVariable [, dimension ] )`

Parameters

`array` The name of the array to check.
`dimension` The number of the dimension to check.

Returns The upper bound of the subscript range for the specified dimension of the `arrayVariable`.

Comments The dimensions of an array are numbered starting with 1. If the dimension is not specified, 1 is used as a default.

LBound can be used with UBound to determine the length of an array.

Example The following example sets the last element of this array to an end marker.

```
Data$ (1, UBound (Data,2)) = "**End"
```

UCase\$ function

Converts the characters of a string to upper case. Can be used only as a function.

Syntax `UCase$ ( string$ )`

Parameters

`string$` The string to convert.

Returns A copy of the source string, with all lower case letters converted to upper case. The translation is based on the country specified in the Windows Control Panel.

Example The following example returns the value "STOP."

```
Upper_case = UCase$("Stop")
```

Val function

Converts a string to a number. Can be used only as a function.

Syntax `Val( string$ )`

Parameters

`string$` The string or string field to convert.

Returns Returns a numeric value corresponding to the first number found in the specified string. Spaces in the source string are ignored. If no number is found, 0 is returned.

Example The following example turns a string into a number and sets the number variable to that result.

```
Num_var = Val("123.4")
```

While ... Wend

Controls a repetitive action.

Syntax

```
While condition  
statementblock  
Wend
```

Comments The condition is tested, and if TRUE, the statement block is executed. This process is repeated until the condition becomes FALSE.

The While statement is included in the macro language for compatibility with older versions of Basic. The Do statement is a more general and powerful flow control statement.

Example

```
While InputBox$ ("Type the secret word") <> "Secret"  
MsgBox "That's not it"  
Wend
```

Write statement

Writes data to a sequential file. The file must be opened in output or append mode.

Syntax Write [#] filenumber% [,expressionlist]

Parameters

filenumber% An integer expression identifying the open file to write to.
expressionlist Specifies one or more values to be written to the file.

Comments An expression must be string and/or numeric expressions, separated by commas. If expressionlist is omitted, the Write statement writes a blank line to the file. (See "Input\$ function" on page 323 or "Input # statement" on page 323.)

Example Write #1, A\$, B

Index

A

Abs function 291
Access 12
ActiveTitle\$() function 234
AddGroup method 266
AddMenu command 236
AddSort method 267
AddSummary method 268
AddTable method 269
aliases
 column aliases 43
 field aliases in crosstabs 152
aligning
 Align Toolbox 111
 characters 95
 objects 110–117
 text 117–118
AllDataBases\$ property 284
AllOwners\$ property 285
AllTables\$ property 285
application events 225–228
Asc function 291
Assert statement 292
assigning aliases 203
assigning column aliases 43
Atn function 292
Avery label styles 91

B

BASIC conventions
 application data types 224
 arrays 223
 conversions 224
 dialog box records 224
 numbers 223
 records 223
 strings 223
 trappable errors 225
Beep statement 293
Best Fit feature 96
bitmaps (.BMP files) 125
Btrieve 13
building a report
 adding
 graphs 127
 tables 39
 without using a join 41
 choosing
 columns 41
 assigning aliases 43
 visually 43

 with a dialog box 42
 report styles 35
 report types 34
 creating crosstabs 136
 entering text 100
 formatting 85–124
 grouping 53
 after loading data 53
 on the server 55
 joining tables 39
 using dialog boxes 40
 visual joins 39
 linking to a macro 163
 report query, description
 of 37
 sorting 52
 using
 direct SQL 43
 Report Query dialog
 boxes 37
 the ribbon 52
 the toolbar 52
 with direct SQL 47
ButtonGroup statement 294

C

\$CStrings metacommand 303
Call statement 295
calling applications from
 ReportSmith 213
CancelButton statement 296
Caption statement 297
CDbl function 297
character sizing 94
ChDir statement 297
ChDrive statement 298
CheckBox statement 298
choosing
 columns
 for use in query only 43
 value only 43
 defined connections 39
 parts of a report 85
 records 59
CInt function 299
CLng function 300
Close statement 300
CloseReport command 236
CloseRS command 237
columns
 adjusting width 96
 choosing 41
 column mode 86, 96
 excluding automatically 42
 form mode 86
 including automatically 42
 inverting in crosstabs 151
 moving 86, 96
 using Best Fit 96
 selecting 86
 blocks of 42
 fields in 87
 multiple 87
 visually 43
ComboBox statement 301
Commit method 270
Connect command 237
connection types
 IDAPI 9
 native 10
 ODBC 10
connections
 creating 8
 defined 7
 deleting 8
 saving 7, 8
 storing 188
 to graphics files 189
Const statement 301
conventions
 keyboard 3
 text 3
converting
 a columnar report to a
 graph 127
 to SQL 46
copying a view 206
copying data between
 applications 219
Cos function 302
CreateDdeItem statement 238
CreateDdeTopic statement 239
Creating a simple macro 162
Creating and working with
 macros 162
crosstabs
 aliases 152
 based on a columnar
 report 137
 calculating fields 151
 columns 144
 creating by selecting
 columns 138
 creating from scratch 136
 dates 148

