24x7 Remote Control Java API Reference

24x7 Scheduler™ Multi-platform Edition 7.0

Copyright © SoftTree Technologies, Inc. 2004-2024
All rights reserved

Table of contents

ABOUT THIS MANUAL 5

CONVENTIONS USED IN THIS DOCUMENT 5
ABBREVIATIONS AND TERMS 5
TRADEMARKS 5

OVERVIEW 7

ABOUT THE 24X7 JAVA API 7
USING 24X7 JAVA API IN YOUR PROGRAM 7
A QUICK EXAMPLE: JAVA APPLICATION 8

MAIN CLASS – J24X7 9

CONSTANTS 9
FIELDS 9
CONSTRUCTOR 9
PUBLIC METHODS 10
PUBLIC METHODS 10
addAgentProfile 11
addCalendar 13
addCalendarDay 15
addDatabaseProfile 16
addHoliday 18
addJobQueue 19
addJobQueueEx 20
addTemplate 22
changeFolder 23
closeSession 24
createFolder 25
createJob 26
deleteAgentProfile 28
deleteCalendar 29
deleteCalendarDay 30
deleteDatabaseProfile 31
deleteFolder 32
deleteHoliday 33
deleteJob 34
deleteJobQueue 35
deleteTemplate 36
disableJob 37
enableJob 38
getAgentList 39
getAgentProfile 40
getCalendar 42
getCalendarDays 44
getCalendarList 46
getDatabaseList 48
getDatabaseProfile 49
getDriverList 51
getEventLog 53
getFolderList 55

getFolderProperty 56
getForecast 57
getGlobalVariable 58
getHolidays 59
getJobDefinition 61
getJobList 62
getJobListEx 64
getJobLog 66
getJobQueueSize 68
getJobQueueList 69
getJobQueueMonitor 70
getJobProperty 72
getJobPropertyEx 73
getJobStatus 75
getJobTemplateData 77
getLastError 79
getMonitor 80
getProcessList 82
getTemplate 83
getTemplateCatalog 84
getToken 85
getStatusReport 87
getUserRole 88
getVersion 89
holdJob 91
killJob 92
killShellCommand 93
logMessage 94
openSession 96
protectJob 98
releaseJob 99
queueJob 100
readFile 101
runJob 102
runScript 103
runShellCommand 104
setFolderProperty 106
setGlobalVariable 107
setJobProperty 108
setJobPropertyEx 109
setJobTemplateData 111
setTemplate 112
test 113
unprotectJob 114
updateAgentProfile 115
updateCalendar 116
updateDatabaseProfile 117
updateJob 118
updateJobQueue 119
updateJobQueueEx 120
utilRunScript 122
writeFile 124

ADDITIONAL CLASSES 125

BOOLEANHOLDER CLASS [COM.SYBASE.DPB] 126
INTEGERHOLDER CLASS [COM.SYBASE.DPB] 127

StringHolder Class [com.sybase.dpb] 128
RemoteException Class [com.sybase.dpb] 129

JOB PROPERTIES IN JDL FORMAT 130

INSTALLATION 136

Minimal System Requirements 136
Installation Steps 136
Post-installation Steps 136
Security Issues 136

LICENSING 136

About This Manual

This user manual describes all features of the 24x7 Remote Control Java API for 24x7 Scheduler™ Multi-platform Edition. This manual contains information for developers and experienced users of the 24x7 Scheduler who want to create custom interfaces to the 24x7 Scheduler and/or tightly integrate it with other applications and systems.

The complete 24x7 Scheduler documentation consists of several manuals and 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 JSP, API, and other.

Trademarks

24x7 Automation Suite, 24x7 Scheduler, DB Audit, DB Audit Expert, SoftTree SQL Assistant are trademarks of SoftTree Technologies, Inc.

Windows 10, 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.

Oracle is a registered trademark of Oracle Corporation.

IBM, DB2, UDB are registered trademarks of International Business Machines Corporation

All other trademarks appearing in this document are trademarks of their respective owners.
All rights reserved.

Overview

About the 24x7 Java API

24x7 Java API interface is a set of pure Java classes that enable communication between a Java application and a distributed 24x7 Scheduler server. The main class is *j24x7*, which is part of *com.softtreetech.scheduler* package. This class is provided as a JavaBeans™ proxy to the 24x7 Remote Control within 24x7 Scheduler running on the same or remote computer. The interface also includes *jdpb_24x7,* which is provided for backward compatibility.

Additional supporting and communication classes are contained in the *com.sybase.dpb* and *com.objectspace.jdl* packages that must be installed with the **24x7 Remote Control Java** interface. The *com.sybase.dpb* package is developed and copyrighted by Sybase, Inc. and the *com.objectspace.jdl* package is developed and copyrighted by ObjectSpace, Inc. All the required classes can be found in *JDPB24x7.jar* and *jgl3.1.0.jar* files installed with 24x7 Scheduler.

Only classes required for using the **24x7 Java** API interface are described in this manual. For more information see Additional Classes topic.

Using 24x7 Java API in Your Program

The following steps summarize typical programming tasks required to incorporate the methodality of the 24x7 Scheduler Java API in a client application.

To code a client application

1. Create an instance of a j24x7 class.
2. Connect to an active 24x7 Scheduler server or 24x7 Remote Agent by calling the **openSession** method of the j24x7 class. For more information about connecting to 24x7 remote servers, see openSession method description.

1. Call necessary 24x7 Java API methods for performing the required tasks. For more Information see Public Methods topic.
2. Terminate the connection to the 24x7 Scheduler server by calling the **closeSession** method of the j24x7 class. For more information, see closeSession topic.

For more information about managing 24x7 Schedulers and scheduled jobs see the 24x7 Scheduler User's Guide.

A Quick Example: Java application

The following paragraphs provides a quick example of calling **24x7 Java API** from a Java application.

1. Declare and instantiate j24x7 class variable

```
// declare imports
import com.softtreetech.scheduler.*;
import import com.sybase.dpb.*;

// instantiate j24x7 class
j24x7 remote24x7 = new j24x7();
```

2. Connect to the server:

```
// Open new session
Integer portNo = new Integer(1096);
String hostName = new String("LocalHost");
String userID = new String("scott");
String userPass = new String("tiger");
String serial = new String("[your license key here]");

remote24x7.openSession(hostName, portNo, userID, userPass, serial );
```

3. Access methods or properties of the 24x7 Remote Control

```
// Get job list
StringHolder buffer = new StringHolder("");
remote24x7.getJobList(buffer, new Boolean(false));

// Change start time for job #12 to 6:00 AM
remote24x7.setJobProperty("12", "START_TIME", "6:00");
```

4. Disconenct from the server

```
// Done. Close session
remote24x7.closeSession();
```

Main Class – j24x7

Constants

static int RC_ERROR static int RC_WARNING static int RC_INFO	These constants can be used with the `logMessage` method for the message Severity parameter. Use RC_ERROR to write an error message. Use RC_WARNING to write an warning message. Use RC_INFO to write all other informational messages.
static int SECURITY_OFF static int ROLE_ADMIN static int ROLE_STANDARD static int ROLE_RESTRICTED static int ROLE_GUEST	These constants can be used with the `getUserRole` method for the return value. `For more details, see` description of the getUserRole method.

Fields

All fields are read-only. Do not attempt to modify their values directly.

public boolean **connectedToRemote**	Indicates session status TRUE – session is opened FALSE – session is closed See `openSession` method for more details.

Constructor

public j24x7( )

The constructor for j24x7 class has no parameters.

Public Methods

All supported Java interface methods are described in the following topics. For your convenience methods are listed in alphabetical order.

addAgentProfile

Integer addAgentProfile (String ProfileName,
String ComMethod,
String Location,
String Port,
String Options) throws RemoteException

The **addAgentProfile** method creates new 24x7 Remote Agent or 24x7 Master Scheduler profile.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

profileName	Name of the new Remote Agent Profile. This name must be unique.
comMethod	Name of the communication driver to be used for the connections to the Agent Values are: • WinSock • NamedPipes – this method is deprecated and should not be used in new applications. **Important Note:** NamedPipes method is only valid for 24x7 Scheduler Windows Edition.
location	The location of the Agent. For the **WinSock** driver specify one of the following values: • The IP address (for example, 199.99.99.91) • The host name of the remote computer (network computer name in workgroup) • LocalHost (this indicates that the Agent resides on the local machine) For the **NamedPipes** driver specifies the location portion of the pipe name. The combination of the Location and Application values forms the pipe name. The pipe name is constructed as follows: \\location\PIPE\application. If no location is specified, a local pipe name is constructed using a dot (.) in the machine name portion.
port	For the **WinSock** driver specify either of the following: • The port number for the Agent (for example, 1096). Each server application requires a unique port number on the server machine. If you specify a port number, select a number that is greater than 4096 and less than 65536. • The service name for Agent. The service name is an indirect reference to the port number. The mapping of the service name to the port number is specified in the TCP/IP services file. Normally, this file (SERVICES) is located in the Windows directory for Win95/98/Me and in the C:\WINNT\SYSTEM32\DRIVERS\ETC directory for 24x7 Remote Control Java Interface - 11 - Windows

