24x7 Scheduler™ 3.4

24x7 Remote Automation Server™ for UNIX and Linux Reference

Table of contents

About This Manual

This user manual describes all features of the 24x7 Remote Control COM API for 24x7 Scheduler™ v3.4.16 running on Microsoft Windows 95/98/Me/NT/2000/XP/2003 operating systems. This manual contains information for developers and experienced users of the 24x7 Scheduler who want to create custom interfaces to the 24x7 Scheduler system and/or tightly integrate it with other applications and systems.

The complete 24x7 Scheduler documentation consists of several manuals and also an on-line interactive help system. The on-line help is available at any time when you are running 24x7 Scheduler. Depending on what you are doing, you can press F1, select Help from the Menu Bar, or select the Help button on a dialog.

Conventions Used in This Document

This section describes the style conventions used in this document.

Italic
An *italic* font is used for filenames, URLs, emphasized text, and the first usage of technical terms.

`Monospace`
A `monospaced` font is used for code fragments and data elements.

Bold
A **bold** font is used for important messages, names of options, names of controls and menu items, and keys.

Graphical marks
- This mark is used to indicate product specific options and issues and to mark useful tips.
- This mark is used to indicate important notes.

Abbreviations and Terms

This guide uses common abbreviations for many widely used technical terms including COM, ASP, API, and other.

Trademarks

24x7 Automation Suite, 24x7 Scheduler, DB Audit, DB Audit Expert, DB Mail, DB Tools for Oracle are trademarks of SoftTree Technologies, Inc.

Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP are registered trademarks of Microsoft Corporation. UNIX is registered trademark of the X/Open Consortium. Sun, SunOS, Solaris, SPARC are trademarks or registered trademarks of Sun Microsystems, Inc. Ultrix, Digital UNIX and DEC are trademarks of Digital Equipment Corporation. HP-UX is a trademark of Hewlett-Packard Co. IRIX is a trademark of Silicon Graphics, Inc. AIX is a trademark of International Business Machines, Inc. AT&T is a trademark of American Telephone and Telegraph, Inc.

Microsoft SQL Server is a registered trademark of Microsoft Corporation.

Overview

The 24x7 Remote Automation Server™ (24x7 RA Server) for UNIX provides methods for building robust cross-platform Linux, UNIX and Windows automation tasks.

The 24x7 Remote Automation Client (24x7 RA Client) is used to communicate to the 24x7 RA Server. The 24x7 RA Client is tightly integrated with the 24x7 Scheduler running on Windows servers and workstations. Usage of the 24x7 RA Client is totally transparent and hidden from the end user. The Client and Server parts communicate using standard TCP/IP protocol. Together they provide transparent cross-system file access and process management methods that allow automating a wide variety of tasks to be performed on different Linux, UNIX and Windows systems as if they were just one big system. 24x7 RA Server supports multiple simultaneous connections with independent user sessions and is capable to process multiple concurrent tasks.

A quick automation example

Perhaps, you have a need to load some data file residing on your Windows workstation to a database server running on your Linux server computer. The following script demonstrates how easily files can be transferred and then loaded into the database. Note that this method does not require any other software such as expensive database client software installed on your Windows system. It also requires minimal network resources as the file is only transferred once and the actual data loading is performed on the database server side. This entire processing is automatically monitored by 24x7 Scheduler built-in advanced error-checking and process tracing functions. The job owner can be automatically notified in case of the job failure or success.

```
// connect to the Linux server
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
// upload file
RAFileTransfer "ToRemote", "c:\\data\\myfile.data", "/home/linuxuser/myfile"
// run database import utility
Dim pid, number
Dim import_output, string
RARunAndWait "/database/bin/import myfile.data", 0, pid, output
// disconnect from the server
RADisconnect

// Notify job owner about job completion and attach the output of the import utility.
// This output should contain the import processing statistics such as processing
// time and number of loaded records
MailSend "Exchange Settings", "mypassword", "myname@mycompany", &
               "@D"mm/dd/yy h:mm:ss" myfile.data file was successfully loaded", &
               import_output
```