- deleting items 149
- field aliases 152
- grouping 148
- inverting rows and columns 151
- manipulating data 148
- modifying 140
 - values 143
- parts of 132
- positioning 138
- reports 145
- resizing rows and columns 145–148
- rows 142
- selecting items in 139–140
- sorting 150
- styles 155
- types of 132–135
- values 143
- CSng function 302
- CurDir\$ function 303
- Current command 239
- CurrentPage Function
 - command 240
- customizing ReportSmith 185
 - alphabetically listing items 188
 - connection file 188
 - Dynamic Data Access 187
 - fonts 188
 - menu and toolbar 189
 - picture file search path 189
 - rulers 188
 - start-up options 187

D

- data
 - bringing into a report 37
 - copying 219
 - grouping 53
 - manipulating in a report 51
 - sorting 52
- data dictionary
 - adding tables 201
 - assigning aliases 203
 - creating a view 199
 - headings 205
 - overview 198
 - tutorial 207
 - using from ReportSmith 206
 - workplace 199
- data sources
 - connecting to 9
 - diagram 9
 - PC through IDAPI 11–12
 - PC through ODBC 12–24

- SQL through IDAPI 24
- SQL through native connections 24–30
- supported (table) 10
- supported list of 10
- DataBase\$ property 285
- databases
 - Access 5
 - Btrieve 5
 - connecting to 5
 - DB2 5
 - dBASE 5
 - defining connections 39
 - deleting connections 8
 - Excel 5
 - FoxPro 5
 - Informix 5
 - INGRES 5
 - InterBase 5
 - ODBC drivers 5
 - ORACLE 5
 - Paradox 5
 - Raima 5
 - saving
 - connection information 7
 - connections to 8
 - SQLBase 5
 - SQLServer 5
 - storing connections 188
 - Sybase 5
 - Teradata 5
 - Unify 5
 - used with ReportSmith 5
 - Watcom SQL 5
- DataSet control 171
 - methods 173
 - properties 172
 - scope 171
- date formats 97
- Date\$ function 304
- Date2Str command 240
- DateField command 241
- dates
 - creating custom 100
 - formats for 97
 - formatting columns 98
 - in crosstabs 148
- dBASE 11
- dBASE/xBASE 15
- DdeExecute command 241
- DdePoke command 242
- DdeRequest command 242
- Declare statement 304
- defining connections 39
- Deftype statement 306
- deleting graphs 128
- Derived fields

- macro-derived 68
- sample macro derived field 69
- derived fields
 - creating 66
 - macro-derived fields 66
 - SQL-derived fields 66
 - uses for 67
- DerivedField command 243
- Dialog boxes
 - Macro Commands 162
- Dialog statement 307
- Dim statement 307
- Dir\$ function 309
- Disconnect method 271
- Do...While statement 310
- DoEvents command 243
- drawing objects 118
- Dynamic Data Access
 - options 187
- Dynamic Data Exchange (DDE) 211
 - application names 212
 - conversations 212
 - items 213
 - synchronization 214
 - topics 213

E

- EnableIcon command 244
- EnableMenu command 244
- EnableRMenu command 245
- entering text 100
- Environ\$ function 311
- Eof function 312
- Erl function 312
- Err function 313
- Err statement 313
- Error statement 313
- Error\$ function 314
- Error\$ property 285
- Excel 17
- ExecSQL command 246
- ExecuteMenu command 245
- Exit statement 314
- Exp function 315

F

- field labels 106
 - formatting 97
- Field\$ command 247
- Field\$ method 271
- FieldFont command 248
- fields

- calculating in crosstabs 151
- creating 66, 71
- creating aliases 204
- data fields, inserting 103
- derived fields, inserting 104
- form mode 86
- inserting 103
- inserting labels for 106
- macro-derived 66
- moving 86
- numeric 99
- renaming in a data
 - dictionary 204
- report variables 104
- resizing field height 108
- selecting 87
 - field labels 85
- SQL-derived 66
- summary fields 105
- suppressing in groups 55
- system fields 105
- FieldText command 248
- FileAttr function 315
- files installed 3
- Fix function 316
- For...Next statement 317
- formatting
 - adding borders 101
 - bitmaps 125
 - for background, list of 92
 - characters 94
 - creating styles 121
 - crosstabs 154–158
 - dates 97, 100
 - default report styles 121
 - drawing objects 118
 - extracting styles 123
 - field labels 97
 - fonts 94
 - inserting fields 103
 - labels 91
 - margins 90
 - modifying styles 122
 - null values 99
 - numbers 97
 - numeric fields 99
 - orientation 90
 - paginating 110
 - patterns 102
 - point size 94
 - report styles 120
 - resizing objects 88
 - ribbon 95
 - setting
 - margins 89
 - paper size 90
 - shadow borders 102
 - techniques 93

