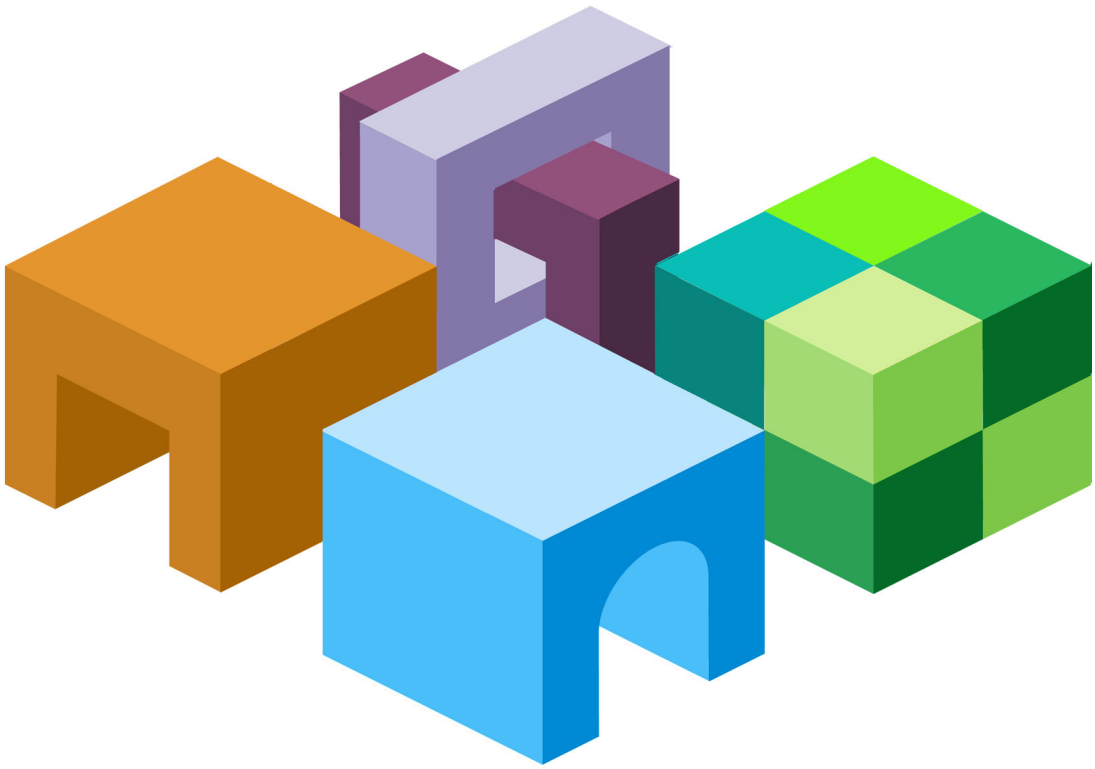


HYPERION® SQR® PRODUCTION REPORTING ACTIVATOR

RELEASE 11.1.1

USER'S GUIDE

ORACLE®
ENTERPRISE PERFORMANCE MANAGEMENT SYSTEM

SQR Production Reporting Activator User's Guide, 11.1.1

Authors: EPM Information Development Team

Contents

1 Introduction

In This Chapter

About SQR Production Reporting Activator

Oracle's Hyperion® SQR® Production Reporting Activator is a set of ActiveX controls that allow you to runOracle's Hyperion® SQR® Production Reporting programs launched from within an application, and to view and print the output from within the same application.

Table 1 ActiveX Controls

ActiveX Control	Description
Production Reporting ActiveX Control	RunsProduction Reporting programs on the local PC or on remote servers.
Production Reporting Viewer ActiveX Control	Allows you to view the report output. You can also print the report or send it to other people.
Production Reporting Print ActiveX Control	Prints the report output.

Other software components include a Visual Basic Sample Program and online help.

Prerequisites for Using SQR Production Reporting Activator

The minimum prerequisites required to use SQR Production Reporting Activator include:

- Production Reporting on the remote server.

  Required to run Production Reporting programs remotely from the Production Reporting ActiveX Control.

- TCP/IP access from the PC hosting the Production Reporting ActiveX Control to the server remotely running Production Reporting.

Required to run Production Reportingprograms remotely from the Production Reporting ActiveX Control.

- Application Development Environments such as Visual Basic for embedding SQR Production Reporting Activator controls.

Installing SQR Production Reporting Activator

The installation registers all Production Reporting ActiveX controls. If registration of any control fails, you must register them manually before using or running the Visual Basic sample program.

➤ To register Production Reporting ActiveX controls manually:

1 **Terminate all the running programs and restart Windows.**

This prevents file-sharing violations.

2 **Go to** `Hyperion\products\biplus\bin\SQR\Activator\bin` **and run** `reg_em.bat`.

This registers the threeSQR Production Reporting Activator ActiveX Controls.

To run SQR Production Reporting Activator remotely, you must configure the remote server account. If the remote server runs on Windows, ensure that FTP and REXEC services are installed on the server machine. (Since Windows does not ship with the REXEC utility, you need to install the Production Reporting Remote program.)

Redistribution ofSQR Production Reporting Activator Controls

To distribute your application using SQR Production Reporting Activator ActiveX controls, you must redistribute the SQR Production Reporting Activator ActiveX controls to the client machines as part of your application. This is accomplished by installing theSQR Production Reporting Activator Redistribution procedures from the SQR Production Reporting Activator media.

Refer to the license agreement to resolve licensing questions.

2 SQR Production Reporting Activator ActiveX Control

In This Chapter

Using the Production Reporting ActiveX Control Local

The Production Reporting ActiveX Control runsProduction Reporting programs locally on a client machine and remotely on a server machine. An example of a server machine could be your database server using Production Reporting.

To use the Production Reporting ActiveX Control `Local` to run Production Reporting programs locally on your PC, make a call to the `LocalRun` or `LocalRunNoWait` methods of the ActiveX Control instance. These methods take one argument: the Production Reporting command line. The Production Reporting command line includes the name of the Production Reporting program to execute along with database connectivity and other Production Reporting command line options. In Visual Basic for example,

```
CommandLine - "report.sqr acctg/acctg@T:sun:orac7 -keep"
Sqr0cx.LocalRun(CommandLine)
```

`LocalRun()` loads the SQR.DLL from the locally installed Production Reporting product, runs the Production Reporting program specified in the command line, waits until it finishes, and returns its return code. `LocalRunNoWait()`, launches SQR.EXE from the locally installed Production Reporting product and passes on the command line.

`LocalRun` and `LocalRunNoWait` look for the Production Reporting executables by checking the SQR.INI in the operating system's *system* directory. Since Production Reporting is available in various versions supporting all major RDBMSs on the market, SQR.INI has different sections for each version installed, such as [Environment:Oracle], [Environment:ODBC], and so on. There is also a common section, [Environment:Common], for database-independent parameters.

The Production Reporting ActiveX Control has a property, `DatabaseName`, that guides `LocalRun` and `LocalRunNoWait` to locate the executables. If `DatabaseName` is valid, executables pointed to by the SQRDIR entry in that section are picked up; otherwise, [Environment:Common] is used. If SQR.DLL is not found, the Production Reporting ActiveX Control fails and throws an error exception.

Troubleshooting Production Reporting Local Functions

Table 2 discusses some problems you may encounter when troubleshooting Production Reporting LOCAL functions and suggested solutions to the problems.

Table 2 Situations and Solutions for Troubleshooting Production Reporting LOCAL Functions

Situation	Solution
`LocalRun` fails, SQR.DLL is not found.	Either SQR.INI is not found in the system directory, or the directory pointed to by SQRDIR in the SQR.INI section [Environment:DatabaseName] does not contain SQR.DLL. Reinstall Production Reporting, and ensure that SQR.INI is in the system directory (such as C:\WINNT) and is updated correctly.
`LocalRunNoWait` fails.	Either SQR.INI is not found in the system directory, or the directory pointed to by SQRDIR in the SQR.INI section

Situation	Solution
	[Environment:DatabaseName] does not contain SQR.EXE. Reinstall Production Reporting for Windows, and ensure that SQR.INI is in the system directory (such as C:\WINNT) and updated correctly.
When running `LocalRun` or `LocalRunNoWait`, the Production Reporting start up dialog box appears.	This occurs because the command line arguments passed to these function calls are not complete. The first two elements must be the Production Reporting program filename and a connection string. If the Production Reporting program does not involve database manipulation or access, use a slash (/) as a placeholder for the connection string, and specify the -xl flag.

Using the Production Reporting ActiveX Control Remote

The Production Reporting `Remote` methods allow an application to connect to a server, submit and run Production Reporting programs remotely, receive the report output, and/or transfer files.

Production Reporting Remote is very useful for running long Production Reporting programs that contain large amounts of data.Production Reporting Remote benefits include:

- Reduced network traffic.
- Server execution that frees the PC for other applications.
- Utilization of the processing power of a server system that is typically greater than that of a local PC.