	NT4/2000/XP/2003. You may need to edit this file manually to add 24x7 Port or Service. For the **NamedPipes** driver specify the application portion of the pipe name. The combination of the Location and Application values forms the pipe name. The pipe name is constructed as follows: *\\location\PIPE\application*. The Application must be unique for the Location you specify.
options	**Important Note:** This parameter is only valid for 24x7 Scheduler Windows Edition. Specifies one or more additional communications options. If you specify more than one option, you need to separate the options with commas. For the Local driver this property is ignored. **BufSize=n** Sets the connection buffer size to the value specified. **MaxListeningThreads=n** Determines the maximum number of listening threads available on the Master Scheduler and Remote Agents. **MaxRetry=n** Specifies how many times the Remote Control will try to connect when the Agent's listening port is busy. Applies to the **WinSock** driver only. **NoDelay=1** Specifies that each packet be sent without delay. Corresponds to the TCP_NODELAY option. Setting this option may degrade performance significantly. Do not use this option unless you thoroughly understand its implications. Applies to the **WinSock** driver only. **RawData=1** Specifies that raw data be passed over the network. By default, the **WinSock** driver obscures the data that is passed over the network. Setting this option to 1 overrides the default behavior. Both the Remote Control and Agent must have the same setting. If there is a discrepancy between the Remote Control and Agent's settings, the communication fails. Setting this option to 1 may improve performance slightly. Applies to the **WinSock** driver only.

See also:

updateAgentProfile
deleteAgentProfile
getAgentList

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.addAgentProfile( "Print Server", "Winsock",
                                       "192.168.100.1", "1096", "");
if (retCode.intValue() != 1)
      /* Print error */
      System.out.println( obj.getLastError() );
```

addCalendar

Integer addCalendar (String Name,
Boolean Exclusive,
IntegerHolder CalendarID) throws RemoteException

The **addCalendar** method creates new business calendar.

Important Note: This method is only valid for 24x7 Scheduler Multi-platform Edition.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Name	Name of the new Business Calendar. This name should be unique.
Exclusive	Type of the calendar – exclusive means the calendar contains non-business dates only; inclusive means that the calendar contains business dates only
CalendarID	An IntegerHolder class variable used as an output parameter. In case of successful method execution, the value of the **CalendarID** parameter is populated with the unique numeric id of the new calendar. This ID can be used in the following code to add dates to the created calendar.
Description	Holiday name or description limited by 50 characters.

See also:

deleteCalendar
updateCalendar
addCalendarDay
getCalendarList

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

IntegerHolder calId = new IntegerHolder();
Integer retCode = obj.addCalendar( "Twice a Month Calendar"
                            new Boolean(false), calId );
if (retCode.intValue() != 1) {
       /* Print error /
       System.out.println( obj.getLastError() );
       return;
}

// Print new calendar id
System.out.println( "New calendar ID: " + calId.getValue() );
// add 2 days to the new calendar
addCalendarDay( calId, 2011, 2, 10, "First monthly report" );
```

```
addCalendarDay( calId, 2011, 2, 15, "Second monthly report " );
```

addCalendarDay

Integer addCalendarDay (Integer CalendarID,
Integer Year,
Integer Month,
Integer Day,
String Description) throws RemoteException

The **addCalendarDay** method adds new date to the specified business calendar.

Important Note: This method is only valid for 24x7 Scheduler Multi-platform Edition.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

CalendarID	Unique ID of the target Business Calendar
Year	Year part of the date. Supported range: 1900-3000
Month	Month part of the holiday date. Supported range: 1-12
Day	Day part of the holiday date. Supported range: 1-31. Day must be a valid day for the specified Year and Month.
Description	Optional description of the date.

See also:

deleteCalendar
deleteCalendarDay
getCalendarDays

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

IntegerHolder calId = new IntegerHolder();
Integer retCode = obj.addCalendar( "Twice a Month Calendar"
                          new Boolean(false), calId );
if (retCode.intValue() != 1) {
       /* Print error /
       System.out.println( obj.getLastError() );
       return;
}

// Print new calendar id
System.out.println( "New calendar ID: " + calId.getValue() );
// add 2 days to the new calendar
addCalendarDay( calId, 2011, 2, 10, "First monthly report" );
addCalendarDay( calId, 2011, 2, 15, "Second monthly report " );
```

addDatabaseProfile

Integer addDatabaseProfile (String ProfileName,
String DatabaseDriver,
String Server,
String DatabaseName,
Boolean AutoCommit,
String User,
String Password) throws RemoteException

The **addDatabaseProfile** method creates new database profile.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

ProfileName	Name of the new Database Profile. This name should be unique.
DatabaseDriver	The name of the database driver that you want to use for connection. This name must much any supported DBMS name as they are specified in the Database Profiles dialog in 24x7 Scheduler (see Tools/Database Profiles menu).
Server	(Optional) The database server name. The format for the name differs for different database systems. Specify an empty string "" if you connect to a local database and the server name is not required for connection. If you connect using ODBC (DatabaseDriver="ODBC") specify name of the desired ODBC profile as the Server name.
DatabaseName	(Optional) The name of the database. Specify an empty string "" if the database name is not required for connection.
AutoCommit	AutoCommit mode. Some databases support AutoCommit mode. Specify TRUE to turn AutoCommit on or specify FALSE otherwise. If DBMS does not support AutoCommit then this parameter is ignored.
User	The name of the user to logon to the database.
Password	The password to use to logon to the database.

See also:

updateDatabaseProfile
getDatabaseProfile
deleteDatabaseProfile
getDatabaseList

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.addDatabaseProfile( "Prod Server",
             "SQL Server", "Neptune",
             "warehouse", new Boolean(true), "sa", "******");
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

addHoliday

Integer addHoliday (Integer Year,
Integer Month,
Integer Day,
String Description) throws RemoteException

In 24x7 Scheduler Windows Edition, the **addHoliday** method adds new holiday record to the 24x7 Scheduler Holiday Table.

In 24x7 Scheduler Multi-platform Edition this method adds new exception date to the **[default]** calendar.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Year	Year part of the holiday date. Supported range: 1900-3000
Month	Month part of the holiday date. Supported range: 1-12
Day	Day part of the holiday date. Supported range: 1-31. Day must be a valid day for the specified Year and Month.
Description	Holiday name or description limited by 50 characters.

See also:

deleteHoliday
getHolidays

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.addHoliday( new Integer(2003), new Integer(11),
                                  new Integer(27), "Thanksgiving Day");
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

addJobQueue

Integer addJobQueue (String QueueName,
Integer MaxSize) throws RemoteException

The **addJobQueue** method creates new job queue.

Note: This method has been deprecated. Use addJobQueueEx method.

Note: In 24x7 Scheduler Windows Edition, adding new job queue does not have an immediate effect. The new queue is available to jobs only after the 24x7 Scheduler is restarted. In 24x7 Scheduler Multi-platform Edition, the new queue is available immediately to run jobs.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

QueueName	Name of the new Job Queue. This name should be unique.
MaxSize	Maximum queue size in Mbytes. 1 Mbytes should be sufficient in most cases.

See also:

addJobQueueEx
updateJobQueue
getJobQueue
deleteJobQueue
getJobQueueList

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

Integer retCode = obj.addJobQueue( "Payroll jobs", new Integer(1));
if (retCode.intValue() != 1)
      /* Print error */
      System.out.println( obj.getLastError() );
```

addJobQueueEx

Integer addJobQueueEx (String QueueName,
Integer MaxSize,
Integer MaxJobs,
Boolean NearCapacityAlerts,
String EmailSender,
String EmailPassword,
String EmailRecipients) throws RemoteException

The **addJobQueueEx** method creates new job queue.

Note: In 24x7 Scheduler Windows Edition, adding new job queue does not have an immediate effect. The new queue is available to jobs only after the 24x7 Scheduler is restarted. In 24x7 Scheduler Multi-platform Edition, the new queue is available immediately to run jobs.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

QueueName	Name of the new Job Queue. This name should be unique.
MaxSize	Maximum allowed disk space usage in Mbytes. 1 Mbytes should be sufficient in most cases. As of 24x7 Scheduler Multi-platform Edition v4.3, this queue parameter is reserved for future use. The parameter is used in 24x7 Scheduler Windows Edition only. Specify zero value for unlimited disk space usage.
MaxJobs	Maximum number of jobs allowed in a queue at any moment in time. The count includes both running and queued jobs, including jobs placed on hold. Specify zero value for unlimited number of jobs.
NearCapacityAlerts	Enables sending email alerts in case a queue is at or near its maximum capacity. The capacity is controlled by **MaxSize** and **MaxJobs** parameters
EmailSender	Email account (for MAPI email protocol) or email address (for SMTP email protocol) to use for sending "near capacity" and "over capacity" email alerts.
EmailPassword	Email password to use for authenticating email sender to the email server. Specify an empty string or null value if password is not required.
EmailRecipients	Comma-separated list of email recipient addresses for "near capacity" and "over capacity" email alerts.