24x7 Remote Automation Server (24x7 RA Server)

Supported Linux and UNIX systems

24x7 RA Servers are available in two flavors:

1. **Native code version** (C/C++ compiled and source code programs)
2. **Universal Java version** (compiled Java programs)

The 24x7 RA Server (Native code version) can be installed and run on a UNIX server or workstation. The following platforms are supported at this time:

- Linux ® 32-bit version RedHat compatible distributions;
  Linux ® 32-bit version Debian compatible distributions
- Sun Solaris ® 32-bit (Intel i386) version 5.6 and later
- Sun Solaris ® 32-bit (Sparc) version 5.6 and later;
  Sun Solaris ® 64-bit (Sparc) version 7 and later
- Digital UNIX ® 64-bit version 4.0B and later, including Tru64™ UNIX ®
- FreeBSD ® (Intel i386) version 3.1 and later
- IBM AIX ® 32-bit version V3R3.2 and later
  IBM AIX ® 64-bit version V4 and later
- HP-UX® version 10.0 and later

The 24x7 RA Server (Universal Java version) can be installed and run on any UNIX server or workstation supporting 32-bit or 64-bit Java JDK 1.4.2 or later.

Installation and Uninstallation

24x7 RA Servers – Native Code version

Installation

The 24x7 RA Server consists of single executable file which supports self-installing options. The 24x7 RA Server versions are available for Linux and a number of UNIX platforms.
Before you can run 24x7 RA Server, you must install it. The installation process is very simple:

1. Create a directory where you want to install the server, for example **/24x7/ra_server**
2. Copy **ra_server** file to the created directory. You can use the standard UNIX **cp** command to copy the file. Set current directory to the directory where you copied 24x7 RA Server files. To set the current directory use the standard **cd** command.

3. Run **./ra_server install** command to install the server with default options. You can use optional command line parameters to customize 24x7 RA Server installation and settings. For supported command line parameters please run **./ra_server –h**. For example if you need to change the default port number, on which the server will be listening for remote clients requests (default is 1096) and the default log path (default is */var/log/ra_server.log*) you can run a command like the following
./ra_server install –f /24x7/ra_server/ra_server.log –p 1097

Note: You must logon as a root or other user with an administrative account before installing 24x7 RA Server

Uninstallation

1. Set current directory to the directory where you copied 24x7 RA Server files. In this directory run **./ra_server uninstall** command. This is it.

24x7 RA Servers – Universal Java version

Before you can run 24x7 RA Server, you must install it. The installation process is straight and consists of 3 procedures:

1. Java installation

1. If you don't yet have Java installed in your system, install Java 2 Platform, Standard Edition v 1.4.x or better.

2. Make sure $JAVA_HOME is correct for all Unix accounts.

2. 24x7 Remote Access Server Installation

1. Create a directory where you want to install the server, for example **/24x7/ra_server**

2. Copy **jra_server.tgz** file to the created directory. You can use the standard UNIX **cp** command to copy the file. Set current directory to the directory where you copied 24x7 RA Server files. To set the current directory use the standard **cd** command.

3. Unzip the installation package
gtar xvfz jra_server.tgz

4. Set permissions for shell scripts in **/24x7/ra_server** in order to make them executable
chmod 755 *.sh *.pl

5. Allow users to create log files
chmod 777 .

6. *(optional step)* Configure Unix **syslogd** daemon to start with -r in order to support 24x7 RA Server syslog options.

7. Copy Unix PAM security module configuration file to the right place for your operation system (for example, for Linux it is /etc/pam.d)
cp ra /etc/pam.d/

8. Start 24x7 RA Server**./server.sh** with default options. You can use optional command line parameters to customize 24x7 RA Server settings. For example if you need to

change the default port number, on which the server will be listening for remote clients requests (default is 1096) and the default log path (default is */var/log/ra_server.log*) you can run a command like the following
./server.sh–f /24x7/ra_server/ra_server.log –p 1097

To allow syslog logging add –s switch to command line
./server.sh–f /24x7/ra_server/ra_server.log –p 1097 -s