Table 3 Production Reporting Remote Methods

Method Type	Methods
Connection/Global methods	RemoteConnect (Method) RemoteDisconnect (Method) RemoteIsConnected (Method) RemoteCancel (Method) RemoteSetDebug (Method) RemoteGetShellType (Method) RemoteGetServerType (Method)
Running Production Reporting Program methods	RemoteRun (Method) RemoteRunNoWait (Method) RemoteGetResults (Method)
FTP/REXEC methods	RemoteExec (Method) RemoteExecNoWait (Method) RemotePut (Method) RemoteGet (Method) RemoteDeleteFile (Method)

Method Type	Methods
	RemoteCheckFile (Method) RemoteGetUniqueSqrFile (Method) RemoteGetUniqueSqtFile (Method) RemoteCleanup (Method)

➤ To use Production Reporting Remote:

1. **Use `RemoteConnect` to open a connection to the server.**
2. **Use other Production Reporting Remote methods as needed.**
3. **Use `RemoteDisconnect` to close the connection.**

The following Visual Basic example opens a connection to the server, runs the Production Reporting program REPORT.SQR on the server, and closes the connection.

```
Sqr0cx.RemoteConnect("sun4, "myuser", "mypassword", " ")
Sqr0cx.RemoteRun("report.sqr acctg/acctg")
Sqr0cx.RemoteDisconnect
```

The next example opens a connection to the server, transfers the Production Reporting program to the server, removes any ERR file, runs the Production Reporting program on the server, checks for an ERR file, gets the SPF output file, and closes the connection.

```
' connect to server
Sqr0cx.RemoteConnect("boston", "reports", " ", " ")
' transfer Production Reporting program to server
Sqr0cx.RemotePut("sales.sqr", "sales.sqr", False)
' remove any error file
Sqr0cx.RemoteDeleteFile("sales.err")
' run report on server
Sqr0cx.RemoteExec("sqr sales.sqr tutorial/secret -e
-nolis", "output.txt")
' check if errors file exists, if does error occurred.
If Sqr0cx.RemoteCheckFile("sales.err") Then
' there's an error file...
' get output SPF file
If Sqr0cx.RemoteGet("sales.spf", "sales.spf", True)
' disconnect and close
Sqr0cx.RemoteDisconnect
```

Note:

SQR Production Reporting Activator only supports a single concurrent remote connection.

TCP/IP Connectivity

Production Reporting Remote uses Windows Sockets as a standard API on TCP/IP for its remote connectivity. Production Reporting Remote supports any Windows Sockets compliant TCP/IP package.

Production Reporting Remote relies on standard TCP/IP services, FTP and REXEC, to perform file transfers and remote execution.

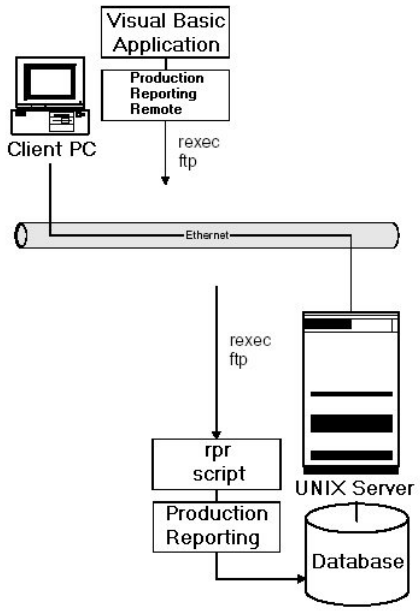
Executing an Production Reporting Program on the Server

Use one of the following two modes to execute anProduction Reporting program on the server. (The modes determine how Production Reporting Remote handles the results of the Production Reporting program.)

- **Production Reporting Remote Synchronous**—Executes anProduction Reporting program on the server, captures any error messages, and notifies you of the completion status. For synchronous mode, use RemoteRun (Method). Control is returned immediately, and you can retrieve the report results later.
- **Production Reporting Remote Asynchronous**—Does not return status information and does not provide notification upon completion or termination of the program. For asynchronous mode, use RemoteRunNoWait (Method)RemoteRunNoWait combined with RemoteGetResults (Method) .

In addition to the methods that run Production Reporting programs and retrieve their results, Production Reporting Remote has general purpose file transfer and remote command methods (such as RemoteGet (Method) and RemotePut (Method). You can use these methods to further customize remoteProduction Reporting execution, as well as exchange files between the local machine and the server.

Figure 1 Production Reporting Remote Process



Running, Saving, and Printing

You can run an Production Reporting program that already resides on the server, or you can upload it and then execute it.

To run an Production Reporting program that already resides on the server, use RemoteExec (Method) or RemoteExecNoWait (Method) with a command line that is valid on the server. The following is valid for UNIX:

```
Sqr0cx.RemoteExec("sqr /usr/tutorial/ex14.sqr tutorial/secret")
```

To save or print the Production Reporting output to a specific location on the server, use the `RemoteExec` (or `RemoteExecNoWait`) method with an appropriate command. For example:

```
Sqr0cx.RemoteExec("cp /usr/tutorial/ex14.spf
/archive/rpts/ex14.spf")
Sqr0cx.RemoteExec("lp /usr/tutorial/ex14.spf")
```

You can spool the output file on the server. All referenced image files must already reside on the server to properly embed graphics in your output file. (See File References in an Production Reporting Program)

File References in an Production Reporting Program

When an Production Reporting program has references to image files or include files, the files must reside on the server when the report executes. If the image and include files are not already on the server, use RemotePut (Method) to copy them to the server before executing the report.

Reference the image or include files in an Production Reporting program with filenames that are recognizable by the server operating system. Do not use the local PC filenames. However, you can write an Production Reporting program so that you can use the specific filenames on both the PC and the server. For example:

```
#if {sqr-platform} = 'WINDOWS'
#include 'C:\SQR\INCLUDE\MYFILE.SQI'
#else
#include '/sqr/include/myfile.sqi'
#endif
```

Cancelling Tasks in Progress

During sometimes lengthy operations such as `RemoteConnect`, `RemotePut`, `RemoteGet`, `RemoteDisconnect`, `RemoteExec`, `RemoteExecNoWait`, `RemoteDeleteFile`, `RemoteRun`, `RemoteRunNoWait`, and `RemoteCheckFile`, you can call `RemoteCancel` to cancel the operation.

When you call `RemoteCancel` while `RemoteExec` waits for the remote command to complete, the connection to the host closes and the waiting terminates. However, `RemoteCancel` does not terminate the remotely executing program, and there is no provision for cancelling remote execution successfully initiated using `RemoteExecNoWait`.

During such operations, Production Reporting Remote frequently yields control back to the window message queue. This means that any application, including the calling application, gains control while the background Production Reporting Remote operation is in progress.

Do not disrupt a running Production Reporting Remote session. In particular, the calling application must not close the window that initiated Production ReportingRemote until Production Reporting Remote is complete.

Security

Production Reporting Remote uses standard database and operating system security. In most cases, you need a username and password to log in. The login data is transmitted to and from the server without any encryption.

Detecting the Success of Remote Execution

For `RemoteExec`, the standard output of the executed program is written to a local PC file. For `RemoteExecNoWait`, the standard output of the executed program is written to a remote file on the server.

For `RemoteRun`, the following three conditions are tested:

- The existence of an SPF file
- The existence of an ERR file
- The existence of the message "Production Reporting: End of Run" in the standard output

If there are no errors, and there is an output (SPF) file or the "Production Reporting: End of Run" message in the OUT file, the report ran successfully.

When running a report using `RemoteRunNoWait`, the outcome is unknown. To obtain the outcome, use the `RemoteGetResults` method. This time, there are three possible states: success, failure, and still running. `RemoteGetResults` checks for the "Production Reporting: End of Run" message in the OUT file. If that message exists, the execution is complete and successful. Otherwise, it checks for the ERR file. If this file exists and is Production Reporting ActiveX Control is not empty, then the execution failed. The *rsqr* script generates an "Production Reporting Remote:Done" message when the report is complete. If the OUT file exists but the "done" message has not been seen, it is assumed that the program is still running. The message REMOTE_ERR_RUNNING is returned.

Debugging

Production Reporting Remote also offers run-time debugging. Using the `RemoteSetDebug` method, you can set debugging for a session to one of three options:

Table 4 Options for Setting a Debugging Session

Option	Description
0	No debugging
1	Production Reporting Remote debugging messages are written to SQREMOTE.LOG in the current directory.
2	TOC-level debugging is turned on in addition to the Production Reporting Remote debugging.

For best results, turn on debugging immediately before the `RemoteConnect` call.

When you enable Production Reporting Remote debugging, messages are written to the SQREMOTE.LOG file in the current directory. If the file already exists, messages are appended to its end. The first message contains a time stamp. These messages show the steps that methods such as `RemoteRun` make. They also show the remote filenames and commands used. Note that some error messages (for example, "file does not exist on the server") are normal.

When you turn on TCP-level, several files with a DBG extension are created. These files contain information regarding the messages that are exchanged with the server. These messages can help you find problems.