- time formats 98
- using
 - page setup 89
 - the grid 93
 - the ribbon 52
- wallpaper 92
- FoxPro 19
- FreeFile function 318
- Function...End Function statement 318

G

- GetAllField\$ method 272
- GetAllFields\$ method 249
- GetColumnAlias\$ method 272
- GetDataSources\$ method 249, 273
- GetDdeItem function 250
- GetField\$ function 319
- GetFieldList\$ method 273
- GetFieldName\$ function 250
- GetGroup\$ method 274
- GetIncludePath\$ command 250
- GetNext command 250
- GetPrevious command 251
- GetRandom command 251
- GetRecordLimit function 252
- GetRepVar command 252
- GetSort\$ method 274
- GetSQL command 252
- GetSQL\$ method 275
- GetSummary\$ method 275
- GetTable\$ method 275
- GetTableAlias\$ method 276
- GetTableLink\$ method 276
- Global statement 320
- GoTo statement 320
- graphics
 - bitmaps (.BMP files) 125
 - pointing to files 189
- graphs
 - converting a columnar report 127
 - creating 127
 - deleting 128
 - modifying 128
 - positioning 128
- grid 93
- GroupBox statement 321
- grouping 53
 - after loading data 53
 - crosstabs 148
 - database grouping 55
 - inserting a header or footer 57

- on the server 55
- report grouping 53
- suppressing fields 55

H

- headers and footers
 - inserting 57
 - list of types 107
 - placing items in 107
- headings, adding to data dictionary 205
- Hex\$ function 321
- hiding a report section 109
- hWin_Active() function 253
- hWin_RS() function 253

I

- \$Include metaccommand 323
- icons, running reports from 191
- Id property 286
- If...Then...Else statement 322
- IncludeFields\$ method 276
- Informix 24
- Ingres 25
- Input # statement 323
- Input\$ function 323
- InputBox\$ function 324
- installation
 - files installed 3
- InStr function 325
- Int function 326
- InterBase 24
- Introduction to macros 159
- IsMenuChecked command 253
- IsMenuEnabled command 254

J

- joining tables 39
 - before loading into a report 40
 - modifying links 40
 - removing a join 41
 - visually 39

K

- Kill statement 326
- KillMenu command 255

L

- labels
 - Avery label codes 91
 - choosing a report type 32

- field 97
 - setting up 91
- LastLoaded\$() function 255
- LBound function 327
- LCase\$ function 327
- Left\$ function 328
- Len function 328
- Let Assignment statement 329
- Line Input # statement 329
- linking
 - macros to events 163
 - objects to a report 218
 - removing a join 41
 - tables 39, 41
- ListBox statement 330
- Load method 277
- LoadReport command 256
- Lof function 331
- Log function 331
- LTrim\$ function 331

M

- Macro Commands dialog box 162
- Macro derived fields 68
 - sample creation of 69
- Macros
 - creating (simple) 162
 - creating and working with 162
 - introduction 159
 - refreshing a temporary table 180
- macros 221
 - advanced concepts 165–173
 - application events 221, 225–228
 - arguments 166
 - command reference 233
 - DataSet control 171
 - functions and statements 166
 - linking to an event 163
 - loading 165
 - object reference 221
 - report events 230
 - sample macros 173–180
 - saving as a .MAC file 164
 - scoping 160
 - uses for 160
 - using Dynamic Data Exchange 169
- margins
 - setting 89, 90
- master/detail report 33
- Menu commands
 - Tools | Derived Fields 69

- Tools | Macro 162
- menus, customizing 189
- Mid statement 332
- Mid\$ function 332
- MkDir statement 333
- modes
 - column 86
 - form 86
- moving
 - columns 86, 96
 - column mode 96
 - data using the grid 93
 - objects 85–88, 219
- Msgbox function 334
- Msgbox statement 335

N

- \$NoCStrings metacommand 336
- Name statement 336
- Name\$ property 286
- null values 99
- number formats 97