Note: You must logon as a root before installing 24x7 RA Server

3. Perl-Authen-PAM installation

(*) Perl-Authen-PAM packages are available for most systems. PAM packages are free and can be download using the following links:

RPM for Linux: http://dag.wieers.com/packages/perl-Authen-PAM
DEB for Linux: http://packages.debian.org/stable/interpreters/libauthen-pam-perl

For Solaris, FreeBSD, OpenPAM, HP-UX, Darwin, Mac OS 10 do the following:

1. Download the Perl Authen::PAM module (Authen-PAM-0.15.tar.gz or later). It is available at http://www.perl.com/CPAN/authors/id/N/NI/NIKIP/ . Save the downloaded file in /24x7/ra_server directory.

2. Install Perl Authen::PAM module

```
cd /24x7/ra_server
tar -zxvf Authen-PAM-0.15.tar.gz
cd Authen-PAM-0.15
perl Makefile.PL
make
make install
```

Uninstallation

1. Simply delete all files from the directory where you previously installed 24x7 RA Server – Universal Java version. This is it.

Security

All versions of 24x7 RA Server, except Digital UNIX versions, support strong authentication capabilities. For the user authentication 24x7 RA Server utilizes Pluggable Authentication Modules (PAM) that is standard part of all modern Linux and UNIX distributions. In other words, 24x7 RA Server is a PAM-aware application and so it is possible to switch between the authentication mechanism(s) it uses without rewriting and recompiling 24x7 RA Server. For more information about PAM, have a look at the Linux-PAM Guide for System Administrators.

In addition to user authentication capabilities 24x7 RA Server also implements Linux/UNIX native TCP daemon, which gets its instructions on whether to allow or deny access from the two files:

/etc/hosts.allow and /etc/hosts.deny.

It first scans /etc/hosts.allow for rules that match the particular service and computer host name, and then searches /etc/hosts.deny. If no match is found, access is allowed. By default, the files /etc/hosts.allow and /etc/hosts.deny are empty, allowing access to all available TCP services on your Linux machine to everyone. When editing hosts.allow and hosts.deny files consider the following two general recommendations:

1. Use IP addresses instead of system or domain names.
2. Set up /etc/hosts.deny to deny everything (ALL), then permit only specific sites with /etc/hosts.allow.

For more information about /etc/hosts.allow and /etc/hosts.deny files run **man host_access(5)** command from the Linux/UNIX command prompt.

Sharing

24x7 RA Server is a true client-server application that can simultaneously provide services to multiple 24x7 RA Clients.

24x7 Remote Automation Client (24x7 RA Client)

Installation

24x7 RA Client is installed as an optional component provided with the 24x7 Automation Suite

To install 24x7 RA Client, simply check this option in the graphical installer when installing 24x7 Automation Suite.

Uninstallation

The 24x7 RA Client is automatically uninstalled with the 24x7 Automation Suite.

Automating Linux and UNIX Jobs

To automate your Linux/UNIX processing schedule JAL script type jobs using 24x7 Scheduler. In these jobs call remote automation methods described in the following topics. You can write such script yourself using available RA Client methods (see the following topic) or use available appropriate job template provided for Linux/UNIX automation jobs. For information and template usage instructions refer to *24x7 Scheduler User's Guide and JAR Reference*.

24x7 RA Client Methods

Host Connectivity

RAConnect

Description: Establishes new connection to 24x7 Remote Automation Server running on Linux or UNIX server or workstation, creates new session and performs authentication

Syntax: RAConnect host, port, user_id, password

Argument	Description
server_name	A string whose value is the network name or TCP/IP address of the remote Linux/UNIX computer running 24x7 RA Server.
port	A number whose value is the port number on which the Remote Automation Server is listening for client connections. The default value is 1096. You can change this value when configuring 24x7 RA Server.
user_id	A string whose value is the name of the user to log on to the remote Linux/UNIX computer
password	A string whose value is the password to use to log on to the remote Linux/UNIX computer

Return value: None.

Usage: Use RAConnect statement to establish a new RA session before executing other RA methods.