See also:

updateJobQueueEx
getJobQueue
deleteJobQueue
getJobQueueList

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

Integer retCode = obj.addJobQueueEx( "Payroll jobs",
       new Integer(1), new Integer(200), new Boolean(true)
       "alerter@domain.com", null, "helpdesk@domain.com" );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

addTemplate

Integer addTemplate (String Section,
String Name,
String Template,
String Text) throws RemoteException

The **addTemplate** method creates new job template.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Section	Name of the template section in the catalog. **addTemplate** automatically creates the specified **Section** if it does not yet exist.
Name	Name of the new template. This name must be unique within the **Section**.
Template	Name of the new template file. The file name must be unique.
Text	Template text.

See also:

deleteTemplate
setTemplate
getTemplate
getJobTemplateData
getTemplateCatalog
setJobTemplateData

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// Obtain template code from the submitted form
String TemplateCode = request.getParameter("TEMPLATE_CODE");
// Create new template using template code
Integer retCode = obj.addTemplate( "FTP jobs",
                           "Upload Monthly Reports",
                           "%HOME%\\Templates\\ftp_month_reports.ini",
                            TemplateCode );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

changeFolder

Integer changeFolder (String JobID,
String TargetFolderID) throws RemoteException

The **changeFolder** method moves a job with the specified **JobID** to a job folder specified by the **TargetFolderID** value.
Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
TargetFolderID	Folder ID converted to string or folder name.

See also:

createJob
setJobProperty
createFolder

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// Move job #624 to folder with id -1
Integer retCode = obj.changeFolder( "624", "-7");
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );

// Move job "My job" to folder "My folder"
retCode = obj.changeFolder( "My job", "My folder");
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

closeSession

Integer closeSession ( ) throws RemoteException

The **closeSession** method closes work session with the 24x7 Master Scheduler or 24x7 Remote Agent and terminates the connection. The session must be previously opened using the **openSession** method.

The **closeSession** method does not destroy **j24x7 class**. The same class instance can be reused to open another session.

If you destroy j24x7 class instance and a session is still open, the j24x7 class automatically calls **closeSession**. In web based applications object clean up and destruction is often left to the web server so that in such applications you usually do not need to call **closeSession** explicitly.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters: None

See also:

openSession
Using 24x7 Java API in Your Program

Example:

```
// create j24x7 class
j24x7 obj = new j24x7();
// open session
obj.openSession( "LocalHost", new Integer(1096), "john_doe",
              "password", "00000-00000-00000");

// ... do something here ...

// close session
obj.closeSession();
```

createFolder

Integer createFolder (String FolderName,
String FolderDescription,
IntegerHolder FolderID) throws RemoteException

The **createFolder** method creates new job folder.

Return: Returns 1 if the method succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

If the method succeeds, **FolderID** parameter is populated with the ID of the new folder.

Parameters:

FolderName	The name of the new folder
FolderDescription	The description that you want to attach to the folder so other people can see how this folder should be used. You can specify an empty string if no description is needed.
FolderID	An IntegerHolder class variable used as an output parameter. In case of successful method execution, the ID of the newly created folder is copied to the **FolderID** parameter. This value is unique and it can be used in other methods as a unique folder identifier.

See also:

deleteFolder
getFolderProperty
setFolderProperty

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
StringHolder buffer = new StringHolder("");
Integer retCode = obj.createFolder( "Weekly Payroll",
      "Store payroll related jobs in this folder", buffer);
if (retCode.intValue() != 1)
      /* Print error */ System.out.println( obj.getLastError() );
else
      System.out.println("Weekly Payroll folder id is " + buffer);
```

createJob

Integer createJob (String JobDefinition) throws RemoteException

The **createJob** method creates new job and adds it to the active job pool.

Return: Job ID of the created job as a positive number or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobDefinition	The Job definition in JDL format. For a wide variety of examples see job templates available in the [24x7 install directory]\Template subdirectory. The default path is *C:\Program Files\24x7 Automation 3\Template*. For supported job properties and their JDL names see 24x7 User's Guide. As a quick reference, see Job Properties in JDL Format topic in this manual.

See also:

setJobProperty
setJobTemplateData
changeFolder
updateJob
disableJob
protectJob
runJob

Example 1:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// Get job definition from the submitted form
String JobDefinition = request.getParameter("JOB_DEF");
// create new job
Integer retCode = obj.createJob( JobDefinition );
if (retCode < 0)
       /* Print error */
       System.out.println( obj.getLastError() );
else
       System.out.println("Job ID: " + retCode.toString());
```

Example 2:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// create job definition
String jobDefinition = "NAME=My New Job\r\n" +
       "COMMAND=c:\\winnt\\notepad.exe\r\n" +
       "SCHEDULE_TYPE=D\r\n" +
       "FRIDAY=Y\r\n" +
       "SUNDAY=Y\r\n" +
       "MONDAY=Y\r\n" +
```

```
        "TUESDAY=Y\r\n" +
        "WEDNESDAY=Y\r\n" +
        "THURSDAY=Y\r\n" +
        "SATURDAY=Y\r\n" +
        "START_TIME=10:00\r\n" +
        "ASYNC=N\r\n" +
        "LOG=Y\r\n" +
        "DESCRIPTION=job description\r\n" +
        "JOB_TYPE=P\r\n" +
        "FOLDER=1\r\n" +
        "QUEUE=[default]";

// create new job
Integer retCode = obj.createJob( JobDefinition );
if (retCode < 0)
        /* Print error */
        System.out.println( obj.getLastError() );
else
        System.out.println("Job ID: " + retCode.toString());

// close session here ...
```

deleteAgentProfile

Integer deleteAgentProfile (String ProfileName) throws RemoteException

The **deleteAgentProfile** method deletes existing profile of 24x7 Remote Agent or 24x7 Master Scheduler.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

ProfileName	Name of an existing Remote Agent Profile.

See also:

addAgentProfile
updateAgentProfile

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.deleteAgentProfile( "QA Server" );
if (retCode.intValue() != 1)
      /* Print error */
      System.out.println( obj.getLastError() );
```

deleteCalendar

Integer deleteCalendar (Integer CalendarID) throws RemoteException

The **deleteCalendar** method deletes business calendar from 24x7 settings.

Important Note: This method is only valid for 24x7 Scheduler Multi-platform Edition.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

CalendarID	Numeric ID of the business calendar to be deleted.

See also:

addCalendar
getCalendar
deleteCalendarDay

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.deleteCalendar( new Integer(25) );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

deleteCalendarDay

Integer deleteCalendarDay (Integer CalendarID
Integer Year,
Integer Month,
Integer Day) throws RemoteException

The **deleteCalendarDay** method deletes the specified date from the specified business calendar.

Important Note: This method is only valid for 24x7 Scheduler Multi-platform Edition.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

CalendarID	Numeric ID of the business calendar to be deleted.

See also:

addCalendarDay
getCalendarDays
deleteCalendar

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// delete 10-Feb-2011 from calendar #25
Integer retCode = obj.deleteCalendarDay( new Integer(25),
       new Integer(2011), new Integer(2), new Integer(10) );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

deleteDatabaseProfile

Integer deleteDatabaseProfile (String ProfileName) throws RemoteException

The **deleteDatabaseProfile** method deletes existing database profile.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

ProfileName	Name of an existing Database Profile.

See also:

addDatabaseProfile
updateDatabaseProfile

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.deleteDatabaseProfile( "QA2" );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

deleteFolder

Integer deleteFolder (String FolderID) throws RemoteException

The **deleteFolder** method deletes job folder from 24x7 Job Database. If the specified folder contains jobs, all these jobs are also deleted.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

FolderID	Folder ID converted to string or folder name.

See also:

createFolder
setFolderProperty

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.deleteFolder( "Payroll jobs" );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

deleteHoliday

Integer deleteHoliday (Integer Year,
Integer Month,
Integer Day) throws RemoteException

In 24x7 Scheduler Windows Edition, the **deleteHoliday** method deletes matching holiday record from the 24x7 Scheduler Holiday Table.

In 24x7 Scheduler Multi-platform Edition this method deletes matching date from the **[default]** calendar.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Year	Year part of the holiday date. Supported range 1900-3000
Month	Month part of the holiday date. Supported range 1-12
Day	Day part of the holiday date. Supported range 1-31. Day must be a valid day for the specified Year and Month.

See also:

addHoliday
getHolidays

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.deleteHoliday( 2003, 11, 25 );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

deleteJob

Integer deleteJob (String JobID) throws RemoteException

The **deleteJob** method deletes job from both the active job pool and the 24x7 Job Database.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.

See also:

disableJob
createJob
deleteFolder

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.deleteJob( "246" );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

deleteJobQueue

Integer deleteJobQueue (String QueueName) throws RemoteException

The **deleteJobQueue** method deletes existing job queue.