O

- objects
 - canceling links 218
 - creating 217
 - drawing 118
 - editing links 218
 - embedding 217
 - linking and embedding 215
 - linking to a report 218
 - moving 219
 - OLE 2.0 215
- Oct\$ function 336
- OKButton statement 337
- OLE objects 215
- On Error statement 337
- Open statement 338
- Option Base statement 338
- OptionButton statement 339
- OptionGroup statement 339
- options
 - choosing font lists 188
 - customizing 185
 - data access options 187
 - for columns 42
 - listing items
 - alphabetically 188
 - pointing to graphics 189
 - printer 182
 - selecting measurements 188
 - start-up options 187
 - storing connections 188
- ORACLE 26

- overriding a view 206
- Owner\$ property 286

P

- page numbers, inserting 110
- page setup 89
- paginating a report 110
 - repaginate button 108
- paper size, setting 90
- Paradox 12, 20
- parts of a report 51
- point size 94
- Print statement 340
- printing
 - canceling 183
 - choosing a printer 182
 - current page only 182
 - entire report 183
 - options 182
 - paginating a report 110
 - paper size 90
 - range of pages 183
 - setting up the printer 181
- PrintReport command 257

R

- Randomize statement 340
- Recalc method 277
- Recalc statement 257
- Record property 286
- RecordCount command 258
- RecordCount property 287
- records, choosing 59
- ReDim statement 341
- Refreshing a temporary table 180
- Rem statement 342
- RemoveGroup method 277
- RemoveSort method 278
- RemoveSummary method 278
- RemoveTable method 279
- RemoveTableLink method 279
- repaginate button 108
- ReplaceTable method 280
- report events 230
- report query 37
 - Report Query dialog boxes 37
- Report Query Files (.RQF) 83
- report styles
 - applying 120
 - creating 121
 - default 121
 - deleting 123
 - extracting 123
 - modifying 122

- using 120
- report types
 - choosing 34
 - columnar 31
 - crosstabs 32
 - custom 33
 - form 32
 - labels 32
 - master/detail 33
 - summary-only 34
- report variables
 - changing values 75
 - creating 72
 - data types used in 73
 - entry types used in 73
 - inserting 104
 - modifying 75
 - samples of 75–78
- ReportBasic
 - uses for 160
- ReportBasic Macro options
 - in macro-derived fields 69
- Reset statement 342
- resizing
 - crosstab rows and columns 148
 - field and record height 108
 - objects 88
- Resume statement 342
- ResumeEvent command 258
- Rgb command 259
- ribbon 52, 95
- Right\$ function 343
- RmDir statement 343
- Rnd function 344
- rows
 - form mode 86
 - in crosstabs 142
 - inverting in crosstabs 151
 - selecting 87
- RTrim\$ function 344
- rulers 188
- RunMacro command 168, 259
- running a report from an icon 191
- runtime
 - sample DDE code 195
- runtime viewer
 - installing 194
 - setting up 193
 - uses for 193

S

- sample data dictionary 207
- Sample macros

- refreshing a temporary table 180
- samples
 - locations 4
- Save method 280
- scoping macros 160
- Seek function 345
- Seek statement 345
- Select Case statement 346
- selecting
 - a printer 182
 - a report body 88
 - blocks of columns 42
 - field labels 85
 - headers and footers 88
 - multiple columns 87
 - rows 87
 - text 88
- selection criteria 58
- selection criteria, specifying 58
- Selection\$ property 287
- SelectReport statement 260
- sending menu commands 214
- SetColumnAlias method 280
- SetDataFilter statement 260
- SetDdelItem statement 261
- SetField\$ function 347
- SetFromActive method 281
- SetFromLoading method 281
- SetIncludePath command 261
- SetRecordLimit statement 262
- SetRepVar command 260
- SetSQL command 262
- SetTableAlias method 282
- SetTableLink method 282
- setting
 - margins 89
 - paper size 90
- SetUserSQL method 283
- Sgn function 347
- Shell function 348
- ShowRS command 262
- Sin function 348
- Softbridge Basic Language (SBL) 159
- sorting data
 - in crosstabs 150
 - in reports 52
- Space\$ function 349
- SQL
 - browsing tables and columns 48
 - clearing the Browser 49
 - converting to 46
 - definition of 45
 - entering formulas 48

- restrictions for entry 49
- testing statements 47
- using
 - direct SQL entry 44
 - list boxes 49
 - viewing SQL text 45
- SQL Links 10
- SQL Server/Sybase 28
- SQLBase 29
- SQL-derived fields 67
- Sqr function 349
- Stop statement 349
- Str\$ function 350
- StrToDate command 263
- String\$ function 350
- style extractor 123
- styles
 - crosstab 155
 - report 120
- Sub...End Sub statement 351
- SumField command 264
- summary fields
 - creating 71
 - inserting 105
- suppressing a report area 109
- synchronization of DDE 214
- system fields 105