Example:
```
Dim process_id, number
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
RARun "/home/linuxuser/program", process_id
RADisconnect
```

RADisconnect

Description: Closes RA session previously established using RAConnect statement

Syntax: RADisconnect

Arguments: None.

Return value: None.

Usage: Use RAConnect statement to establish a new RA session. After you are done with all remote processing close session and release allocated resource by calling RADisconnect statement.

Process Operations

RARun

Description: Runs the specified program or command on the remote Linux/UNIX computer.

Syntax: RARun command, return

Argument	Description
command	A string whose value is the full or partial path and filename of the module to execute. The module may be a shell command or other executable file. If a partial name is specified, the current directory is used by default to search for the file. The module name must be the first white space-delimited token. Other tokens are passed as command line parameters. You should always include the full path - don't rely on the environment variables, because they may be different at the time the command runs, depending on what account the program is being run under. Also, keep in minds that other programs can modify the environment variables as well. You can use macro-parammeters inside program name and command line parameters string to pass dynamic information (such as the current month) to the scheduled program.
return	A numeric variable that receives the ID of the created process.

Return value: Number. Returns the operation system unique of the created process. You can use that number to control and manipulate further process execution.

Usage: You can use RARun for any application that you can run from the operating system. If you specify command line parameters, the application determines the meaning of those parameters. A typical use for command line parameters is to identify a data file to be opened when the program is executed.

Example:

```
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
// move remote files
Dim process_id, number
RARun "mv /trade/ftp/*.dat  /trade/backup", process_id
// close connection
RADisconnect
```

RARunAndWait

Description: Runs the specified program or command on the remote Linux/UNIX computer and enters an efficient wait state until this process finishes or the timeout interval elapses. In the latter case, the 24x7 RA Server forcedly terminates the process.

Syntax: RARunAndWait command, timeout, process_id, output

Argument	**Description**
command	A string whose value is the full or partial path and filename of the module to execute. The module may be a shell command or other executable file. If a partial name is specified, the current directory is used by default to search for the file. The module name must be the first white space-delimited token. Other tokens are passed as command line parameters. You should always include the full path - don't rely on the environment variables, because they may be different at the time the command runs, depending on what account the program is being run under. Also, keep in minds that other programs can modify the environment variables as well. You can use macro-parammeters inside program name and command line parameters string to pass dynamic information (such as the current month) to the scheduled program.
timeout	A number whose value is the maximum time interval within which you allow the specified process to run. Use 0 timeout to allow infinite waiting.
process_id	A numeric variable that receives the ID of the created process.
output	A string variable that receives the data written by the created process to the standard error and standard output.

Return value: Number and String. Returns the operation system unique identifier of the created process. You can use that number to control and manipulate further process execution. Also returns output of the process.

Usage: You can use RARunAndWait for any application that you can run from the operating system. If you specify command line parameters, the application determines the meaning of those parameters. A typical use for command line parameters is to identify a data file to be opened when the program is executed.

You can use the output parameter to get the process output. For example, executing RARunAndWait "ls", 0, pid, output will return the list of files on the remote computer. The result will be placed into output variable.

Example:

```
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
// move remote files
Dim process_id, number
Dim output, string
RARunAndWait "mv /trade/ftp/*.dat  /trade/backup", 0, process_id, output
// now, files from the remote directory are backed, let's copy new file
RARunAndWait "do_ftp", 0, process_id, output
// and finally start the load process
RARun "do_load", process_id
```

```
// close connection
RADisconnect
```

RAProcessExitCode

Description: Returns exit code of the last process executed using RARunAndWait command.

Note: This command is only supported with 24x7 RA Servers Universal Java versions!

Syntax: RAProcessExitCode exit_code

Argument	Description
Exit_code	A number whose value is exit code of the last executed command.

Return value: Number.

Usage: Use RAProcessExitCode to obtain exit code of the last executed process. You can then evaluate the exit code for success or failure. Success is traditionally represented with exit code **0**; failure is normally indicated with a non-zero exit-code. This value can indicate different reasons for failure and varies for different programs and different operation system. For example, GNU *grep* returns 0 on success, 1 if no matches were found, and 2 for other errors (syntax errors, non-existent input files, etc). Refer to your system documentation for description of supported exit codes.