Note: In 24x7 Scheduler Windows Edition deleting new job queue does not have an immediate effect. It takes effect only after the 24x7 Scheduler is restarted. In 24x7 Scheduler Multi-platform Edition queue can be deleted only if it is empty. The deletion takes immediate effect.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

QueueName	Name of an existing Job Queue.

See also:

addJobQueue
updateJobQueue
getJobQueue
getJobQueueList

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.deleteJobQueue( "Weekly processing" );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

deleteTemplate

Integer deleteTemplate (String Section,
String Name,
Boolean DeleteFile) throws RemoteException

The **deleteTemplate** method deletes template from the template catalog (TEMPLATE.INI) and optionally deletes the template file.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Section	Name of the template section in the template catalog from which to delete the template specified by **Name** parameter.
Name	Name of the template to be deleted.
DeleteFile	If **DeleteFile** parameter is set to TRUE the template source file is also deleted. If **DeleteFile** is set to FALSE the template name is deleted from the catalog, but the file is left on disk.

See also:

addTemplate
getTemplate
getTemplateCatalog
setTemplate
setJobTemplateData

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.deleteTemplate( "Web reports",
                              "FTP upload", new Boolean(false) );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

disableJob

Integer disableJob (String JobID) throws RemoteException

The **disableJob** method disables job and removes it from both the active job pool. The job is not deleted from the 24x7 Job Database and can be later enabled again.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name

See also:

enableJob
deleteJob

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.disableJob( "246" );
if (retCode.intValue() != 1)
      /* Print error */
      System.out.println( obj.getLastError() );
```

enableJob

Integer enableJob (String JobID) throws RemoteException

The **enableJob** method enables job and places it backs to the active job pool.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.

See also:

disableJob
deleteJob

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.enableJob( "246" );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

getAgentList

Integer getAgentList (StringHolder Buffer,
Boolean HTMLFormat) throws RemoteException

The **getAgentList** method obtains list of names of configured 24x7 Remote Agent Profiles.

If the method succeeds, **Buffer** variable is populated with the name list. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise, it is populated with a plain text containing each entry on a new line.

Note: You can use getToken method to parse the returned list and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the profile list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list of 24x7 Remote Agent Profile names. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

addAgentProfile
deleteAgentProfile
getAgentProfile

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
StringHolder buffer = new StringHolder("");
Integer retCode = obj.getAgentList( buffer, new Boolean(true) );
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
       // Print agent list
       System.out.println( buffer.getValue() );
```

getAgentProfile

Integer getAgentProfile (String ProfileName,
StringHolder ComMethod,
StringHolder Location,
StringHolder Port,
StringHolder Options) throws RemoteException

The **getAgentProfile** method retrieves properties of an existing 24x7 Remote Agent profile.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

If the method succeeds, all StringHolder parameters are populated with profile properties.

Parameters:

ProfileName	Remote Agent profile name.
ComMethod	A StringHolder class variable used as an output parameter. In case of successful method execution, name of the communication driver is copied to the **ComMethod** parameter.
Location	A StringHolder class variable used as an output parameter. In case of successful method execution, location of the agent is copied to the **Location** parameter.
Port	A StringHolder class variable used as an output parameter. In case of successful method execution, the communication port number is copied to the **Port** parameter. Use getValue() and toInteger() methods to convert the returned value to a number.
Options	A StringHolder class variable used as an output parameter. In case of successful method execution, the string value containing additional communications options is copied to the **Options** parameter.

See also:

addAgentProfile
updateAgentProfile
deleteAgentProfile
getAgentList

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

StringHolder method = new StringHolder("");
StringHolder location = new StringHolder("");
StringHolder port = new StringHolder("");
```

```
StringHolder options = new StringHolder("");

// get profile properties as a tab-separated string
Integer retCode = obj.getAgentProfile( "Oracle Agent",
                    method, location, port, options );
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
{
       System.out.println("Profile: Oracle Agent");
       System.out.println("Communication Method: " +
                           method.getValue());
       System.out.println("Host: " + location.getValue());
       System.out.println("Port: " + port.getValue());
       System.out.println("Options: " + options.getValue());
}
```

getCalendar

Integer getCalendar(Integer CalendarID,
StringHolder Name,
BooleanHolder Exclusive) throws RemoteException

The **getCalendar** method retrieves properties of an existing Business Calendar.

Important Note: This method is only valid for 24x7 Scheduler Multi-platform Edition.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

If the method succeeds, Name and Exclusive parameters are populated with calendar properties.

Parameters:

CalendarID	Numeric ID of the business calendar whose properties to return.
Name	A StringHolder class variable used as an output parameter. In case of successful method execution, name of the specified calendar is copied to the **Name** parameter.
Exclusive	A BooleanHolder class variable used as an output parameter. In case of successful method execution, type of the calendar is copied to the **Exclusive** parameter.

See also:

getCalendarList
getCalendarDays
addCalendar
deleteCalendar
updateCalendar

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

StringHolder name = new StringHolder("");
BooleanHolder excl = new BooleanHolder(false);

// get properties of calendar #25
Integer retCode = obj.getCalendar( 25, name, excl );
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
{
       System.out.println("Calendar name: " + name.getValue());
```

```
        System.out.println("Is exclusive: " + excl.getValue());
    }
```

getCalendarDays

Integer getCalendarDays(Integer CalendarID,
Integer Year
StringHolder Buffer,
Boolean HTMLFormat) throws RemoteException

The **getCalendar** method retrieves properties of an existing Business Calendar.

Important Note: This method is only valid for 24x7 Scheduler Multi-platform Edition.

If the method succeeds, **Buffer** variable is populated with the list of calendar dates and their properties. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise, it is populated with a plain text containing each date entry on a new line. Each line contains date year, month, day, and description separated by tab characters.

Note: You can use getToken method to parse the returned list and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

CalendarID	Numeric ID of the business calendar whose days to return.
Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the list is copied to the **Buffer**.
Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list of calendar days. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

getCalendarList
getCalendar
addCalendarDay
deleteCalendarDay

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

StringHolder buffer = new StringHolder("");

// get days for calendar #25
Integer retCode = obj.getCalendarDays( 25, buffer );
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
{
       // parse and convert output values and print them
       String[] date_entries = buffer.getValue().split("\n");
       for (int i = 0; i < date_entries.length; i++) {
              String[] elements = date_entries[i].split("\t");
              System.out.println("Date: " +
                     new Date(     Integer.parseInt(elements[0]),
                                   Integer.parseInt(elements[1]),
                                   Integer.parseInt(elements[2])
                     )
              );
              System.out.println("Description: " + elements[3]);
}
```

getCalendarList

Integer getCalendarList (StringHolder Buffer,
Boolean HTMLFormat) throws RemoteException

The **getCalendarList** method obtains list of all job calendars.

Important Note: This method is only valid for 24x7 Scheduler Multi-platform Edition.

If the method succeeds, **Buffer** variable is populated with the list of calendar Ids, names, and types. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise, it is populated with a plain text containing each entry on a new line. Each line contains calendar ID, name, and type separated by tab characters.

Note: You can use getToken method to parse the returned list and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the calendar list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the result. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

addCalendar
deleteCalendar
getCalendar
getCalendarDays
updateCalendar

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

StringHolder buffer = new StringHolder("");
Integer retCode = obj.getCalendarList(buffer, new Boolean(true) );
if (retCode.intValue() != 1) {
      // Print error
      System.out.println( obj.getLastError() );
}
```

```
        // Print calendar names and properties
        String[] cal_entries = buffer.getValue().split("\n");
        for (int i = 0; i < cal_entries.length; i++) {
               String[] elements = cal_entries[i].split("\t");
               System.out.println( "ID: " + elements[0]);
               System.out.println( "Name: " + elements[1]);
               System.out.println( "Exclusive: " + elements[2]);
}
```

getDatabaseList

Integer getDatabaseList (StringHolder Buffer,
Boolean HTMLFormat) throws RemoteException

The **getDatabaseList** method obtains list of names of configured Database Profiles.

If the method succeeds, **Buffer** variable is populated with the list of database profile names. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise, it is populated with a plain text containing each entry on a new line.