T

- Table\$ property 287
- table-driven values 78
- tables
 - adding 39
 - adding to a data dictionary 201
 - creating temporary tables 82
 - joining 39
 - number of links between 40
 - renaming 204
- Tan function 351
- temporary tables 82–83
- Teradata 29
- TestSelection\$ method 284
- Text 22
- text
 - editing 100
 - entering and formatting 100
 - selecting 88
- Text statement 352
- TextBox statement 352
- time 98
- Time\$ function 353
- Timer function 353
- toolbar 52
 - customizing 189
 - displaying 52

Tools | Derived Fields 69
Tools | Macro 162
topics in this guide 1
TotalPages function 265
Type statement 354

U

UBound function 354
UCase\$ function 355
Unify 30

V

Val function 355
values in crosstabs 143
view
 choosing a view parent 201
 copying 206
 creating 199
visual joins 39

W

While...Wend syntax 356
width, adjusting columns 96
Write statement 356

Z

zero values 99



Creating Reports



VERSION 2.5

Borland®
ReportSmith™

Borland International, Inc., 100 Borland Way
P.O. Box 660001, Scotts Valley, CA 95067-0001

Redistributable files

You can redistribute the following files in accordance with the terms of Borland's No Nonsense License Statement:

RED110.DLL	DRVACSS.HLP	BTRV110.DLL
DRVBTRV.HLP	RCS0.RST-RCS8.RST	RSCTSFMT.RST
DRVDBASE.HLP	XBS110.DLL	RS_DBLIB.DLL
XLSISAM.DLL	DRVEXCEL.HLP	DRVFOX.HLP
RS_GUP.DLL	IDAPI01.DLL	IDAPICFG.EXE
IDASCI01.DLL	IDBAT01.DLL	IDDBAS01.DLL
IDODBC01.DLL	IDPDX01.DLL	IDQRY01.DLL
IDR10009.DLL	ILD01.DLL	IDAPI.DNF
IDAPICFG.HLP	RS_INGR.DLL	B4RPT.MAC
DONDFMT.MAC	DATE.MAC	DISABLE.MAC
ENABLE.MAC	GREETING.MAC	ID2NAME.MAC
LOADREP.MAC	LOADREPS.MAC	RECNO.MAC
THETIME.MAC	CTL3D.DLL	MSJETDSP.DLL
ODBC.DLL	ODBCADM.EXE	ODBCINST.DLL
SIMADMIN.DLL	SIMBA.DLL	ODBCINST.HLP
COPOBJ.DLL	OLE2.DLL	OLE2CONV.DLL
OLE2DISP.DLL	OLE2NLS.DLL	OLE2PROX.DLL
STORAGE.DLL	TYPELIB.DLL	OLE2.REG
STDLE.TLD	RS_ORA6.DLL	RS_ORA7.DLL
BIBAS04.DLL	BIFLT04.DLL	BIMDS04.DLL
BIUTL04.DL	ODBCCURS.DLL	PXENGCFG.EXE
PXENGWIN.DLL	BIPDX04.DLL	BIPDX04.HLP
QEBI.LIC	README.TXT	BC30RTL.DLL
DATALIB2.DLL	DLOLE2.DLL	IMAGEMAN.DLL
IMGPCX.DIL	IMGTIFF.DIL	RSSQLIF.TXT
RS_FMT.DLL	RS_IDABP.DLL	RS_ODBC.DLL
RS_OGLWL.DLL	RS_OLE2C.DLL	RS_OLE2U.DLL
RS_OWL31.DLL	RS_SIFUT.DLL	RS_SQLIF.DLL
RS_SYS.DLL	RS_SYS.DLL	RS_TBP1.DLL
RS_TBP2.DLL	RS_TCL31.DLL	RS_TKRIB.DLL
RS_DFWEN.DLL	RS_TKSTB.DLL	RS_TKTB.DLL
RS_UTIL.DLL	SSMRTHEAP.DLL	REPVAR.FRM
REPVAR.MAK	RSR_CTAB.DLL	RSR_DBAC.DLL
RSR_DBUI.DLL	RSR_MAIN.DLL	RSR_RPT.DLL
RSR_RUTL.DLL	RSR_XPRT.DLL	RSTEST.FRM
RSTEST.FRM	RS_RUN.EXE	RS_RUN.HLP
RUN_TEST.EXE	RS_SQLIF.INI	CTL3DV2.DLL
TXTISAM.DLL	DRVTEXT.HLP	

Borland may have patents and/or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents.

COPYRIGHT © 1995 Borland International. All rights reserved. All Borland product names are trademarks or registered trademarks of Borland International, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.

Printed in the U.S.A.