Example:

```
// connect to remote computer
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
// run some remote process
Dim process_id, number
Dim output, string
RARunAndWait "grep error file.log", 0, process_id, output
// grab process exit code
Dim exit_code, number
RAProcessExitCode exit_code
// close connection
RADisconnect

// analyze exit code
IfThen exit_code, HANDLE_ERROR
Exit

HANDLE_ERROR:
MessageBox "Unable to run GREP"
```

RAProcessKill

Description: Terminates the specified process on the remote Linux/UNIX computer.

Syntax: RAProcessKill process_id

Argument	**Description**
process_id	A number whose value is the id of the process which you want to terminate

Return value: None.

Usage: The RAProcessKill statement can be used to unconditionally cause a remote process to exit. Use it only in extreme circumstances.

Normally, for the process_id you use the value previously returned by a call to RARun statement.

You can also use RAProcessList statement to get the list of the currently running processes and then from that list find out the ID of the desired process.

Note:
Termination of a process does not always forces termination of all its child processes!

Example:

```
// connect to remote computer
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
// run some remote process
Dim process_id, number
RARun "/home/linuxuser/some_program", process_id
// wait 10 seconds
Wait 10
// stop the process
RAProcessKill process_id
// close connection
RADisconnect
```

RAProcessList

Description: Reports running processes, their process Ids, and other process parameters.

Syntax: ProcessList style, return

Argument	Description
style	A boolean whose value is indicates the output format and scope Use false value to get the flat list of top-level processes. Use true value to get the tree-style list of all processes. Child processes are indented by the appropriate number of spaces.
return	A string variable that receives the returned value

Return value: String. Returns Ids and names of all running process as a multi-line string. The format of the output may vary on different Linux/UNIX distributions.

Usage: Use the returned value to find out process state or ID for the desired application. The application must be running at the time when you call RAProcessList statement.

Note: Because of the security restrictions in Linux/UNIX the returned string may include incomplete list of running processes.

Example:

```
// connect to remote computer
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
// run some remote process
Dim process_id, number
RARun "/home/linuxuser/some_program", process_id
// get the process list
Dim processes, string
RAProcessList false, processes
MessageBox processes
// close connection
RADisconnect
```

Directory Operations

RADir

Description: Returns comma-separated list of files in the current working directory on the connected UNIX/Linux computer.

Syntax: RADir mask, list

Argument	Description
file_mask	A string whose value is the file mask to use for searching remote files
list	A string variable that receives the returned value

Return value: Comma-separated list of files on the remote 24x7 RA Server.

Usage: To get the list of all files use "." for the file_mask. To get the list of all files with recursive traverse in subdirectories use "*" for the file_mask.

Example:
```
      Dim file_list, string
      RADir "*.log", file_list
```

RAGetWorkDir

Description: Reports name of the current working directory on the remote 24x7 Remote Automation Server running on Linux/UNIX server or workstation. Each RA user session runs in its own environment and has session logical working directory.

Syntax: RAGetWorkDir dir

Argument	Description
dir	A string variable that receives the returned value

Return value: Full path of the working directory.

Usage: RAGetWorkDir statement is equivalent to Linux/UNIX shell pwd command executed on the remote host.

Example:

```
Dim work_dir, string
RAGetWorkDir work_dir
Concat "Current working directory on the remote host is ", work_dir,
work_dir
MessageBox work_dir
```

RASetWorkDir

Description: Changes current working directory on the remote 24x7 Remote Automation Server running on Linux/UNIX server or workstation. Each RA user session runs in its own environment and has session logical working directory.

Syntax: RASetWorkDir dir

Argument	Description
dir	A string whose value is the name of the remote directory

Return value: None.

Usage: RASetWorkDir statement is equivalent to Linux/UNIX shell cd command executed on the remote host.