Note:

When you turn on debugging, certain items such as passwords are written unencrypted to the log or debug files. Be sure to erase these files when you finish debugging remote operations.

Troubleshooting Production Reporting Remote

Table 5 discusses some problems you may encounter when troubleshooting Production Reporting Remote and suggested solutions to the problems.

Table 5 Situations and Solutions for Troubleshooting Production Reporting Remote

Situation	Solution
An error message such as "File Not Found" displays when anyProduction Reporting Remote methods are called.	Check that the necessary components are accessible. These components include SQRREM32.DLL, PTTCP32C.DLL, and PTFTP32C.DLL. The DLL files must reside within the Windows System directory or somewhere in the path.
An error occurs when attempting to open a connection to a server using the Production Reporting ActiveX Control method `RemoteOpen`.	This typically indicates an invalid hostname, account login, or account password. It may also indicate that the server is down or is not accessible. Use the TCP/ IP *ping* utility to ensure that the server is up and is accessible. Use the TCP/ IP *ftp* utility to ensure that you can log into the server using *ftp*.
An error occurs when attempting to execute a command on a UNIX server.	This typically indicates that an invalid command was used or the permissions on the command are not correct. View the contents of the output file to see what error occurred. The filename of the output file is a parameter passed to the Production Reporting Remote methods `RemoteExec`

Situation	Solution
	and `RemoteExecNoWait`. When using the method `RemoteExec`, the output file is placed on the client PC. When using `RemoteExecNoWait`, the output file is placed on the remote server. An error can also indicate that the UNIX login does not have REXEC permission. Use the TCP/IP *rexec* utility to ensure that you have REXEC permission. This permission must be configured on the server. See your UNIX server documentation for more details, or ask your system administrator.
An error occurs when attempting to run a report remotely.	This typically indicates that server components of Production Reporting Remote are not correctly configured. View the contents of the standard output file OUT (this is not the SPF file) or the error file, if it is exists, to see what error occurred. The filenames of the output file and the errors file are the same as the filename of theProduction Reporting program except that they have the file extension OUT and ERR, respectively. If an error file exists, the Production Reporting program ran but had problems. If an error file does not exist, the program that runs Production Reporting programs could not be started. Another cause of the problem could be that the Production Reporting program (script) running on the remote server requires additional resources, such as bitmaps, include files, and so on, that have not been copied to the server, or the report at runtime can not correctly locate them. When running remotely, it is generally the Production Reporting programmer's responsibility to ensure that all the necessary conditions are fulfilled prior to executing the Production Reporting program.
An error occurs when attempting to copy a file to or from a UNIX server.	This typically indicates an invalid filename or permissions. You must have read permission to the directory on the UNIX server to get files from it. You must have write permission to the directory on the UNIX server to put files into it. You must also have permission to overwrite the local file.

Production Reporting ActiveX Control Properties

TheProduction Reporting ActiveX Control includes the following property:

- DatabaseName (Property)

DatabaseName (Property)

Type	Access
String	Read/write

Description

The database name that Production Reporting ActiveX Control `Local` uses to find the value of SQRDIR, which locates the installed Production Reporting executables. It should be the same that appears in [Environment:DatabaseName] in your SQR.INI file. If this section is empty or invalid, [Environment:Common] is used as a default.

Production Reporting ActiveX Control Methods

Production Reporting ActiveX Control includes the following methods:

- LocalRun (Method)
- LocalRunNoWait (Method)
- RemoteCancel (Method)
- RemoteCheckFile (Method)
- RemoteCleanup (Method)
- RemoteConnect (Method)
- RemoteDisconnect (Method)
- RemoteDeleteFile (Method)
- RemoteExec (Method)
- RemoteExecNoWait (Method)
- RemoteGet (Method)
- RemoteGetResults (Method)
- RemoteGetServerType (Method)
- RemoteGetShellType (Method)
- RemoteGetUniqueSqrFile (Method)
- RemoteGetUniqueSqtFile (Method)
- RemoteIsConnected (Method)
- RemotePut (Method)
- RemoteRun (Method)
- RemoteRunNoWait (Method)
- RemoteSetDebug (Method)

LocalRun (Method)

Return Value

(none)

Arguments

Parameter	Data Type	Meaning
cmdLine	String	Production Reporting command line

Description

Runs an Production Reporting program locally. It uses DatabaseName (Property) to locate and load SQR.DLL, executes the command line, and returns control until Production Reporting is done. The command line must at least contain an Production Reporting program filename and a valued database connection string.

You can include other Production Reporting command line options. A common option is `-keep`, to generate SPF files. Other common options are `-e`*ErrorFile* and `-o`*LogFile*. When you run an SQT (pre-compiled Production Reporting program), use the `-rt` flag. (See "Production Reporting Command-line Flags" in Volume 2 of the *Production Reporting Developer's Guide* for details on the available command-line flags.)

LocalRunNoWait (Method)

Return Value

(none)

Arguments

Parameter	Data Type	Meaning
cmdLine	String	Production Reporting command line

Description

Launches an Production Reporting program locally. It uses DatabaseName (Property) to locate and launch. SQR.EXE, passes the command line, and immediately returns control. The command line must at least contain an Production Reporting program filename and a valued database connection string.

You can include other Production Reporting command line options. A common option is `-keep`, to generate SPF files. Other common options are `-e`*ErrorFile* and `-o`*LogFile*. When you run an SQT (pre-compiled Production Reporting program), use the `-rt` flag. (See "Production Reporting Command-line Flags in Volume 2 of the *Production Reporting Developer's Guide* for details on the available command-line flags.)

RemoteCancel (Method)

Return Value

(none)

Arguments

(none)

Description

Cancels the currently executing background method such as `RemoteConnect`, `RemoteDisconnect`, `RemotePut`, or `RemoteGet`. Once the specified program or command stars running asynchronously, using either the `RemoteRunNoWait` or `RemoteExecNoWait` methods, the `RemoteCancel` method does not cancel it. `RemoteCancel` also does not disconnect from the server. If a currently executing background method exists, the method fails with code REMOTE_ERR_ABORTED when it is cancelled.

RemoteCheckFile (Method)

Return Value

Type	Meaning
Boolean	True—File found. False—File not found

Arguments

Parameter	Data Type	Meaning
remoteFile	String	Pathname of file to check on the remote server.

Description

Checks if a file exists on the remote server. The filename is not case-sensitive.

RemoteCleanup (Method)

Return Value

Type	Meaning
Boolean	True—File found

Type	Meaning
	False—File not found

Arguments

Parameter	Data Type	Meaning
remoteFile	String	Pathname of file to clean up on the remote server.

Description

Deletes the SQR, SPF, OUT, and ERR files associated with the filename given in the `remoteFile` argument.

RemoteConnect (Method)

Return Value

(none)

Arguments

Parameter	Data Type	Meaning
host	String	Host name of the server.
user	String	User name to connect.
pass	String	Password for the user.
account	String	Account to connect (only needed on some systems - leave blank if not used).

Description

Opens a connection to a remote server. The parameters are similar to those used to connect to a remote server using a telnet session. This information is not used for database login.

This operation automatically detects if you are using the Bourne Shell, C Shell, or Korn Shell. The results of this Shell check are used by `RemoteRun` and `RemoteRunNoWait`. It does this by checking the $SHELL environment variable once the connection opens.

While `RemoteConnect` is connected with the server, it yields the CPU to other applications on the PC. You should not close or start anotherProduction Reporting Remote operation while the current operation is in progress.

RemoteDisconnect (Method)

Return Value

(none)

Arguments

(none)

Description

Closes the connection to the remote server. Call `RemoteDisconnect` when your application finishes using Production Reporting Remote.

RemoteDeleteFile (Method)

Return Value

(none)

Arguments

Parameter	Data Type	Meaning
remoteFile	String	Pathname of file to delete on the remote server (use server filename format).

Description

Deletes a file on the remote server using a previously established connection (see RemoteConnect (Method)).

RemoteExec (Method)

Return Value

(none)

Arguments

Parameter	Data Type	Meaning
command	String	The command line parameters to run on the server.
stdoutFile	String	Name of the file on the local machine used to capture the standard output from the specified command.

Description

Executes the specified command on the remote server. This method waits for the command to complete, during which time your PC is in a "hold" state. If you need to continue using your PC after submitting a report for execution, use RemoteExecNoWait (Method) . `RemoteExec` is recommended for running shorter duration Production Reporting programs.

RemoteExecNoWait (Method)

Return Value

(none)

Arguments

Parameter	Data Type	Meaning
command	String	The command line parameters to run on the server.
stdoutFile	String	Name of the file on the remote machine used to capture the standard output from the specified command.

Description

Executes the specified command on the remote server. This method does not wait for the command to complete. Before running this command, use RemoteDeleteFile (Method) to delete any pre-existing output files. Later, use RemoteGet (Method) to retrieve the output.