1E0R195

9596979899-9 8 7 6 5 4 3 2 1

W1

Contents

Introduction	1	Connecting to SQL databases through IDAPI	24
Who should use this guide	1	InterBase	24
Topics covered in this guide.	1	Informix	24
What you can do with ReportSmith	2	Connecting to SQL databases through native connections.	24
Terms, techniques, and conventions	3	DB2.	25
The keyboard	3	Ingres	25
Text	3	Hardware and software requirements.	25
Procedures	3	ORACLE	26
Installed files	3	Hardware and software requirements.	26
Directory locations for files	4	ORACLE connection issues.	27
Sample tables, reports, and macros	4	SQL Server/Sybase	28
		Hardware and software requirements.	28
		SQL Server/Sybase connection issues.	29
		SQLBase.	29
		Teradata	29
		Unify.	30
Chapter 1		Chapter 2	
Connecting to a data source	5	Types of reports you can create	31
Databases ReportSmith can access	5	Default report types.	31
Connecting to tables	5	Columnar	31
Saving database connection information	7	Crosstab	32
Connecting to data sources	9	Form	32
Connection types	9	Labels	32
Through IDAPI	9	Custom report types	33
What are SQL Links?	10	Master/detail.	33
Through ODBC	10	Summary-only	34
Through native API	10	Choosing a report type.	34
Supported data sources	10		
Connecting to PC data sources through IDAPI	11	Chapter 3	
dBASE	11	Bringing data into a report	37
Paradox.	12	Using Report Query dialog boxes	37
Connecting to PC data sources through ODBC	12	Adding tables	39
Access	12	Joining tables.	39
Hardware and software requirements	12	Visual joins	39
Access connection issues	13	Using the Create New Table Link dialog box	40
Btrieve	13	Choosing columns	41
Hardware and software requirements	14	Using the Table Columns dialog box	42
Btrieve connection issues	15	Visually	43
dBASE/xBASE	15	Assigning column aliases.	43
Hardware and software requirements	16	Using direct SQL entry	44
dBASE connection issues	17	What is SQL?	45
Excel	17	Viewing SQL text	45
Hardware and software requirements	17	Converting a report to SQL text entry	46
Excel connection issues	18	Creating a report with direct SQL entry	47
FoxPro	19	Using the Table and Column Browser.	48
Hardware and software requirements	19	Using the SQL list boxes.	49
FoxPro connection issues	20	Restrictions for SQL text entry	49
Paradox.	20		
Hardware and software requirements	21		
Paradox connection issues	21		
Text	22		
Hardware and software requirements	22		

Chapter 4	
Manipulating data	51
Parts of a report	51
The toolbar	52
The ribbon	52
Sorting	52
Grouping	53
Report grouping	53
Database grouping	55
Inserting a header or footer	57
Specifying selection criteria	58
Choosing records	59
Choosing fields	63
Creating a derived field	66
SQL-derived fields	67
Macro-derived fields	68
Sample macro-derived field	69
Creating a summary field	71
Creating a report variable	72
Data types and entry types	73
Sample report variables	75
Sample 1: Entering or choosing a value	75
Sample 2: Choosing from a list	77
Sample 3: Choosing table-driven values	78
Creating a master /detail report	78
Naming a detail report	79
Using the Detail Sets dialog box	80
Displaying field labels	80
Inserting a field into a report section.	81
Formatting a report section	81
Saving a report	81
Creating a temporary table	82
Adding a report background.	92
Using the grid	93
Formatting techniques	93
Characters.	94
Fonts and point sizes.	94
Aligning characters in columns	95
Positioning columns.	96
Field labels	97
Numbers and dates	97
Customizing numeric fields	99
Leaving null values blank	99
Entering and formatting text	100
Adding borders	101
Inserting fields.	103
Data fields.	103
Derived fields.	104
Report variables	104
System fields	105
Summary fields	105
Inserting field labels	106
Placing items into a header or footer	107
Resizing field and record height.	108
Using Shrink and Grow	108
Specifying row height	108
Choosing items to display and print.	109
Hiding a section of a report	109
Paginating a report	110
Aligning objects	110
The Align toolbox	111
Aligning text	117
Aligning left	117
Aligning center	118
Aligning right	118
Drawing	118
The Drawing Toolbox	119
Using report styles	120
Applying a style	120
Assigning a default style	121
Creating a style.	121
Modifying a style	122
Deleting a style	123
Using the Style Extractor	123
Displaying data fields as pictures.	124
Chapter 6	
Graphs	127
Converting a columnar report to a graph	127
Converting a crosstab to a graph	129
Placing a graph	129