Example:

```
        RASetWorkDir "/project/ftp/incoming"
```

High-Level File Operations

RAFileDateTime

Description: Reports last modification date and time of a remote file.

Syntax: RAFileDateTime file, return

Argument	Description
file	A string whose value is the name of a remote file
return	A datetime variable that receives the returned value

Return value: Datetime. Returns the date and time that a file was last modified..

Example:
```
// connect to host and get file modification time
Dim file_time, datetime
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
RAFileDateTime "/home/linuxuser/myfile", file_time
RADisconnect
// display file time in a message box
MessageBox file_time
```

RAFileSave

Description: Saves local data in the specified file on the remote Linux/UNIX computer.

Syntax: RAFileSave file, data

Argument	Description
file	A string variable whose value is the name of the remote file into which you want to save the text
text	A string whose value is the data want to save in the file

Return value: None.

Usage: If the file already exists then RAFileSave overwrites it.

Example:

```
Dim buffer, string

// Read local comma-separated file into variable
FileReadAll "c:\\data\\sometext.txt", buffer

// ... Replace commas with tabs, find first comma to begin with
…. Code fragment omitted …

// Connect to remote host and write tab separated data to the remote file
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
RAFileSave "/data/loaddata", buffer
RADisconnect
```

RAFileSize

Description: Reports the length of a remote file in bytes.

Syntax: RAFileSize file, return

Argument	Description
file	A string whose value is the name of the remote file for which you want to know the length. If file is not on the current working directory, you must specify the fully qualified file name
return	A numeric variable that receives the returned value

Return value: Number. Returns the length in bytes of the file identified by file.

Example:

```
Dim size, number
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
RAFileSize "/home/linuxuser/myfile", size
RADisconnect
```

RAFileTransfer

Description: Copies file from local computer to remote Linux/UNIX computer or from remote to local computer provided 24x7 Remote Automation Server is running at a given remote computer.

Syntax: RAFileTransfer method, source, destination

Argument	Description
Method	A string whose value describes which way to copy files. Must be one of the following: • "ToRemote" - transfers files from local to remote computer • "ToLocal" - transfers files from remote to local computer
Source	A string whose value is the name of file you want to transfer
Target	A string whose value is the name of destination files

Return value: None

Usage: RAFileTransfer statement allows sending files in both directions. If the target file already exists, the statement overwrites the existing file. The transfer time greatly depends on the size of the file and network bandwidth.

Example:

```
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
// upload file
RAFileTransfer "ToRemote", "c:\\data\\myfile.txt",
"/home/linuxuser/myfile"
// run program that converts the uploaded file
Dim pid, number
Dim output, string
RARunAndWait "/home/linuxuser/convert myfile", pid, output
// download converted file
RAFileTransfer "ToLocal", "/home/linuxuser/myfile",
"c:\\data\\myfile.bcp"
RADisconnect
```

RAFileReadAll

Description: Reads file on remote Linux/UNIX computer and stores contents of the file in a local script variable.

Syntax: RAFileReadAll file, buffer

Argument	Description
file	A string whose value is the name of the file that you want to read into the buffer
buffer	A string variable that receives the returned value

Return value: file contents

Usage: RAFileReadAll statement can be used to load remote files into memory of the local computer and than scan or manipulate file contents. For example, to verify whether remote process has completed successfully you may need to read the remote log file and search for certain messages in that file

Example:
```
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
Dim buffer, string
// read file
RAFileReadAll "/home/linuxuser/load.log", buffer
RADisconnect

// search for keywords indicating successful load
Dim pos, number
InStr buffer, " rows loaded ", pos

IfThen pos, SUCCESS
       // report error
       LogAddMessage "ERROR", "Load Failed"
       Exit

SUCCESS:
       // all done
```

Low-Level File Operations

RAFileOpen

Description: Opens the specified remote file for reading or writing and assigns it a unique file number. You use this number to identify the file when you read, write, or close the file.