RemoteGet (Method)

Return Value

(none)

Arguments

Parameter	Data Type	Meaning
LocalFile	String	Pathname of destination file on local client (use DOS format).
RemoteFile	String	Pathname of file to get from remote server (use server format).
Bin	Boolean	Specifies if transfer is in binary or ASCII form. Text files should use ASCII mode, while other files use binary. True—Transfer as binary. False—Transfer as ASCII.

Description

Copies a file from the remote server to the local client (PC) using a previously established connection (see RemoteConnect (Method)). This is similar to using FTP.

RemoteGetResults (Method)

Return Value

Type	Meaning
Boolelan	True—Got result okay. False—The process started by `RemoteRunNoWait` has not finished yet.

Arguments

Parameter	Data Type	Meaning
commandLine	String	The same command line passed to `RemoteRunNoWait`. Specifies the target SPF output file.
Remote ProgramName	String	The temporary name returned by `RemoteRunNoWait`. Specifies the name of the program to check.

Description

Checks for and retrieves the results of an Production Reporting report including the output SPF file, if any, created with `RemoteRunNoWait`.

The outcome (SPF, ERR, and/or OUT file) is downloaded to the PC. The SPF report output file has the same pathname as the specified Production Reporting program file except that it has an SPF extension.

During program execution, Production Reporting errors are trapped, and an error file is created and downloaded to the PC. The name of the error file on the PC is based on the name of the Production Reporting program on the PC, with the file extension changed to ERR.

The standard output of a program executed on the server is generally spooled into a local file. The name of the spool file on the PC is based on the name of the Production Reporting program on the PC, with the file extension changed to OUT.

`RemoteGetResults` returns TRUE when the Production Reporting program successfully completes. Successful report completion is achieved if there are no errors and the message "Production Reporting: End of Run" is found in the OUT file. If completion is successful, the Production Reporting program output is downloaded to the PC.

`RemoteGetResults` returns FALSE when the Production Reporting program is still running. This occurs when there are no errors and no "Production Reporting: End of Run" message is found in the OUT file. When completion is not successful, Production Reporting program output is not downloaded to the PC. After the outcome is successfully downloaded, a cleanup

is automatically invoked to delete all generated files form the server. This occurs for both successful and failed outcomes.

RemoteGetServerType (Method)

Return Value

Type	Meaning
String	"VAX_SERVER", "NT_SERVER", "UNIX_SERVER", or "UNKNOWN_SERVER"

Arguments

(none)

Description

Returns a value that indicates the type of server to which Production Reporting Remote has an open connection.

RemoteGetShellType (Method)

Return Value

Type	Meaning
String	"BOURNE_SHELL", "C_SHELL", "NT_SHELL", "VAX_SHELL", or "UNKNOWN_SHELL"

Arguments

(none)

Description

Returns a value that indicates the type of shell or command interpreter to which Production Reporting Remote has a connection.

RemoteGetUniqueSqrFile (Method)

Return Value

Type	Meaning
String	A unique temporary Production Reporting filename on the server.

Arguments

(none)

Description

Returns the unique filename for an Production Reporting program.Production Reporting Remote methods `RemoteExec` and `RemoteExecNoWait` use this filename to run an Production Reporting program on the server. Using the unique filename to run Production Reporting programs on the server prevents multi-user conflicts. The Production Reporting Remote methods `RemoteRun` and `RemoteRunNoWait` utilize this functionality to prevent multi-user conflicts.

This routine generates a filename with the format SQXXX.SQR, where XXX is a six-digit random number. It also checks that the filename does not already exist on the server. If the file already exists, it retries this operation. If, after five attempts, it still fails to generate a unique filename, it aborts and returns the error code SQRR_ERR_GEN_UNIQUE.

There is a chance that another client PC can generate the same unique identifier; however, the chance is small since the unique identifier includes six digits.

RemoteGetUniqueSqtFile (Method)

Return Value

Type	Meaning
String	A unique temporary SQT filename on the server.

Arguments

(none)

Description

Returns the unique filename for an SQT program.Production Reporting Remote methods `RemoteExec` and `RemoteExecNoWait` use this filename to run an Production Reporting program on the server. Using the unique filename to run Production Reporting programs on the server prevents multi-user conflicts. The Production Reporting Remote methods `RemoteRun` and `RemoteRunNoWait` utilize this functionality to prevent multi-user conflicts.

This routine generates a filename with the format SQXXX.SQR, where XXX is a six-digit random number. It also checks that the filename does not already exist on the server. If the file already exists, it retries this operation. If, after five attempts, it still fails to generate a unique filename, it aborts and returns the error code SQRR_ERR_GEN_UNIQUE.

There is a chance that another client PC can generate the same unique identifier; however, the chance is small since the unique identifier includes six digits.

RemoteIsConnected (Method)

Return Value

Type	Meaning
Boolean	True if connected, False if not connected.

Arguments

(none)

Description

Checks if the connection to the remote server is open and if the underlying TCP/IP connection is still intact.

RemotePut (Method)

Return Value

(none)

Arguments

Parameter	Data Type	Meaning
localFile	String	Pathname of file to send from the local client (use DOS format).
remoteFile	String	Pathname of destination file on the remote server (use server format).
bin	Boolean	Specifies if the file copy is binary or ASCII. Text files should use ASCII, while others should use binary. True—Copy as binary False—Copy as ASCII.

Description

Copies a file from the local client to the remote server using a previously established connection (see RemoteConnect (Method)). This is similar to using FTP.Production Reporting program files (*.SQR) are ASCII.Production Reporting compiled programs (*.SQT) are binary.

RemoteRun (Method)

Return Value

(none)

Arguments

Parameter	Data Type	Meaning
commandLine	String	The Production Reporting command line. In general, the command line needs the report filename, database user id, and database password. It does not need the -keep flag or remote connectivity. Parameters can be passed to the Production Reporting program by placing them at the end of the command line. See "Production Reporting Command-line Flags" in Volume 2 of the *Production Reporting Developer's Guide* for more information about the Production Reporting command line.

Description

Runs the specified Production Reporting report on the remote server. This method copies the Production Reporting report file to the server, runs the Production Reporting report remotely on the server, waits for the report to complete running, retrieves the output SPF file from the server, and performs a cleanup operation.

The `RemoteRun` method generates a unique random name for the Production Reporting program when it is copied to the server. This eliminates contention with other users who may be running a program with the same name using the same directory on the server.

The standard output of a program executed on the server is spooled into a local file. The name of the spool file on the PC is based on the name of the Production Reporting program on the PC, with the file extension changed to OUT.

Production Reporting errors are trapped and documented when they occur. If errors occur, an error file is created and downloaded to the PC. The name of the error file on the PC is based on the name of the Production Reporting program on the PC, with the file extension changed to ERR.

The output SPF file has the same pathname as the specified Production Reporting report file, but with an SPF extension.

When the Production Reporting Viewer is displaying report output, close the report prior to running an Production Reporting program to prevent file sharing violations due to overwriting an open SPF file.

Note:

It is assumed that there is no -F flag or `NewReport` command used in the Production Reporting program, and that the program generates a single report. It is also assumed that there is no -E flag on the command line, and that the Production Reporting command on the server has the extension SQR. If these assumptions are not true, do not use the `RemoteRun` method; instead use the `RemoteGet`, `RemotePut`, and `RemoteExec` methods directly.

The following command line runs the program *mytest.sqr* using the database login *acctg* and the password *acctg*:

```
mytest.sqr acctg/acctg
```

RemoteRunNoWait (Method)

Return Value

Type	Meaning
String	The generated remote program name. Use this as an argument from `RemoteGetResults` and `RemoteCleanup`.

Arguments

Parameter	Data Type	Meaning
CommandLine	String	The Production Reporting command line. In general, the command line needs the report filename, database user id, and database password. It does not need the -keep flag or remote connectivity. Parameters can be passed to the Production Reporting program by placing them at the end of the command line. See"Production Reporting Command-line Flags in Volume 2 of the *Production Reporting Developer's Guide* for more information about the Production Reporting command line.
StdOutFile	String	The temporary name that Production Reporting Remote generates to run it on the server. Pass this output parameter as an input parameter to `RemoteGetResults`.

Description

Runs the specified Production Reporting program on the remote server without waiting for execution to complete. This method copies the Production Reporting program file to the server, runs the Production Reporting report remotely on the server, does not wait for the report to complete running, and returns immediately. Use the `RemoteGetResults` method to check the report run status and to retrieve the report output.

Like the `RemoteRun` method, the `RemoteRunNoWait` method generates a unique random name for the Production Reporting program when it is copied to the server. These temporary files are cleaned up when `RemoteGetResults` is called. This eliminates contention with other users who may be running a program with the same name using the same directory on the remote server.

The SPF report output file has the same path and filename as the specified Production Reporting program file, but the filename has an SPF extension.

As with the `RemoteRun` method, when the Production Reporting Viewer is displaying report output, close the report prior to running an Production Reporting program to prevent file sharing violations due to overwriting of an open SPF file.