Note: You can use getToken method to parse the returned list and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the profile list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list of Database Profile names. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

addDatabaseProfile
deleteDatabaseProfile
getDatabaseProfile
getDriverList

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

StringHolder buffer = new StringHolder("");
Integer retCode = obj.getDatabaseList( buffer, new Boolean(true) );
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
       // Print agent list
       System.out.println( buffer.getValue() );
```

getDatabaseProfile

Integer getDatabaseProfile (String ProfileName,
StringHolder DatabaseDriver,
StringHolder Server,
StringHolder DatabaseName,
BooleanHolder AutoCommit,
StringHolder User,
StringHolder Password) throws RemoteException

The **getDatabaseProfile** method retrieves properties of an existing database profile.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

If the method succeeds, all StringHolder parameters are populated with profile properties.

Parameters:

ProfileName	Database profile name.
DatabaseDriver	A StringHolder class variable used as an output parameter. In case of successful method execution, name of the database driver is copied to the **DatabaseDriver** parameter.
Server	A StringHolder class variable used as an output parameter. In case of successful method execution, name of the database server is copied to the **Server** parameter.
DatabaseName	A StringHolder class variable used as an output parameter. In case of successful method execution, name of the database is copied to the **DatabaseName** parameter.
AutoCommit	A BooleanHolder class variable used as an output parameter. In case of successful method execution, state of the AutoCommit mode is copied to the **AutoCommit** parameter.
User	A StringHolder class variable used as an output parameter. In case of successful method execution, name of the user to logon to the database is copied to the **User** parameter.
Password	A StringHolder class variable used as an output parameter. In case of successful method execution, password used to logon to the database is copied to the **Password** parameter.

See also:

addDatabaseProfile
updateDatabaseProfile
deleteDatabaseProfile

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
StringHolder databaseDriver = new StringHolder("");
StringHolder server = new StringHolder("");
StringHolder databaseName = new StringHolder("");
StringHolder user = new StringHolder("");
StringHolder password = new StringHolder("");
BooleanHolder autoCommit = new BooleanHolder(new Boolean(false));

// get profile properties
Integer retCode = obj.getDatabaseProfile( "Prod Server",
                    databaseDriver, server, databaseName,
                    autoCommit, user, password );
if (retCode.intValue() != 1)
      // Print error
      System.out.println( obj.getLastError() );
else
{
      // Print profile properties
      System.out.println("Profile Properties");
      System.out.println("Profile name: Prod Server");
      System.out.println("Driver: " + databaseDriver.getValue());
      System.out.println("Server Name: " + server.getValue());
      System.out.println("Database Name: " +
                                 databaseName.getValue());
      System.out.println("AutoCommit Mode: " +
                                 autoCommit.getValue().toString());
      System.out.println("User: " + user.getValue());
      System.out.println("Password: " + password.getValue());
}
```

getDriverList

Integer getDriverList (StringHolder Buffer,
Boolean HTMLFormat) throws RemoteException

The **getDriverList** method obtains list of database interfaces/drivers installed and configured in the settings of the remote 24x7 Scheduler.

Note: The list of database drivers in 24x7 Scheduler Windows Edition is fixed and cannot be customized by users. The list of database drivers in 24x7 Scheduler Multi-platform Edition is user customizable and may be different on different systems.

If the method succeeds, **Buffer** variable is populated with the names of database drivers. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise, it is populated with a plain text containing each database driver on a new line.

Note: You can use getToken method to parse the returned text and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the database drivers list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the result. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

getDatabaseList
getDatabaseProfile

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

StringHolder buffer = new StringHolder("");
Integer retCode = obj.getDriverList( buffer, new Boolean(false) );
if (retCode.intValue() != 1)
      // Print error
      System.out.println( obj.getLastError() );
else
```

```
// Print retuned driver list
System.out.println( buffer.getValue() );
```

getEventLog

Integer getEventLog (String LogName, Integer Records, StringHolder Buffer, Boolean HTMLFormat) throws RemoteException

The **getEventLog** method obtains log records from a Windows Event Log

Important Note: This method is only valid for 24x7 Scheduler Windows Edition.

If the method succeeds, **Buffer** variable is populated with the values from the specified Windows event log. You can use this method to test obtain results of remote executed jobs and Windows processes that write to Windows event logs. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise, it is populated with a plain text containing each log entry on a new line. Each line contains tab-separated event time, message severity, event source, and messages.

Note: You can use getToken method to parse the returned text and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

LogName	A string value specifying log name. This can be one of the following values: • **Application** • **System** • **Security**
Records	An Integer value specifying how many most recent records to return from the specified event log.
Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the message list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the result. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

getJobLog
readFile

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// start some Windows service
Integer retCode = obj.runShellCommand(
                    "NET START some_service",
                    "", new Boolean(false), new Integer(60) );
if (retCode.intValue() != 1) {
       System.out.println( obj.getLastError() ); // Print error
       return;
}

// get event log records indicating service start status
StringHolder buffer = new StringHolder("");
retCode = obj.getEventLog( "Application", new Integer(10),
                           buffer, new Boolean(false) );
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
       // Print retuned event log records
       System.out.println( buffer.getValue() );
```

getFolderList

Integer getFolderList (StringHolder Buffer,
Boolean HTMLFormat) throws RemoteException

The **getFolderList** method obtains list of all job folders.

If the method succeeds, **Buffer** variable is populated with the list of job folder IDs and names. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise, it is populated with a plain text containing each entry on a new line. Each line contains folder ID and name separated by a tab character.

Note: You can use getToken method to parse the returned list and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the folder list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the result. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

createFolder
deleteFolder
setFolderProperty
getFolderProperty

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

StringHolder buffer = new StringHolder("");
Integer retCode = obj.getFolderList(buffer, new Boolean(true) );
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
       // Print folder list
       System.out.println( buffer.getValue() );
```

getFolderProperty

Integer getFolderProperty (String FolderID,
String PropertyName,
StringHolder Buffer) throws RemoteException

The **getFolderProperty** method obtains value of the specified job **PropertyName** for the specified **JobID**. The **PropertyName** must be a valid JDL job property name. The following folder properties are supported: "FOLDER_NAME", "DESCRIPTION", "FOLDER", "MODIFY_TIME", "MODIFY_USER", "MODIFY_TERMINAL"

If the method succeeds, **Buffer** variable is populated with the value of the folder property.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

FolderID	Folder ID converted to string or Folder name.
PropertyName	The folder property name.
Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the property value is copied to the **Buffer**.

See also:

createFolder
setFolderProperty

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

StringHolder buffer = new StringHolder("");
Integer retCode = obj.getFolderProperty( "-12",
                                         "FOLDER_NAME", buffer );
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
       // Print folder name
       System.out.println( buffer.getValue() );
```

getForecast

Integer getForecast (StringHolder Buffer,
Boolean HTMLFormat) throws RemoteException

The **getForecast** method obtains 7-days job forecast.

If the method succeeds, **Buffer** variable is populated with job forecast report. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise, it is populated with a plain text containing each entry on a new line.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the job list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list of forecasted jobs. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

getMonitor
getJobQueueMonitor

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

StringHolder buffer = new StringHolder("");
Integer retCode = obj.getForecast( buffer, new Boolean(true) );
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
       // Print report
       System.out.println( buffer.getValue() );
```

getGlobalVariable

Integer getGlobalVariable (String VariableName,
StringHolder Buffer) throws RemoteException

The **getGlobalVariable** method obtains value of the global variable on the target 24x7 Remote Agent or 24x7 Master Scheduler.

If the method succeeds, **Buffer** variable is populated with the returned value.

Important Note: This method is only valid for 24x7 Scheduler Windows Edition.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

VariableName	The name of the global variable that must exist on the target 24x7 Agent or Master Scheduler. Variable names are case-insensitive.
Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the value of the specified global variable is copied to the **Buffer**.

See also:

setGlobalVariable
runScript

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

StringHolder buffer = new StringHolder("");
Integer retCode = obj.getGlobalVariable( "my_variable", buffer );
if (retCode.intValue() != 1)
{
	// Print error
	System.out.println( obj.getLastError() );
}
else
{
	// Print the returned global value
	System.out.println( buffer.getValue() );
}
```

getHolidays

Integer getHolidays (Integer Year,
StringHolder Buffer,
Boolean HTMLFormat) throws RemoteException

The **getHolidays** method obtains list of holidays for the specified **Year**.

In 24x7 Scheduler Windows Edition, the **getHolidays** returns dates from the 24x7 Scheduler Holiday Table.

In 24x7 Scheduler Multi-platform Edition this method returns dates from the **[default]** calendar.

If the method succeeds, **Buffer** variable is populated with the list of holidays. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise, it is populated with a plain text containing each entry on a new line. Each line contains holiday date and description separated by tab characters.

Note: You can use getToken method to parse the returned list and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Year	The year for which you want to obtain the list of holidays.
Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the holiday list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list of Holidays. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

addHoliday
deleteHoliday

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

StringHolder buffer = new StringHolder("");
Integer retCode = obj.getHolidays( new Integer(2003), buffer,
```

```
                                                new Boolean(true) );
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
       // Print holidays
       System.out.println( buffer.getValue() );
```

getJobDefinition

Integer getJobDefinition (String JobID,
StringHolder Buffer,
Boolean HTMLFormat) throws RemoteException

The **getJobDefinition** method obtains values of all job properties for the specified **JobID**.

If the method succeeds, **Buffer** variable is populated with the job definition. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as an HTML table. When displayed in a Web browser the table looks like the Job Properties View presented in the 24x7 GUI. If **HTMLFormat** is FALSE, the **Buffer** variable is populated with a plain text containing all job properties in JDL file format. The returned data is compatible with the createJob method.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the job properties are copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the result. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

createJob
getJobProperty
getFolderProperty
getJobTemplateData
updateJob

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

StringHolder buffer = new StringHolder("");
Integer retCode = obj.getJobDefinition( "Stock Tracker Job",
                          buffer, new Boolean(true) );
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
       // Get returned definition
       String JobDef = buffer.getValue();
```

getJobList

Integer getJobList (StringHolder Buffer,
Boolean HTMLFormat) throws RemoteException

The **getJobList** method obtains list of all jobs.