Chapter 7		
Crosstabs	131	
Parts of a crosstab	132	
Types of crosstabs	132	
Value-only	132	
Single-column	133	
Single-row	133	
Single-row and single-column	133	
Double-column	134	
Double-row	135	
The Crosstab menu	135	
The Crosstab toolbox	135	
Creating a crosstab	136	
Method 1: From scratch	136	
Method 2: Based on a columnar report	137	
Method 3: Selecting three columns	138	
Placing a crosstab	138	
Expanding a boundary	139	
To cancel creating a crosstab	139	
Selecting crosstab items	139	
Modifying a crosstab	140	
Rows	142	
Values	143	
Columns	144	
Crosstab report options	145	
Resizing rows and columns	145	
Individual rows	146	
Individual columns	147	
Uniform row and column sizing	148	
Manipulating data	148	
Grouping	148	
First Month of Year	149	
Format	149	
Deleting rows, columns or values	149	
Sorting	150	
Inverting rows and columns	151	
Calculating fields	151	
Creating a field alias	152	
Formatting techniques	154	
Working with crosstab styles	155	
Applying a style	155	
Creating a style	156	
Extracting a style	156	
Chapter 8		
Macros	159	
Introduction to macros	159	
What is ReportBasic?	159	
Uses for macros	160	
Scoping macros	160	
Global macros	161	
Report macros	161	
Using macros with other applications	161	
Creating and working with macros	162	
Creating a simple macro	162	
Linking a macro to an event	163	
Saving a macro to a .MAC file	164	
Loading a macro	165	
Advanced macro concepts	165	
Commands as functions or statements	166	
Using arguments	166	
Arguments and events	167	
Calling other macros	168	
The RunMacro command	168	
Using macros with DDE: Two examples	169	
Example 1: Loading a report through SQL Windows	169	
Example 2: Assigning a value to a report variable 170		
Using the Creation event	170	
Suppressing a group footer	171	
Using the DataSet Control	171	
Creating a DataSet Control Object	171	
Modular scope	171	
Local scope	172	
Global Scope	172	
Properties and methods	172	
Properties	172	
Methods	173	
Sample macros	173	
Example 1: Conditional formatting	174	
Example 2: Setting report variables	174	
Example 3: Loading a series of reports	176	
Example 4: Connecting to a database	176	
Example 5: Displaying a greeting	176	
Example 6: Creating a counter field	177	
Example 7: Determining a percent of total	177	
Example 8: Creating a summary	178	
Example 9: Sample dialog box	179	
Example 10: Refreshing a temporary table	180	
Chapter 9		
Printing a report	181	
Setting up the printer	181	
Selecting the printer	182	
Printing a report	182	
Chapter 10		
Customizing ReportSmith	185	
Setting options	186	
Controlling data access	187	
Choosing start-up options	187	
Selecting measurements	188	

Listing items alphabetically	188	Linking objects to a report	218
Choosing a font list	188	Dragging data between applications	219
Storing connections	188		
Pointing to graphics files	189	Appendix B	
Customizing the menu and toolbar	189	Macro reference	221
Disabling File New	189	Macro object reference	221
Adding a custom menu item	191	Basic conventions	221
Running a report from an icon	191	Variable data types	222
		Application events	225
Chapter 11		Creation Events	229
Runtime Viewer	193	Report Events	230
Overview	193	Data field events	232
Installing the Runtime Viewer	194	Command reference	233
Using Runtime Viewer	194	Data types and type conversion	233
Exporting data	195	ReportBasic command summary	234
Setting options	195	ActiveTitle\$ function	235
Sample DDE code	195	AddMenu	236
		CloseReport	236
Chapter 12		CloseRS	237
Creating and using a data dictionary	197	Connect	237
Understanding a data dictionary	197	CreateDDEItem statement	238
Overview	198	CreateDDETopic statement	239
The data dictionary workplace	199	Current	239
Creating a view	199	CurrentPage function	240
Choosing a view parent	199	Date2Str	240
Adding tables and fields	201	DateField	241
Creating groups	203	DDEExecute	241
Assigning aliases	204	DDEPoke	242
Creating headings	205	DDERequest	242
Copying a view	206	DerivedField	243
Using a data dictionary from ReportSmith	206	DoEvents	243
Creating a sample data dictionary	207	EnableIcon	244
		EnableMenu	244
Appendix A		EnableRMenu	245
Using ReportSmith with		ExecuteMenu	245
other applications	211	ExecSQL	246
Using DDE	211	ExecuteMenu	247
Initiating DDE conversations	212	Field\$	247
Applications	212	FieldFont	248
Topics	213	FieldText	248
Items	213	GetAllFields\$ method	249
Calling ReportSmith	213	GetDataSources\$ method	249
Calling another application	213	GetDDEItem function	250
Sending menu commands	214	GetFieldName\$ function	250
DDE synchronization	214	GetIncludePath\$ command	250
Using OLE 2.0	215	GetNext	250
Embedding objects in a report	216	GetPrevious	251
Editing an embedded object	217	GetRandom	251