Note:

It is assumed that there is no -F flag or `NewReport` command used in the Production Reporting program, and that the program generates a single report. It is also assumed that there is no -E flag on the command line, and that the Production Reporting command on the server has the extension SQR. If these assumptions are not true, do not use the `RemoteRunNoWait` method; instead use the `RemoteGet`, `RemotePut`, and `RemoteExec` methods directly.

The following command line runs the program *mytest.sqr* using the database login *acctg* and the password *acctg*:

```
mytest.sqr acctg/acctg
```

RemoteSetDebug (Method)

Return Value

(none)

Arguments

Parameter	Data Type	Meaning
debugging	Integer	Debugging level: 0 - Turns off debugging. 1 - Output from Production Reporting Remote. 2 - Output from Production Reporting Remote and TCP-level debugging (the underlying TCP/IP provider).

Description

Sets the debugging output level used by Production Reporting Remote. The debugging output level determines if debugging output is generated by the Production Reporting Remote DLL and the TCP-level debugging DLLs that it uses. Debugging output for the Production Reporting Remote DLL is stored in the ASCII text file SQRREM.LOG. Debugging output for TCP-level debugging DLLs is stored in the ASCII text files with the extension DBG. Debugging output from TCP-level debugging DLLs can only be generated by calling `RemoteSetDebug` before the connection is opened.

Advanced Topics

This section discusses advanced Production Reporting topics in the following areas:

- Using SQR Production Reporting Activator with SQT Files
- Running Asynchronously and Checking for Results
- Printing Results from the Server
- Mailing Results from the Server

Using SQR Production Reporting Activator with SQT Files

SQT files are precompiled SQR files that allow portability of Production Reporting programs among platforms. Whether running Production Reporting Local or Remote, Production

Reporting requires that you specify the -rt flag on the command line. For example, to run *somesqtprog.sqt* via Production Reporting/ODBC, enter the following on the command line:

```
Sqr0cx.LocalRun("somesqtprog.sqt MyDSN/MyName/
MyPassword -rt")
```

Running Asynchronously and Checking for Results

When you use `RemoteRunNoWait` to run Production Reporting programs on the server in asynchronous mode, the Production Reporting executable is invoked on the server and control returns immediately to the client application. Thus, the application is free to perform other operations while a long report executes, and to poll the server to see if the Production Reporting program has completed execution.

`RemoteRunNoWait` returns the unique filename of the Production Reporting program invoked on the server. This unique filename prevents multiuser conflicts. Use the `RemoteGetResults` method, and pass the unique filename to perform the "check if complete" operation. If the Production Reporting program has completed execution on the server, TRUE is returned and the SPF output is fetched from the server. You could then display the SPF output using the Production Reporting `Open` method. If the Production Reporting program is still running, TRUE is returned.

It is up to the programmer to implement how an application performs the "check if complete" operation. If made available within an application by the programmer, a user can launch the "check is complete" operation. To do this, add a menu item, toolbar button, button on a form, or some other control to the user interface of the application, and call the method `RemoteGetResults` from the event handler for the control.

If using Visual Basic, use the following code to invoke an Production Reporting program on the server in asynchronous mode. Adjust the values of myserver, mylogin, mypassword, myreport.sqr, dblogin, and dbpasswd to fit your particular environment.

```
Sqr0cx.RemoteConnect("myserver", "mylogin", "mypasswd"," ")
Sqr0cx.RemoteRunNoWait(
"myreport.sqr dblogin/dbpasswd", temp_sqr_file)
Sqr0cx.RemoteDisconnect
```

If using Visual Basic, use the following code to perform the "check if complete" operation. Once again, adjust the values of myserver, mylogin, mypassword, myreport.sqr, dblogin, and dbpasswd to fit your particular environment.

```
Sqr0cx.RemoteConnect("myserver", "mylogin", "mypasswd"," ")
While not Sqr0cs.RemoteGetResults("myreport.sqr dblogin/ dbpasswd"),
'the report is still running, do nothing'
Wend
RemoteDisconnect
```

Printing Results from the Server

You can print the results from an Production Reporting program directly from the server. Printing directly from the server avoids the network overhead required to transfer the results

back to the client for printing. Using asynchronous mode to run reports also frees up the client application to perform other operations.

➤ To print the results from the server:

1 **Log into the server.**

Use a login configured to work with Production Reporting Remote. You should be able to use this login with the `RemoteRun` method.

2 **Create the script with the filename *printsqr.sh* and set its execute permission.**

The script uses the first parameter as the Production Reporting program filename and the second parameter as the database connectivity. The script also uses the $$ value to generate a unique LIS filename to avoid multi-user conflicts. The script also passes to Production Reporting the -f$LIS_FILE parameter to generate the specified LIS file and the -PRINTER:ps parameter to generate Postscript output. Finally, the script uses the UNIXD command lp to print the results.

This example shows a Bourne shell script. Adjust the script as necessary to fit the needs of your particular server environment.

```
#!/bin/sh
. .profile
LIS_FILE=$$.spf
sqr "$1" "$2" -f$LIS_FILE -PRINTER:ps < /dev/null
if [ $? -eq 0 ]; then
Production Reporting ActiveX Control
44 Using Production Reporting Activator
lp $LIS_FILE
fi
rm $LIS_FILE
```

3 **Start theSQR Production Reporting Activator Visual Basic sample program on the client.**

4 **Open a connection to the server using Production Reporting Remote by selecting Remote Connect.**

5 **Copy the sample Production Reporting program *customer.sqr* to the server account by selecting Remote Put under the Production Reporting menu. Then:**

a. Enter *customer.sqr* into the local file and remote file input fields.

b. Select the radio button text transfer and click **OK**.

6 **Execute the following command asynchronously (with no wait) on the server by selecting Remote Exec No Wait under the Production Reporting menu.**

```
printsqr.sh customer.sqr dblogin/dbpasswd
```

Following is the Visual Basic code to invokes the *printsqr.sh* script on the server. Adjust the values of myserver, mylogin, mypasswd, dblogin, and dbpasswd to fit your particular environment. To run an Production Reporting program other than *customer.sqr*, specify the filename of that Production Reporting program. Note how the following code assumes that the Production Reporting program already exists on the server. This avoids copying the Production Reporting program file to the server upon each run and also avoids multi-user conflicts.

```
Sqr0cx.RemoteConnect("myserver", "mylogin", "mypasswd"," ")
Sqr0cx.RemoteExecNoWait("printsqr.sh customer.sqr
dblogin/dbpasswd", "output.txt")
```

`Sqr0cx.RemoteDisconnect`

Mailing Results from the Server

You can use mail to obtain the report results generated when an Production Reporting program is run asynchronously. This avoids having to do a "check if complete" operation when you use asynchronous mode.

You can attach the SPF report as a file to a mail message and mail it out. Since Production Reporting registers the SPF file type in the Windows system registry, many mail programs, such as Outlook, can launch Production Reporting Viewer to view the SPF file attachment.

➤ To use mail to obtain the results of an Production Reporting program:

1 **Log into the server.**

Use a login configured to work with SQR Production Reporting Activator; in particular, a login that you can use with the `RemoteRun` method.

2 **Create the script as the file *mailsqr.sh* and set its execute permission.**

The first parameter to the script is the user ID of the mail account to which you wish to mail the results. The second parameter is the Production Reporting program filename. The third parameter is the database connectivity.

The script uses the $$ value to generate a unique SPF filename and ERR filename to avoid multi-user conflicts. The script passes the -f$SPF_FILE and -e$ERR_FILE parameters to Production Reporting to cause Production Reporting to generate the SPF and ERR files, respectively. The script also uses the UNIX command *uuencode* to uu-encode the generated SPF output file into the mail message. This allows a mail program such as Outlook to display the SPF results as a file attachment. Finally, the script uses the UNIX command mail to send the results using mail.

The following example shows a Bourne shell script. Adjust the script to the needs of your particular server environment.

```
#!/bin/sh
. . profile
SPF_FILE=$$.spf
ERR_FILE=$$.err
sqr "$2" "$3" -f$SPF_FILE -e$ERR_FILE -nolis < /dev/null
if [ $? -eq 0 ]; then
{
echo "attached is report output: $SPF_FILE"
uuencode $SPF_FILE #SPF_FILE
} | mail -s "report $2 run successfully" $1
else
{
echo "error messages follow"
echo
cat $ERR_FILE
} | mail -s "failed to run report $2" $1
fi
```

3 **Start the SQR Production Reporting Activator sample program on the client.**

4 **Open a connection to the server using Production Reporting Remote.**

To do this, select **Remote Connect** under theProduction Reporting menu, enter the login information appropriate for your server login, and click **OK**.

5 **Copy the sampleProduction Reporting program *customer.sqr* to the server account.**

To do this, select **RemotePut** under the Production Reporting menu. Then:

a. Enter **customer.sqr** into the local file and remote file input fields.

b. Select the radio button text transfer, and click **OK**.

6 **Execute the following command asynchronously (with no wait) on the server by selecting Remote Exec No Wait under the Production Reporting menu.**

```
mailsqr.sh. mailuser customer.sqr dblogin/dbpasswd
```

7 **Adjust the values of myserver, mylogin, mypasswd, mailuser, dblogin, and dbpasswd to fit your particular environment.**

To run an Production Reporting program other than *customer.sqr*, specify the filename of the program.