Syntax: FileOpen file_name, file_mode, file_access, append, file_number

Argument	Description
file_name	A string whose value is the name of the remote file you want to open. If file_name is not on the operating system's search path, you must enter the fully qualified name
file_mode	A string constant whose value specifies how the file read or file write operation is determined. Values are: • "LineMode" — Read or write the file, one line at a time • "StreamMode" — Read the file in chunks. For more information, see Usage below. **The file mode parameter is not used at this time. It is reserved for future use.**
file_access	A string constant whose value specifies whether the file is opened for reading or writing. Values are: • "Read" — Read-only access • "Write" — Write-only access • "ReadWrite" — Both read and write access
append	A boolean whose value specifies whether existing data in the file is overwritten when file is opened for write operation. Values are: • True — Write data to the end of the file • False — Replace all existing data in the file append is ignored if the fileaccess argument is "Read"
file_number	A numeric variable that receives the returned file number.

Return value: Number. Returns the file number assigned to file_name.

Usage: You must open the remote file using RAFileOpen statement before you can use RAFileRead and RAFileWrite methods. After you are done with the file processing you should close it by calling RAFileClose statement.

A call to RAFileRead reads the rest of the file starting from the current position (until it encounters an EOF) or as much characters as specified by the maximum number of characters parameter, whichever is shorter.

RAFileOpen, RAFileRead, RAFileWrite, and RAFileClose statements care the same functions as regular FileOpen, FileRead, FileWrite, and FileClose statements used for working with local files, but they work with remote files residing on remote Linux/UNIX computers running 24x7 RA Servers.

All remote file operations are totally transparent. Remote file operations are performed just like if the remote files were residing on the local disk.

Example:

```
Dim file_no, number
// connect to the remote 24x7 agent
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
// open remote file for reading
RAFileOpen "/incoming/loaddata.hdr", "StreamMode", "Read", False, file_no
// read first 8 characters from the file
Dim records_in_file, string
RAFileRead file_no, 8, records_in_file
// close remote file
RAFileClose file_no
// close connection
RADisconnect
```

RAFileRead

Description: Reads data from the remote file associated with the specified file number, which was assigned to the file with the RAFileOpen statement.

Syntax: RAFileRead file_number, bytes, return

Argument	Description
file_number	A number whose value is the file number previously assigned to the file when it was opened by RAFileOpen statement
bytes	A number whose value indicates how many bytes you want to read from the file.
return	A string variable into which you want to read the data

Return value: String.

Usage: The file must be opened previously using RAFileOpen statement.

RAFileRead automatically positions the file pointer after each read operation so that it is ready to read the next chunk of data.

You can use the RAFileSize statement to check the file size before performing various reading operation. If your system has file sharing or security restrictions, you may need to call RAFileSize before you call RAFileOpen.

If you wan to read the entire file in one pass, use RAFileReadAll statement which is more efficient than performing using RAFileSize/RAFileOpen/RAFileRead/RAClose

The RAFileRead will fail if an end-of-file mark is encountered before the specified number of bytes is read.

Example:
```
Dim file_no, number
// connect to the remote 24x7 agent
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
// open remote file for reading
RAFileOpen "/incoming/loaddata.hdr", "StreamMode", "Read", False, file_no
// read first 8 characters from the file
Dim records_in_file, string
RAFileRead file_no, 0, 8, records_in_file
// close remote file
RAFileClose file_no
// close connection
RADisconnect
```

RAFileWrite

Description: Writes data to the remote file associated with the specified file number, which was assigned to the file with the RAFileOpen statement.

Syntax: RAFileWrite file_number, data

Argument	Description
file_number	A number whose value is the file number previously assigned to the file when it was opened by RAFileOpen statement
data	A string whose value is the text that you want to write to the file.

Return value: String.

Usage: The file must be opened previously using RAFileOpen statement.

RAFileWrite automatically positions the file pointer after each write operation so that it is ready to write the next chunk of data.