If the **HTMLFormat** is TRUE, **Buffer** variable is populated with a text formatted as an HTML table; otherwise the **Buffer** variable is populated with a plain text containing all job IDs and names. Each job appears on a new line. Job IDs and names are separated by tab characters.

Note: You can use getToken method to parse the returned list and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Sample output (with **HTMLFormat** set to FALSE):
1 test job 1
5 Backup job
4 Database export
27 Database replication

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the job list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the result. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:
getJobListEx
getFolderList

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// Obtain job list as HTML table
StringHolder buffer = new StringHolder("");
Integer retCode = obj.getJobList( buffer, new Boolean(true) );
if (retCode.intValue() != 1)
      // Print error
      System.out.println( obj.getLastError() );
```

```
else
        // Print job list
        System.out.println( buffer.getValue() );
```

getJobListEx

Integer getJobListEx (StringHolder Buffer,
Boolean HTMLFormat,
String PropertyList) throws RemoteException

The **getJobListEx** method obtains list of all jobs and optionally their properties.

The **getJobListEx** is an extended version of getJobList method. **getJobListEx** allows specifying additional job properties for inclusion in the returned job list. This generally provides much better performance as compared to first calling **getJobList** and then calling getJobProperty method for every returned job in order to obtain job properties.

Job ID and Job Name properties are always included in the returned list that is why they need not be specified in the **PropertyList** parameter. Use **PropertyList** parameter to specify additional JDL properties. Separate multiple properties by commas. For example: "FOLDER_NAME,DISABLED,MODIFY_TIME".

If the **HTMLFormat** is TRUE, **Buffer** variable is populated with a text formatted as an HTML table; otherwise the **Buffer** variable is populated with a plain text containing all job IDs and names and other job parameters specified by **PropertyList** . Each job appears on a new line. Job IDs, names and additional properties are separated by tab characters.

Note: You can use getToken method to parse the returned list and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Sample output (with **HTMLFormat** set to FALSE and **PropertyList** parameter set to "FOLDER_NAME,DISABLED"):

1	test job 1	Backup jobs folder	Y
5	Backup job	Backup jobs folder	N
4	Database export	Database jobs	N
27	Database replication	Database jobs	N

If the **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as an HTML table; otherwise the **Buffer** variable is populated with a plain text containing all job IDs and names. Each job appears on a new line. Job IDs, names, and additional properties specified by **PropertyList** are separated by tab characters.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the job list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the result. **Value** **Meaning** True format output as HTML text False format output as plain text

PropertyList	A comma separated list of job property names. For supported job properties and their JDL names see 24x7 User's Guide. As a quick reference, see Job Properties in JDL Format topic in this manual.

See also:

getJobList
getFolderList
getJobProperty
getJobPropertyEx

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// Obtain job list as HTML table
StringHolder buffer = new StringHolder("");
Integer retCode = obj.getJobListEx( buffer, new Boolean(true),
                                    "JOB_TYPE,DISABLED" );
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
       // Print job list
       System.out.println( buffer.getValue() );
```

getJobLog

Integer getJobLog (String JobID,
StringHolder Buffer,
Boolean HTMLFormat) throws RemoteException

The **getJobLog** method obtains all log records for the specified **JobID**.

If the **HTMLFormat** is TRUE, **Buffer** variable is populated with a text formatted as an HTML table. When displayed in a Web browser the table looks like the Job Log in the 24x7 GUI. If **HTMLFormat** is FALSE, the **Buffer** variable is populated with a plain text containing log records. Each record appears on a new line with columns separated by tab characters. The format of the text is the same as format of the job log file SCHEDULE.LOG.

Notes:

- To obtain the entire log for all jobs specify "0" for the **JobID** parameter.
- You can use getToken method to parse the returned list and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the job log is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the result. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

getForecast
getMonitor

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// Obtain job log for job #285 as HTML table
StringHolder buffer = new StringHolder("");
Integer retCode = obj.getJobLog( "DoUpdates Job",
                                 buffer, new Boolean(true) );
```

```
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
       // Print log
       System.out.println( buffer.getValue() );
```

getJobQueueSize

Integer getJobQueueSize (String QueueName,
IntegerHolder MaxSize) throws RemoteException

The **getJobQueueSize** method retrieves maximum size of an existing job queue.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

If the method succeeds, **MaxSize** parameter is populated with job queue size value.

Parameters:

QueueName	Job Queue name.
MaxSize	A IntegerHolder class variable used as an output parameter. In case of successful method execution, the maximum job queue size is copied to the **MaxSize** parameter.

See also:

addJobQueue
updateJobQueue
deleteJobQueue

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...


// get queue properties
IntegerHolder maxSize = new IntegerHolder(new Integer(0));
Integer retCode = obj.getJobQueueSize( "Weekly jobs", maxSize );
if (retCode.intValue() != 1)
{
      // Print error
      System.out.println( obj.getLastError() );
}
else
{
      // Print queue properties
      AnsiString queueSize = new String(maxSize.getValue());
      System.out.println("Queue Properties");
      System.out.println("Queue name: Weekly jobs");
      System.out.println("Maximum queue Size: " + queueSize + " MB");
}
```

getJobQueueList

Integer getJobQueueList(StringHolder Buffer,
Boolean HTMLFormat) throws RemoteException

The **getJobQueueList** method obtains list of names of configured Job Queues.

If **HTMLFormat** is TRUE, **Buffer** variable is populated with a text formatted as a HTML table; otherwise, it is populated with a plain text containing each entry on a new line.

Note: You can use getToken method to parse the returned list and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the queue list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list of 24x7 Job Queue names. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

getJobQueue
getJobQueueMonitor
addJobQueue

Example:

```
	// ... create j24x7 class (obj) and open session here ...
	// ... For more details, see openSession topic ...
	StringHolder buffer = new StringHolder("");
	Integer retCode = obj.getJobQueueList( buffer, new Boolean(false) );
	if (retCode.intValue() != 1)
		System.out.println( obj.getLastError() );  // Print error
	else
		System.out.println( buffer.getValue() ); // Print list of
queues
```

getJobQueueMonitor

Integer getJobQueueMonitor (String QueueName,
StringHolder Buffer,
Boolean HTMLFormat) throws RemoteException

The **getJobQueueMonitor** method obtains list of running and waiting jobs in the specified job queue. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise, it is populated with a plain text containing each entry on a new line. Each line contains the following tab-separated columns: queue name, unique job run number for that queue, job submition date and time, job priority (High, Normal or Low), job status (Awaiting, Running or Held), job ID, job name, job start time (for running jobs only), size of uncompressed job definition and linked job deployement data, size of compressed job definition and linked job deployement data, compression ratio. The same columns are available in the graphical version of the job queue monitor.

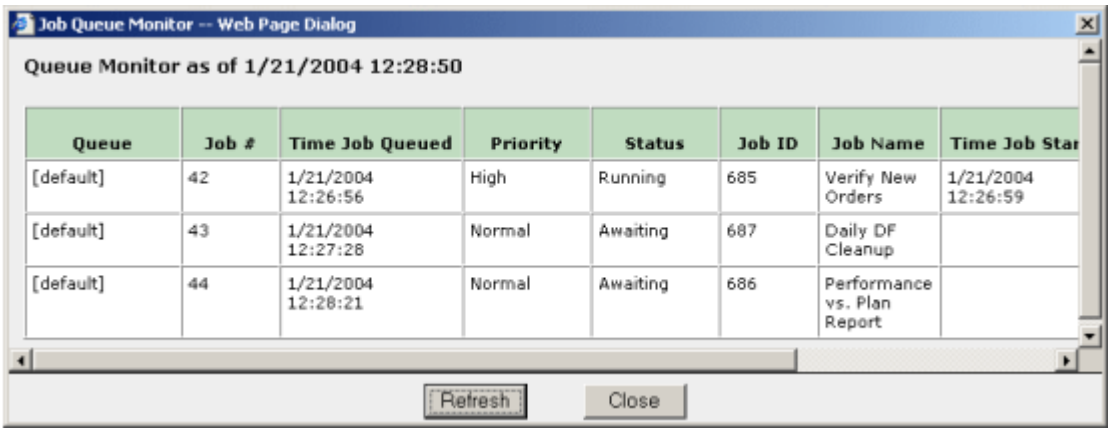
Note: You can use getToken method to parse the returned list and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

QueueName	Job Queue name.
Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the job monitor data is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list of forecasted jobs. **Value** **Meaning** True format output as HTML text False format output as plain text