Following is the Visual Basic code that invokes the *mailsqr.sh*: script on the server:

```
Sqr0cx.RemoteConnect("myserver", "mylogin", "mypasswd"," ")
Sqr0cx.RemoteExecNoWait("mailsqr.sh mailuser customer.sqr
dblogin/dbpasswd", "output.txt")
Sqr0cx.RemoteDisconnect
```

Note how this code assumes that the Production Reporting program already exists on the server. This avoids the performance problem of copying the Production Reporting program file to the server upon each run, and it avoids multi-user conflicts.

3 Production Reporting Viewer ActiveX Control

In This Chapter

About the Production Reporting Viewer ActiveX Control

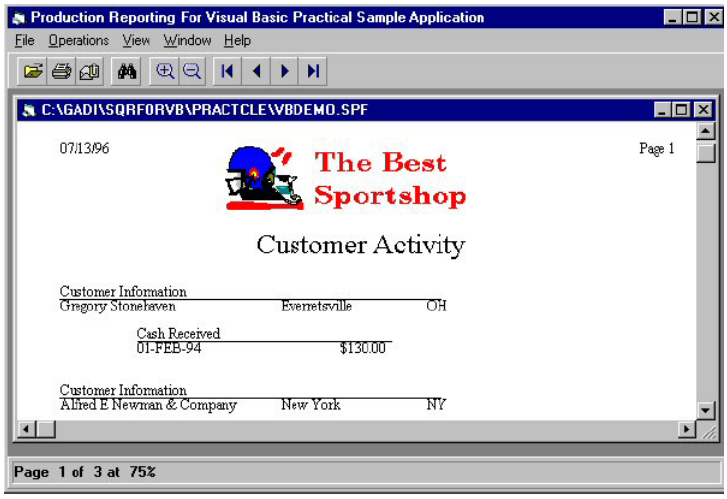
The Production Reporting Viewer ActiveX Control provides a way to display the Production Reporting program report output, the SPF (SQR Portable File) file. You can access the Production Reporting Viewer ActiveX Control functions through the built-in toolbar or through an application's user interface elements (such as menu items and/or buttons).

The Production Reporting Viewer ActiveX Control is responsible for what displays in the application child window. It handles all aspects of displaying the report. You simply place the Production Reporting Viewer ActiveX Control on a window (or form) and use the ActiveX Control methods to manipulate what report displays and how it displays.

A Sample Application

In the sample application shown in Figure 2, a report created with Production Reporting displays in a window that is part of a Visual Basic application. The application has a menu, a toolbar, and a caption. All the commands are delegated to the corresponding methods of the Production Reporting Viewer ActiveX Control.

Figure 2 Sample Application



The report displayed in the above Visual Basic application was created with the Oracle's Hyperion® SQR® Production Reporting Studio. TheProduction Reporting Viewer control displays the report from the SPF file.

Using the Production Reporting Viewer ActiveX Control

➤ To use the Production Reporting Viewer ActiveX Control:

1 **Instantiate an object for the Production Reporting Viewer ActiveX Control.**

You can do this in your development environment visually with application painters or forms, or by editing the application source code.

See the documentation for your development environment for information on how to instantiate ActiveX Controls. For instance, in Visual Basic, place the Production Reporting Viewer ActiveX Control onto a form generating an object in the form that you can reference, such as *frmMyForm.Production ReportingViewer*.

2 **Call the `SizeTo()` method to make the Production Reporting Viewer ActiveX Control occupy the whole window or change whenever the containing window (or form) changes size.**

In Visual Basic, for example, you would use the *Form.Size* subroutine.

3 **Call the `Open` method to open a SPF file; then, call other methods to manipulate how to display a report.**

4 **Call the `Close` method to close the report.**

Troubleshooting the Production Reporting Viewer ActiveX Control

Table 6 discusses some problems you may encounter when troubleshooting the Production Reporting Viewer Active X Control and suggested solutions to the problems.

Table 6 Situations and Solutions for Troubleshooting the Production Reporting Viewer ActiveX Control

Situation	Solution
An error message such as "File Not Found" displays when any Production Reporting Viewer methods are called.	Ensure that the necessary components are accessible. These components include SQRVC.OCX, BCLW32.DLL, LIBSTI32.DLL, and SQRVERR.DAT. They should all be in the same directory as SQRVC.OCX or somewhere in the path.

Production Reporting Viewer ActiveX Control Properties

Production Reporting Viewer ActiveX Control includes the following properties:

- CanCopy (Property)
- CanFind (Property)
- CurrentPage (Property)
- CurrentScale (Property)
- IsOpen (Property)
- PageCount (Property)
- ShowToolBar (Property)

CanCopy (Property)

Type	Access
Boolean	Read

Description

Determines if the `Copy` method is available to be used. For the `Copy` method to work, a report must currently be open and the user must have made some selection.

Type

Boolean

Access

Read

Description

Determines if the `Copy` method is available to be used. For the `Copy` method to work, a report must currently be open and the user must have made some selection.

CanFind (Property)

Type	Access
Boolean	Read

Description

Determines if the `Find` method is available to be used. For the `Find` method to work, a report must currently be open and the Find dialog box must *not* already be open.

CurrentPage (Property)

Type	Access
Long	Read/write

Description

Current page number.

CurrentScale (Property)

Type	Access
Short	Read/write

Description

The zoom scale. Valid values are:

0 = full size (scaled to fit the window)

25 = 25 percent

50 = 50 percent

75 = 75 percent

100 = 100 percent

150 = 150 percent

IsOpen (Property)

Type	Access
Boolean	Read

Description

Indicates if an SPF file is already open.

PageCount (Property)

Type	Access
Long	Read

Description

The total pages number of the open SPF file.

ShowToolBar (Property)

Type	Access
Boolean	Read/write

Description

If true, a built-in toolbar displays.

Production Reporting Viewer ActiveX Control Methods

Production Reporting Viewer ActiveX Control includes the following methods:

- Close (Method)
- Find (Method)
- Open (Method)
- PrintSpf (Method)
- Send (Method)
- SizeTo (Method)
- ZoomIn (Method)
- ZoomOut (Method)

Close (Method)

Return Value

(none)

Arguments

(none)

Description

Closes the currently open SPF report file.

Find (Method)

Return Value

(none)

Arguments

(none)

Description

Displays the Find Text dialog box. You can use this dialog box to search for text in the currently-open report output. Text that is part of a bitmap (such as text in a chart) is not part of the search. The dialog box remains open until you close it. Use the `CanFind` method to determine if `Find` can be used.

Open (Method)

Return Value

(none)

Arguments

Parameter	Data Type	Meaning
spfFile	String	The SPF filename to open.

Description

Opens and displays the specified SPF report file.

PrintSpf (Method)

Return Value

(none)

Arguments

(none)

Description

Displays the Print dialog box. You can use the Print dialog box to select what pages to print and to set up the printer. Press **OK** to print the currently-open report.

Send (Method)

Return Value

(none)

Arguments

(none)

Description

Causes the MAPI email dialog boxes to display. This allows the currently-open report to be transmitted via email as an attachment.

SizeTo (Method)

Return Value

(none)

Arguments

Parameter	Meaning
hWnd	Handle to the specified window.

Description

Sets the window size of the Production Reporting Viewer ActiveX Control to the entire displayable area of the specified window (also known as the client area). You typically use this method to make the control display the entire area of an MDI child window.

ZoomIn (Method)

Return Value

(none)

Arguments

(none)

Description

Sets the zoom scale setting of the currently-displayed report to the next higher setting. The range of zoom scale values is 25, 50, 75, 100, and 150. `ZoomIn` is not available when the `CurrentScale` is Full Page. This will stay on zoom scale 150 if the current zoom scale is 150.

ZoomOut (Method)

Return Value

(none)

Arguments

(none)

Description

Sets the zoom scale setting of the currently-displayed report to the next lower setting. The range of zoom scale values are 150, 100, 75, 50, and 25. `ZoomOut` is not available when the `CurrentScale` is Full Page. This will stay on zoom scale 25 if the current zoom scale is 25.

4 Production Reporting Print ActiveX Control

In This Chapter

About the Production Reporting Print ActiveX Control

The Production Reporting Print ActiveX Control prints an SPF file as the Production Reporting program report output.Production Reporting Print ActiveX Control invokes the Production Reporting Print program, SQRP.EXE, to do the printing; therefore Production Reporting must be installed on the PC.

Using the Production Reporting Print ActiveX Control

➤ To use the Production Reporting Print ActiveX Control:

1 **Ensure that you have installed Production Reporting.**

2 **Call the `Print (spfFile)` method.**

The default printer is set up on the PC.

The Production Reporting Print ActiveX Control locates SQRP.EXE by looking at the SQRDIR entry in the [Environment:Common] section of SQR.INI in the operating system's *system* directory.