GetRecordLimit function	252	IncludeFields\$	276
GetRepVar	252	Load	277
GetSQL	252	Recalc	277
hWin_Active() function	253	RemoveGroup	277
hWin_RS() function.	253	RemoveSort	278
IsMenuChecked.	253	RemoveSummary.	278
IsMenuEnabled	254	RemoveTable.	279
KillMenu	255	RemoveTableLink.	279
LastLoaded\$ function	255	ReplaceTable	280
List RepVar.	256	Save	280
LoadReport	256	SetColumnAlias	280
PrintReport.	257	SetFromActive.	281
Recalc statement	257	SetFromLoading.	281
RecordCount.	258	SetTableAlias.	282
ResumeEvent	258	SetTableLink	282
Rgb	259	SetUserSQL	283
RunMacro	259	TestSelection\$	284
SetDataFilter statement	260	DataSet Control properties summary.	284
SelectReport statement.	260	AllDataBases\$	284
SetRepVar	260	AllOwners\$	285
SetDDEItem statement.	261	AllTables\$.	285
SetIncludePath command	261	DataBase\$	285
SetRecordLimit statement	262	Error\$	285
SetSQL	262	Id.	286
ShowRS.	262	Name\$	286
Str2Date	263	Owner\$.	286
SumField	264	Record.	286
TotalPages function.	265	RecordCount.	287
DataSet control methods summary	265	Selection\$.	287
AddGroup	266	Table\$	287
AddSort	267	Report Control Object	287
AddSummary.	268	Basic commands summary.	288
AddTable.	269	Abs function	291
Connect.	270	Asc function	291
Disconnect	271	Assert statement.	292
Field\$	271	Atn function	292
GetAllFields\$	272	Beep statement.	293
GetColumnAlias\$.	272	Begin Dialog ... End Dialog statement	293
GetDataSources\$	273	Button statement	294
GetFieldList\$.	273	ButtonGroup statement	294
GetGroup\$.	274	Call statement	295
GetSort\$.	274	CancelButton statement	296
GetSummary\$.	275	Caption statement.	297
GetSQL\$	275	CDBl function	297
GetTable\$.	275	ChDir statement.	297
GetTableAlias\$	276	ChDrive statement	298
GetTableLink\$.	276	CheckBox statement	298

CInt function	299	Let (assignment statement)	329
CLng function	300	Line Input # statement	329
Close statement	300	ListBox statement	330
ComboBox statement	301	Lof function	331
Const statement	301	Log function	331
Cos function	302	LTrim\$ function	331
CSng function	302	Mid statement	332
'\$CStrings metacommand	303	Mid\$ function	332
CurDir\$ function	303	MkDir	333
Date\$ function	304	Msgbox function	334
Declare statement	304	Msgbox statement	335
Deftype statement	306	Name statement	336
Dialog	307	'\$NoCStrings metacommand	336
Dim statement	307	Oct\$ function	336
Dir\$ function	309	OkButton statement	337
Do...While statement	310	On Error statement	337
Environ\$ function	311	Open statement	338
Eof function	312	Option Base statement	338
Erl function	312	OptionButton statement	339
Err function	313	OptionGroup statement	339
Err statement	313	Print statement	340
Error statement	313	Randomize statement	340
Error\$ function	314	ReDim statement	341
Exit statement	314	Rem statement	342
Exp function	315	Reset statement	342
FileAttr function	315	Resume statement	342
Fix function	316	Right\$ function	343
For...Next statement	317	Rmdir statement	343
FreeFile function	318	Rnd function	344
Function ... End Function statement	318	RTrim\$ function	344
GetField\$ function	319	Seek function	345
Global statement	320	Seek statement	345
GoTo statement	320	Select Case statement	346
GroupBox statement	321	SetField\$ function	347
Hex\$ function	321	Sgn function	347
If ... Then ... Else	322	Shell function	348
'\$Include metacommand	323	Sin function	348
Input\$ function	323	Space\$ function	349
Input # statement	323	Sqr function	349
InputBox\$ function	324	Stop statement	349
InStr function	325	Str\$ function	350
Int function	326	String\$ function	350
Kill statement	326	Sub ... End Sub statement	351
LBound function	327	Tan function	351
LCase\$ function	327	Text statement	352
Left\$ function	328	TextBox statement	352
Len function	328	Time\$ function	353

Timer function.	353
Type statement	354
UBound function	354
UCase\$ function.	355
Val function	355
While ... Wend	356
Write statement	356

Index 357