Sample queue monitor screenshot



See also:

getForecast
getMonitor

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
StringHolder buffer = new StringHolder("");
Integer retCode = obj.getJobQueueMonitor( "Weekly Jobs", buffer,
                                          new Boolean(true) );
if (retCode.intValue() != 1)
       System.out.println( obj.getLastError() );  // Print error
else
       System.out.println( buffer.getValue() ); // Print monitor data
                                                // as HTML table
```

getJobProperty

Integer getJobProperty (String JobID,
String PropertyName,
StringHolder Buffer) throws RemoteException

The **getJobProperty** method obtains value of the specified job property. The **PropertyName** must be a valid JDL job property name. For information on supported property names and their values see "Job Property Names" topic in the 24x7 on-line help file.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name. Specify ID or name of the job whose property your want to retrieve.
PropertyName	The job property name. Specify name of the property whose value your want to retrieve. For supported job properties and their JDL names see 24x7 User's Guide. As a quick reference, see Job Properties in JDL Format topic in this manual.
Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the property value is copied to the **Buffer**.

See also:

setJobProperty
getJobPropertyEx
getJobDefinition
getJobTemplateData

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// Retrieve value of QUEUE property for job #235
StringHolder buffer = new StringHolder("");
Integer retCode = getJobProperty( "235", "QUEUE", buffer );
if (retCode.intValue() != 1)
       System.out.println( obj.getLastError() );  // Print error
else
       String queueName = buffer.getValue(); // copy returned value
```

getJobPropertyEx

Integer getJobPropertyEx (String JobID,
String PropertyNameList,
StringHolder Buffer) throws RemoteException

The **getJobPropertyEx** method obtains values of one or more job properties. The **PropertyNameList** must contain comma-separated list of valid JDL job property names. For information on supported property names and their values see "Job Property Names" topic in the 24x7 on-line help file.

Note: getJobPropertyEx method is an extended version of getJobProperty method. **getJobPropertyEx** is capable of returning values of multiple properties at once while **getJobProperty** can return only one property at a time. **getJobPropertyEx** is a more efficient and faster method to get multiple properties as it makes just one pass to the 24x7 Scheduler server.

Note: You can use getToken method to parse the returned list and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name. . Specify ID or name of the job whose properties your want to retrieve.
PropertyNameList	Comma-separated list of job property names. Specify names of properties whose values your want to retrieve. For supported job properties and their JDL names see 24x7 User's Guide. As a quick reference, see Job Properties in JDL Format topic in this manual.
Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, property values are copied to the **Buffer** as a tab delimited list.

See also:

getJobProperty
getJobDefinition
setJobProperty
setJobPropertyEx
getJobTemplateData

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// Retrieve values of MSG_FILE, MOVE_DIR, RENAME_SUFFIX properties
// for job #235
StringHolder buffer = new StringHolder("");
Integer retCode = obj.getJobPropertyEx( "235",
             "MSG_FILE,MOVE_DIR,RENAME_SUFFIX", buffer );
if (retCode.intValue() != 1)
       System.out.println( obj.getLastError() );  // Print error
else
{
       // Break the returned value into individual property values
       String fileName = obj.getToken(buffer, "\t");
       String dirName  = obj.getToken(buffer, "\t");
       String suffix = buffer.getValue();
}
```

getJobStatus

Integer getJobStatus (String JobID) throws RemoteException

The **getJobStatus** method obtains the status of the specified job.

Return: Returns a number indicating job status, which could be one of the following:

-1	Error job. Job may have this status because of incomplete job definition or if an error occurred while 24x7 was executing this job.
-2	Job is performing notification action after or before job run, such as sending email message, executing database command, etc.
-3	Job is running.
-5	First job run is pending. This job has been never been started before.
-6	Unknown status. Job may have an unknown status when it is a "Program" type job and the operation system is unable to report exit code of the job process.
0	Successfully finished. Note: an asynchronous job of "Program" type also may have this status after it successfully started the specified program and the started program is still running.
other	Successfully finished (valid for jobs of "Program" type only).

Parameters:

JobID	Job ID converted to string or job name.

See also:

runJob
getForecast
getJobQueueMonitor
getMonitor

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// Run job #235 in the background
Integer retCode = obj.runJob( "235", true );

// ... do something here ...

// Get job status
Integer jobStatus = obj.getJobStatus( "235");
if (jobStatus.intValue() >= 0)
```

```
       System.out.println("Job #235 completed successfully");
else if (jobStatus.intValue() == -2 or jobStatus.intValue() == -3)
       System.out.println("Job #235 completed successfully");
else
       System.out.println("Job #235 failed. Check the job log for " +
                      "details");
```

getJobTemplateData

Integer getJobTemplateData (String JobID,
StringHolder TemplateName,
StringHolder TemplateFile,
StringHolder Data) throws RemoteException

The **getJobTemplateData** method obtains template properties and data used to create or modify the specified job. This data must be previously saved using setJobTemplateData method.

If the **getJobTemplateData** method succeeds **TemplateName** parameter returns name of the original job template, the **TemplateFile** parameter returns name the original job template file and the **Data** parameter returns the saved data.

Return: Returns a number indicating method status, which could be one of the following:

1	Success. Both job definition and job template data have been found and all job ID, template name and file, job and template timestamps match exactly.
0	Success with warnings. Both job definition and job template data have been found but either the job or the template have been modified since the data was saved using **setJobTemplateData** method. It is still should be possible to continue. Use the getLastError method to obtain the warnings.
A negative number	Failure. An error has occurred during method call or if the template data cannot be found. Use the getLastError method to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
TemplateName	A StringHolder class variable used as an output parameter. In case of successful method execution the saved job template name is copied to this parameter.
TemplateFile	A StringHolder class variable used as an output parameter. In case of successful method execution the saved job template file name is copied to this parameter.
Data	A StringHolder class variable used as an output parameter. In case of successful method execution saved job data is copied to this parameter.

See also:

setJobTemplateData
getJobDefinition
getJobProperty

getTemplate

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

StringHolder templateName = new StringHolder("");
StringHolder templateFile = new StringHolder("");
StringHolder templateData = new StringHolder("");
Integer retCode = obj.getJobTemplateData( "235", templateName,
                                templateFile, templateData );
if (retCode.intValue() < 0)
       // Print error message
       System.out.println( "getJobTemplateData error: " +
                                obj.getLastError() );
else
{
       // print results
       System.out.println( "Template name: " +
                                templateName.getValue() );
       System.out.println( "Template file name: " +
                                templateFile.getValue() );
       System.out.println( "Job creation data: " +
                                templateData.getValue() );
}
```

getLastError

String getLastError ( ) throws RemoteException

The **getLastError** method returns error description for the last detected error.

You can use this method to obtain the error description when a method's return value indicates that it failed. You should call **getLastError** immediately after a method call fails.

Parameters: None

Example: See examples for other methods

getMonitor

Integer getMonitor (Integer Interval,
StringHolder Buffer,
Boolean HTMLFormat) throws RemoteException

The **getMonitor** method obtains list of running and pending jobs for the specified time interval (hours). Jobs having non-time based schedules are not included in the pending jobs portion of the report as the 24x7 Scheduler has now way to know when such jobs start conditions will be satisfied.

If the method succeeds, **Buffer** variable is populated with job monitor report. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise, it is populated with a plain text containing each entry on a new line. Each line contains the following tab-separated columns: job ID, job name, job status, and job last start time.

Note: You can use getToken method to parse the returned list and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Sample output (with **HTMLFormat** set to FALSE):

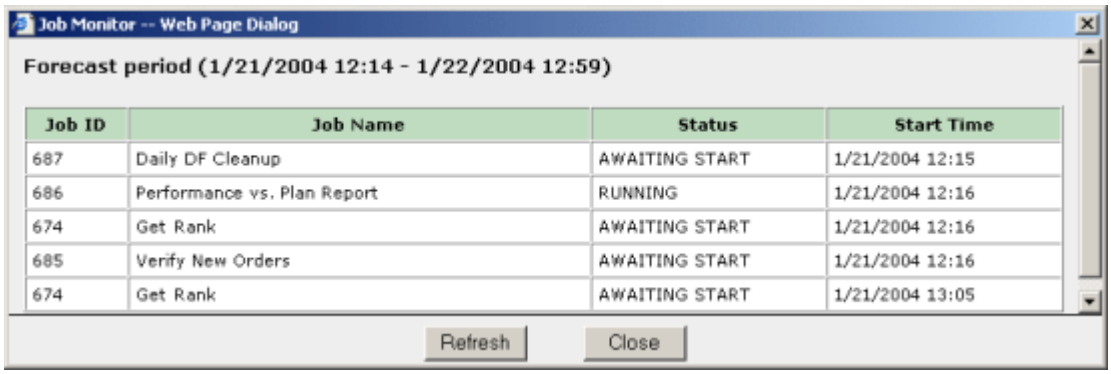
285	Backup job	AWAITING START	10/11/2002 04:20:00 PM
14	BCP	RUNNING	10/11/2002 03:00:00 PM
112	Cube building	RUNNING	10/11/2002 03:10:00 PM
56	Database export	AWAITING START	10/12/2002 02:00:00 AM

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Interval	The time interval within which the monitor report is created. The interval is specified in hours.
Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the job list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list of forecasted jobs. **Value** **Meaning** True format output as HTML text False format output as plain text

Sample monitor screenshot



See also:

getForecast
getJobQueueMonitor

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
StringHolder buffer = new StringHolder("");
Integer retCode = obj.getMonitor( new Integer(24),
                                  buffer,
                                  new Boolean(true) );
if (retCode.intValue() != 1)
      System.out.println( obj.getLastError() );  // Print error
else
      System.out.println( buffer.getValue() ); // Print monitor data
```

getProcessList

Integer getProcessList (StringHolder Buffer,
Boolean HTMLFormat) throws RemoteException

The **getProcessList** method obtains list of running system processes "visible" to the process runing 24x7 Scheduler

If the method succeeds, **Buffer** variable is populated with process report. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise, it is populated with a plain text containing each entry on a new line. The data available on each line may differ for different operation systems and depended on the output of the system **ps** command executed without parameters.