Troubleshooting the Production Reporting Print ActiveX Control

Table 7 discusses some problems you may encounter when troubleshooting the Production Reporting Print ActiveX Control and suggested solutions to the problems.

Table 7 Situations and Solutions for Troubleshooting the Production Reporting Print ActiveX Control

Situation	Solution
Production Reporting Print ActiveX Control fails.	Either SQR.INI is not found in the system directory, or the directory pointed to by SQRDIR in the [Environment:Common] section in SQR.INI does not contain SQRP.EXE. Reinstall Production Reporting, and ensure that SQR.INI is in the system directory (such as C:\WINNT) and updated correctly.

Production Reporting Print ActiveX Control Methods

Production Reporting Print ActiveX Control includes the following method:

- Print (Method)

Print (Method)

Return Value

Type	Meaning
Boolean	*True* if succeeded; *False* if failed.

Arguments

Parameter	Data Type	Meaning
spfName	String	SPF filename

Description

Invokes SQRP.EXE in the latest installed Production Reporting package to print the SPF file.

5 Error Exceptions

In This Chapter

About Error Exceptions

The Production Reporting ActiveX Control and the Production Reporting Viewer ActiveX Control throw an error exception when an error occurs. The tables in this chapter list each error's status code, a text message describing the nature of the error, and the action to take to correct the error.

Production Reporting ActiveX Control Status Codes

Table 8 Production Reporting ActiveX Control Status Codes

Value	Internal Symbol	Description	Action
1000	LOCAL_ERR_EXECUTABLE	Failed to load SQR.DLL or launch SQR.EXE.	Ensure that Production Reporting is installed correctly, that SQR.INI is in the system directory, and that SQRDIR is pointing to where SQR.DLL resides.
1001	LOCAL_ERR_GENERAL	Failed to run Production Reporting.	Check that all components required by Production Reporting are available.
1002	LOCAL_ERRDAT_FILE	Failed to process the file SQRERR.DAT.	Check that SQRERR.DAT is present and accessible by SQR.EXE. This error can also indicate that there are not enough system resources, such as file handles, available.
1003	LOCAL_COMMAND_LINE	Invalid command line option specified.	Correct the command line option passed to the `LocalRun` method.
1004	LOCAL_ERR_CREATE_SQT	Failed to create the SQT file.	Check that there is enough disk space available. This error can also indicate that there are not enough system resources, such as file handles, available.

Value	Internal Symbol	Description	Action
1005	LOCAL_ERROR_COMPILE	Failed to compile Production Reporting.	Modify Production Reporting source code as necessary.
1006	LOCAL_ERR_READ_SQT	Failed to open or read the SQR or SQT file specified in the command line.	Check that the specified SQR or SQT file is present. This error can also indicate that there are not enough system resources, such as file handles, available.
1007	LOCAL_ERR_WRITE_LIS	Failed to create or write the LIS file.	Check that there is enough disk space available. This error can also indicate that there are not enough system resources, such as file handles, available.
1008	LOCAL_ERR_WRITE_ERR	Failed to create or write the ERR file.	Check that there is enough disk space available. This error can also indicate that there are not enough system resources, such as file handles, available.
1009	LOCAL_ERR_WRITE_LOG	Failed to create or write the LOG file.	Check that there is enough disk space available. This error can also indicate that there are not enough system resources, such as file handles, available.
1010	LOCAL_ERR_READ_POSTSCRI	Failed to open or read the file POSTSCRI.STR.	Check that POSTSCRI.STR is present and accessible by SQR.EXE. This error can also indicate that there are not enough system resources, such as file handles, available.
1011	LOCAL_ERR_RECURSIVE	Production Reporting cannot be called recursively.	Production Reporting attempted to run another Production Reporting program. Modify Production Reporting logic as necessary.
1012	LOCAL_ERR_WINDOWS	Failed to allocate memory or a system resource.	Check that there are enough memory or system resources available.
1013	LOCAL_ERR_INTERNAL	Production Reporting caused an internal error.	This typically indicates that Production Reporting attempted an invalid operation, using too large of an array for example. Modify Production Reporting logic as necessary.
12000	REMOTE_ERR_NOT_CONNECTED	Attempted an Production Reporting Remote method when no connection is open.	Add program logic to prevent calling the method when no connection is open. Use `RemoteIs Connected` to determine if a connection is open.
12102	REMOTE_ERR_BUSY	An Production Reporting Remote operation is already in progress.	Add program logic to prevent calling an Production Reporting Remote method until the last method completes. This typically occurs because of the way Production Reporting Remote methods operate in the background. When an Production Reporting Remote method that operates in the background is called, it releases control back to the windows message queue until the background method completes.

Value	Internal Symbol	Description	Action
12103	REMOTE_ERR_ABORTED	An Production Reporting Remote operation was aborted by the `RemoteCancel` method.	No action necessary.
12104	REMOTE_ERR_FAILED	An Production Reporting Remote operation failed.	Check that there are enough memory or system resources available.
12105	REMOTE_ERR_NO_TIMERS	Failed to allocate a Windows timer.	Check that enough Windows timers are available. A finite number of Windows timers are available on the system. Other applications may be using Windows timers. Close those applications to make the timers available.
12106	REMOTE_ERR_TIMER_LOOKUP	Failed to locate the Windows timer number internally.	This should never occur. If it does occur, contact customer support.
12107	REMOTE_ERR_MEMORY	Failed to allocate memory.	Check that there are enough memory or system resources available.
12108	REMOTE_ERR_FILE_OPEN	Failed to open the specified file.	Check that the specified file is present and accessible by SQRREM.DLL. This error can also indicate that there are not enough system resources, such as file handles, available.
12109	REMOTE_ERR_TIMEOUT	The Production Reporting Remote operation timed out.	This occurs when the Production Reporting Remote operation takes more time than specified by the time-out value. Check that the time-out value, specified by the "TimeOut" entry in the [SQR Remote] section of SQR.INI, is set to an appropriate value. The timeout is set to 30 seconds by default. (See"[SQR Remote] Section" in Volume 2 of the *Production Reporting Developer's Guide* for more information.) For example, copying a very large file over a slow network connection may require a large time-out value. This error can also indicate that the specified hostname or IP address passed to `RemoteConnect` is invalid.
12110	REMOTE_ERR_INV_ARG	Invalid argument passed to the Production Reporting Remote method.	Modify the program source code as necessary.
12111	REMOTE_ERR_DELETE	Failed to delete remote file using FTP delete file capability.	Check that the remote file has write permissions. Use an FTP utility, one is typically supplied with your TCP/IP product, to attempt to delete the specified file.
12112	REMOTE_ERR_TYPE	Failed to set the file type to ASCII or binary using the FTP type capability.	This should never occur. If it does occur contact customer support.

Value	Internal Symbol	Description	Action
12115	REMOTE_ERR_SQR	Production Reporting ran, but it generated errors.	This indicates that Production Reporting caused an error. This is typically attributed to failure to connect to the database, failure to compile the Production Reporting program source code, or failure in the Production Reporting program logic. See the contents of the ERR file for details of the error. If it is needed, modify Production Reporting source code as necessary.
12116	REMOTE_ERR_GEN_UNIQUE	Failed to generate a unique file name.	The `RemoteRun` method attempts to generate a unique filename by generating a random filename starting with the characters "sq", "sq029342.sqr" for example. It attempts this many times before aborting with this error. Retry the operation. If this occurs many times, the remote directory may be filled with garbage files with the "sq*.sqr" filename. Delete the garbage files from the remote directory.
12117	REMOTE_ERR_CONNECTED	Attempted to reconnect, but the current connection can not be disconnected.	Check your program logic to ensure that the connection can be disconnected before trying to make another connection.
12118	REMOTE_ERR_RUNNING	Production Reporting is still running at the time `RemoteGetResults` is called.	If Production Reporting processes a large amount of data, it may take a long time to run the report. Use `RemoteGetResults` at a later time to detect when Production Reporting has completed. If it takes an unreasonable amount of time to run Production Reporting , it may indicate that Production Reporting has hung waiting for some resource that is not available. In this situation, modify Production Reporting source code as necessary.
12119	REMOTE_ERR_SQR_CMD	Failed to invoke Production Reporting to run Production Reporting Remote Production Reporting Remote uses the REXEC capability to execute an Production Reporting program on the UNIX server. It does this by using REXEC to run the remote script. The remote script runs the startup script for the account to set the necessary environment variables; then, the remote script runs the Production Reporting executable. The Production Reporting	See the contents of the OUT file for details of the error. This typically indicates that the UNIX account is not configured properly. To check the account: ● Check that the appropriate remote script is present in the home directory of the UNIX account. The filename of the remote script is "rsqr.csh" for C Shell based accounts and "rsqr.sh" for Bourne shell and Korn shell based accounts.