If you wan to overwrite the entire file in one pass, use RAFileSave statement which is more efficient than performing the same operation using RAFileOpen/RAFileWrite/RAClose

Example:

```
Dim file_no, number
// connect to the remote 24x7 agent
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
// open remote file for writing, overwrite if the file exists
RAFileOpen "/staging/loaddata.hdr", "StreamMode", "Write", False, file_no
// write first 8 characters to the file
RAFileWrite file_no, "header\n"

… perform some local or remote processing (perhaps, retrieve database
data) …

// write number of retrieved records to the remote file
RAFileWrite file_no, records
// Add end of line symbol
RAFileWrite file_no, "\n"
// close remote file
RAFileClose file_no
// close connection
RADisconnect
```

RAFileClose

Description: Closes the remote file associated with the specified file number. The file number was assigned to the file with the RAFileOpen statement.

Syntax: RAFileClose file_number

Argument	Description
file_number	The number assigned to the remote file you want to close. The RAFileOpen statements returns the file number when it opens the remote file.

Return value: None

Usage: The file must be opened previously using RAFileOpen statement.

Example:

```
Dim file_no, number
// connect to the remote 24x7 agent
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
// open remote file for reading
RAFileOpen "/incoming/loaddata.hdr", "StreamMode", "Read", False, file_no
// read first 8 characters from the file
Dim records_in_file, string
RAFileRead file_no, 8, records_in_file
// close remote file
RAFileClose file_no
// close connection
RADisconnect
```

RAFileGetPos

Description: Reports current position in the specified file previously opened by RAFileOpen statement.

Syntax: RAFileGetPos filenum, pos

Argument	Description
filenum	The file number previously assigned to the file when it was opened by RAFileOpen statement
pos	A numeric variable that receives the returned value

Return value: Number. Returns the file position after the read/write operation or zero if no operation has been performed after file opening.

Usage: RAFileGetPos always returns position from the beginning of the file.

Example:

```
Dim pos, number
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
// open remote file for writing, append data
RAFileOpen "/incoming/loaddata.hdr", "StreamMode", "Write", True, file_no
// append new data to the file
Dim my_data, string
// get data from some source
RAFileWrite file_no, my_data
// get new current position
RAFileGetPos file_no, pos
// close remote file
RAFileClose file_no
// close connection
RADisconnect
```

RAFileSetPos

Description: Moves the file pointer to the specified position in a file previously opened by RAFileOpen statement. The file pointer is the position in the file at which the next read or write begins.

Syntax: RAFileSetPos filenum, pos, origin

Argument	Description
filenum	The file number previously assigned to the file when it was opened by RAFileOpen statement
pos	A numeric variable that receives the returned value
origin	A string constant whose value specifies from where you want to set then position. Values are: • "START" — At the beginning of the file • "CURRENT" — At the current position • "END" — At the end of the file

Return value: None.

Usage: The file must be opened previously using RAFileOpen statement.

Example:

```
// connect to the host computer
RAConnect "123.45.789.01", 1096, "linuxuser", "valid password"
// open remote file for reading
RAFileOpen "/incoming/loaddata.hdr", "StreamMode", "Read", False, file_no
// set reading position to begin with 9th character
RAFileSetPos file_no, 9, "START"
// get next 8 characters from the source file
Dim my_data, string
RAFileRead file_no, 8, my_data
// close remote file
RAFileClose file_no
// close connection
RADisconnect
```

Licensing

1. **Single installation license:** A separate single installation license is required for every 24x7 Remote Automation Server installation.
2. **Site license:** 24x7 Automation Suite site license users don't need a separate license for using 24x7 Remote Automation Server for Linux/UNIX. If you have a site license you can install and use 24x7 Remote Automation Servers just as you use other components of the 24x7 Automation Suite. The usage of 24x7 Scheduler, 24x7 Remote Agent, 24x7 Remote Automation Server, 24x7 Remote Control, 24x7 Remote Control Java and 24x7 Remote Control COM is governed by the site license agreement.
3. **Redistribution:** You are not allowed to distribute or publish in any form the original or modified source and object code of the 24x7 Remote Automation Server. 24x7 Remote Automation Server for Linux/UNIX platforms is an integral part of 24x7 Automation Suite and may not be used separately as a standalone product. You may not use 24x7 Remote Automation Server with other third-party products without written permission by SoftTree Technologies, Inc.