Value	Internal Symbol	Description	Action
		executable runs the Production Reporting program.	● Check that the remote script has execute permissions. ● Check that the startup script for the UNIX account, ".login" for C Shell based accounts and ".profile" for Bourne shell and Korn shell based accounts, does not perform an invalid operation that causes the shell to terminate. In general, the "$TERM" variable is not set so the script should not utilize it and the standard input is redirected to "/dev/null" so the script should attempt to directly read from it. ● Check that the startup script sets the appropriate environment. In general, it should set the SQRDIR variable, the database environment variables, and the SQRDIR into the UNIX PATH. ● Check that theProduction Reporting executable is present on the UNIX server and can be accessed using the contents of the UNIX PATH.
12120	REMOTE_ERR_FTP_LOGIN	Failed to log into the server using FTP.	Check that the specified hostname, username, and password are correct. Also check that the account has FTP permission. Use an FTP utility, one is typically supplied with your TCP/IP stack, to attempt to log into the UNIX account.
12121	REMOTE_ERR_REXEC_LOGIN	Failed to log into the server using REXEC.	Check that the specified hostname, username, and password are correct. Also check that the account has REXEC permission. Use an REXEC utility, one is typically supplied with your TCP/IP product, to attempt to execute a command such as "ls -l" utilizing the specified UNIX account.
12122	REMOTE_LOCFILE_PERM	Invalid permissions on local file.	Check that the specified local file is not set with read only permission. Use the "attrib" utility to change the permission.
12123	REMOTE_ERR_FTP_GET	Failed to get the file from the server using FTP.	Check the FTP permissions to the remote file. Use an FTP utility, one is typically supplied with your TCP/IP product, to attempt to get the file from the UNIX server.
12124	REMOTE_ERR_FTP_PUT	Failed to put the file to the server using FTP.	Check the FTP permissions to the remote file. Use an FTP utility, one is typically supplied with your TCP/IP product, to attempt to put the file to the UNIX server.

Value	Internal Symbol	Description	Action
12125	REMOTE_ERR_FTP_NAMELIST	Failed to get a directory listing from the server using FTP.	Check the FTP permissions to the remote directory. Use an FTP utility, one is typically supplied with your TCP/IP product, to perform an "ls" operation on remote directory of the UNIX server.
12126	REMOTE_ERR_FTP_PWD	The FTP print working directory operation failed.	Check that the user account has read permissions to the working directory on the remote server.
12127	REMOTE_ERR_UNKNOWN_SERVER	Attempted to open a connection to an unknown server type.	When a connection is opened to a remote server,Production Reporting Remote performs a print working directory operation. It checks for the returned string to determine the server type. An ":\" in the string indicates a Windows server, and a "/" indicates a UNIX server. You must use a Windows or UNIX based server.
12129	REMOTE_ERR_REMOTE_ NOT_FOUND	Attempted to retrieve or otherwise read a remote file that does not exist.	Check that the specified filename is correct.
12130	REMOTE_ERR_REXEC	Failed to execute a command using REXEC.	Make sure you have permission on the server to execute commands remotely using REXEC.

Production Reporting Viewer ActiveX Control Status Codes

Table 9 Production Reporting Viewer ActiveX Control Status Codes

Value	Internal Symbol	Description	Action
53	CTL_E_FILENOTFOUND	Failed to open or read the SPF file.	Check that the SPF file is present and accessible. This error can also indicate that there are not enough system resources, such as file handles, available.
321	CTL_E_INVALIDFILEFORMAT	The file specified to open is not in correct SPF format.	Check that the program logic is passing the correct filename to the `Open` method.
1000	SQRV_ERR_FILENOTOPEN	The specified operation did not take effect because no SPF file is open.	Call the `Open` method to open an SPF file and then do the operations.

6 The Visual Basic Sample Program

In This Chapter

About the Visual Basic Sample Program

The Visual Basic Sample Program is built with Microsoft Visual Basic Version 4.0, demonstrating the usage of all three Production Reporting ctiveX controls.

The program is a Graphical User Interface (GUI) that uses all the properties and methods of these controls. The sample directory of the installation contains an executable file *tryvbocx.exe* (for which all the Visual Basic supporting files have been installed onto the system), as well as its source code. If you have Visual Basic, you can load the project, view its source code, and do further experiments. If you do not have Visual Basic, you can still view the source code (in *.frm files). Do a search on "Sub", and see howSQR Production Reporting Activator objects and methods are used.

The program has four top-level menu items. The first three are for Production Reporting ActiveX Controls, and the last one is Help. The Help menu item displays an About box when the program starts, explaining the basic functions and providing some notes. The other three menus items reflect the methods of the ActiveX Controls, providing minimal extra functions other than displaying results (on the prompt at the bottom of the window), error handling (using Visual Basic's ON ERROR statement and Err object, which is the error object thrown from inside the ActiveX Control's), and other GUI elements to view or set properties. The following sections briefly describe each group.

Production Reporting ActiveX Control Function Group

The main form contains an Production Reporting ActiveX Control object, SQRX. The "Production Reporting" menu item contains options for both local and remote functions of the Production Reporting ActiveX Control, which are routed to the embedded object, SQRX.

`Local Run` and `Local Run No Wait` both prompt for an SQR or SQT filename, then prompt for the complete Production Reporting ommand line, with the filename as the first parameter. If you click **OK**, `SQRX.LocalRun()` or `SQRX.LocalRunNoWait()` are called with this

command line. If successful, `Local Run` and `Remote Run`, automatically display the report output in the Viewer window.

The Production Reporting ActiveX Control has a *DatabaseName* property. Since Production Reporting has versions that run against all the major relational and ODBC compliant database systems, it is possible that you have Production Reporting for Oracle and Production Reporting for ODBC installed on the same machine.Production Reporting distinguishes between them by looking for an environment variable called SQRDIR in the SQR.INI file normally installed in the system directory such as C:\WIN. The SQR.INI file keeps a separate SQRDIR for each supported database in a section like [Environment:Oracle] or [Environment:ODBC].

To invoke the Production Reporting ActiveX Control's `LocalRun` or `LocalRunNoWait` to run an Production Reporting program that is written for Oracle, ensure that Production Reporting for Oracle is called. This is can be done by setting the DatabaseName property to *Oracle*. In the sample program, "Set Local DatabaseName" allows you to select from one of the supported databases. Note that this is a Visual Basic form (dialog box), not one from the ActiveX Control. The "(common)" entry leaves DatabaseName blank, causing Production Reporting ActiveX Control to use [Environment:Common] section in SQR.INI, which is the same as the latest installed Production Reporting product.

To run Production Reporting Remote, you need to establish a connection to a server by selecting **Remote Connect** from the menu. You can connect to another server afterward, which automatically disconnects you from the previous server. When the program closes, it disconnects by itself, but you can choose **Remote Disconnect** if desired.

Once connected, you can run Production Reporting remotely in synchronous mode or in asynchronous mode, using `Remote Run` and `Remote Run No Wait`, respectively. These two options are optimized for running Production Reporting. As a result `Remote Run` allows you to run SQR or SQT programs on remote servers as easily as on local machine.

For `RemoteRunNoWait`, the control is returned immediately with a temporary filename on the server, which is used in `RemoteGetResult` to get the result files of the remote run, or `RemoteCleanup` to delete its related files on the server. The application (in this case, this sample program) should check (repeatedly) if the remote run has finished by calling `RemoteGetResult`. If the result is FALSE, then it is not yet finished. Once finished, `RemoteGetResult` retrieves all the output files to the local machine, just like the output from a local call or a synchronous remote run.

In addition to these Production Reporting -specific methods,Production Reporting ActiveX Control also exposes a number of more generic FTP/REXEC operations, grouped under the "More Remote Methods" option, such as `RemoteGet` (FTP get), `RemotePut` (FTP put), `RemoteExec` (REXEC), `RemoteExecNoWait`, and so on. Some of these operations, such as `RemoteGetUniqueSQRFile` and `RemoteCleanup`, support the remote running of Production Reporting.

Production Reporting Viewer ActiveX Control Function Group

Like the Production Reporting menu option, the Viewer menu option displays all the methods in the SQR Production Reporting ActivatorActiveX Control. To use the Viewer menu, open an

SPF file. (A few sample SPF files are included in the \sample directory of the Oracle's Hyperion® SQR® Production Reporting Activator installation.) All the menu items under Viewer call the Oracle's Hyperion® SQR® Production Reporting Viewer ActiveX Control's methods or read/set properties, except for "Page Navigation", which uses the properties `CurrentPage` and `PageCount` to do things like first page, last page, next page, previous page and go to page. Notice that it does not make sense to call the `SizeTo` method other than from its containing (parent) window. As a result, it is called by the message handler for the frame window when processing WM_SIZE message.

You can have multiple reports open in an application, each in its own Viewer ActiveX control. The last option on the Viewer menu demonstrates this feature. It has four fix size controls on a window (you probably need to enlarge the window to see all), and the program helps you bring up four reports.

Production Reporting Print ActiveX Control Function Group

An instance of the Control is embedded in the main form. When its only method (under the Print menu) is called, the command line passes to that ActiveX Control. The only thing you need to do is provide the SPF filename.

Index

R

S

T

V

Z