Note: You can use getToken method to parse the returned list and extract individual elements or convert it into an array of elements using standard split method of java.lang.String class. For more details, see description and examples for **getToken** method.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the process list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list active processes. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

getForecast
getMonitor
runShellCommand
runJob

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
StringHolder buffer = new StringHolder("");
Integer retCode = obj.getProcessList(buffer,
                                     new Boolean(true) );
if (retCode.intValue() != 1)
       System.out.println( obj.getLastError() );  // Print error
else
       System.out.println( buffer.getValue() ); // Print process data
```

getTemplate

Integer getTemplate (String TemplateFile,
StringHolder Buffer) throws RemoteException

The **getTemplate** method obtains contents of the specified template file.

If the method succeeds, **Buffer** variable is populated with the template text.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

TemplateFile	Template file name.
Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, text of the template file is copied to the **Buffer**.

See also:

setTemplate
getTemplateCatalog

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
StringHolder buffer = new StringHolder("");
Integer retCode = obj.getTemplate( "%HOME%\\Template\\ftp.ini",
                                   buffer );
if (retCode.intValue() != 1)
       System.out.println( obj.getLastError() );  // Print error
else
       System.out.println( buffer.getValue() );  // Print template
```

getTemplateCatalog

Integer getTemplateCatalog (StringHolder Buffer) throws RemoteException

The **getTemplateCatalog** method obtains contents of the TEMPLATE.INI file. TEMPLATE.INI stores catalog of available job templates. The file must exist in the 24x7 Scheduler home directory.

If the method succeeds, **Buffer** variable is populated with the template catalog.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, contents of the TEMPLATE.INI file is copied to the **Buffer**.

See also:

setTemplate
getTemplate
addTemplate

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
StringHolder buffer = new StringHolder("");
Integer retCode = obj.getTemplateCatalog( buffer );
if (retCode.intValue() != 1)
       System.out.println( obj.getLastError() );  // Print error
else
       System.out.println( buffer.getValue() );  // Print catalog
```

getToken

String getToken (StringHolder Buffer,
String Separator) throws RemoteException

The **getToken** method parses the source string (**Buffer**) and obtains first token. If the token is found it removes it along with the following **Separator** string from the source string.

Return: Returns found token or an empty string if the specified **Separator** value cannot be found or if it is found in the beginning of the **Buffer**.

Parameters:

Buffer	A StringHolder class variable used as an input and output parameter. In case of successful method execution, value of the **Buffer** parameter is modified. The new value does not include the found token and the following separator.
Separator	A string whose value is searched within the source string. The **Separator** value must consist of one or more characters.

See also: getToken can be used with many methods that return textual lists and tab-separated results, including

getJobList
getFolderList
getJobPropertyEx
getMonitor
getJobQueueMonitor
and other.

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
int count = 0;
StrinHolder buffer = new StringHolder("");

// Obtain job list using getJobListEx method
Integer retCode = obj.getJobListEx( buffer, new Boolean(false),
                                    "JOB_TYPE" );
if (retCode.intValue() != 1)
{
      // Print error message
      System.out.println( obj.getLastError() );
}
else
{
      // Parse job list and calculate how many jobs we got
      StringHolder temp = new StringHolder( buffer.getValue() );
```

```
        while(!temp.getValue().equals(""))
        {
               obj.getToken(temp, "\t");
               count ++;
        }

        // allocate memory for arrays for job IDs, names and types
        String[] jobID = new String[count];
        String[] jobName = new String[count];
        String[] jobType = new String[count];

        // Fill array elements
        count = 0;
        while(!buffer.getValue().equals(""))
        {
               jobID[count] = obj.getToken(buffer, "\t");
               jobName[count] = obj.getToken(buffer, "\t");
               jobType[count] = obj.getToken(buffer, "\r\n");
               count ++;
        }

        // Print job count
        System.out.println(count.toString() + " jobs found");
}
```

getStatusReport

Integer getStatusReport (String DestinDir) throws RemoteException

The **getStatusReport** method obtains **24x7 Status Report** from the 24x7 Master Scheduler and copies it to the destination directory specified in the **DestinDir** variable.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

DestinDir	The name of the directory into which the **24x7 Status Report** files will be copied. The **DestinDir** can contain either an absolute path to the directory in the format *disk:**dir**subdir1*\...*subdirX* or a relative path in the format *subdir1*\...*sudirX* or a network path name like *SERVER**VOLUME**dir**subdir1*\...*subdirX.* Note: The **DestinDir** must be an existing directory and the directory must be accessible from the computer running 24x7 Scheduler server.

See also:

getJobLog

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode =
      obj.getStatusReport( "\\\\WEBSERV\\web\\root\\24x7\\logs" );
if (retCode.intValue() != 1)
      // Print error
      System.out.println( obj.getLastError() );
else
      // Redirect to Status Reports
      response.sendRedirect( "..\\24x7\\logs\\index.htm" );
```

getUserRole

Integer getUserRole (String UserName) throws RemoteException

The **getUserRole** method obtains ID of the security member group (e.g. user role) for the specified user.

Return: The following values can be returned.
SECURITY_OFF – the security system is turned off. All users have unlimited privileges.
ROLE_ADMIN – user is a member of Administrators group.
ROLE_STANDARD - user is a member of Standard Privileges group
ROLE_RESTRICTED - user is a member of Restricted Privileges group
ROLE_GUEST - user is a member of Guest group
-1 – an error has occurred. Use the getLastError method to obtain the error message.

For more information about returned values see Constants topic.

Parameters:

UserName	The name of the user as it is specified in the 24x7 Scheduler security settings. Names are case-insensitive.

See also:
openSession

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.getUserRole( "scott" );
if (retCode == -1)
       System.out.println( obj.getLastError() ); // Print error
else
       if (retCode != obj.ROLE_ADMIN)
              System.out.println(
              "You are not authorized to use this method" );
```

getVersion

Integer getVersion (StringHolder CodeBase, StringHolder OSVersion, IntegerHolder MajorVersion, IntegerHolder MinorVersion, IntegerHolder BuildNumber) throws RemoteException

The **getVersion** method obtains version numbers and type of scheduler the API session is connected to.

Return: Multiple values. See function parameter descriptions for more info

Note that the full scheduler version is represented as MajorVersion.MinorVersion.BuildNumber, for example 4.3.288

Parameters:

CodeBase	A StringHolder class variable used as an output parameter. In case of successful method execution, the value of the **CodeBase** parameter is populated with the scheduler code type. For 24x7 Scheduler Multi-platform Edition, the returned value is "java" while for 24x7 Scheduler Windows Edition the returned value is "c++"
OSVersion	A StringHolder class variable used as an output parameter. In case of successful method execution, the value of the **OSVersion** parameter is populated with name, type, and version of the operation system running the scheduler.
MajorVersion	An IntegerHolder class variable used as an output parameter. In case of successful method execution, the value of the **MajorVersion** parameter is populated with major version number.
MinorVersion	An IntegerHolder class variable used as an output parameter. In case of successful method execution, the value of the **MinorVersion** parameter is populated with minor version number.
BuildNumber	An IntegerHolder class variable used as an output parameter. In case of successful method execution, the value of the **BuildNumber** parameter is populated with build number.

See also:

openSession

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
StringHolder codeBase, osVersion;
IntegerHolder major, minor, build;
Integer retCode = obj.getVersion( codeBase, osVersion,
                                  major, minor, build );
if (retCode == -1)
       System.out.println( obj.getLastError() ); // Print error
```

```
else
       // Print scheduler version
       System.out.println( "Version: "
              + major.getValue().toString()
              + "." + minor.getValue().toString()
              + "." + build.getValue().toString() );
```

holdJob

Integer holdJob (Long JobQueueID) throws RemoteException

The **holdJob** method places already queued job on hold. The job must be in awaiting start state. Do not confuse holdJob with disableJob. The latest method can be used to disable a job and prevent it from starting again, while the **holdJob** method can be used to lock an already queued job instance and prevent it from running until it is released. Note that this operation affects only one job instance. Other jobs can continue running in the queue as well as other instances of the same job can be also added to the queue and be run.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobQueueID	Run-time job number in queue.

See also:

deleteJob
disableJob
killJob
releaseJob

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.holdJob( 22465 );
if (retCode.intValue() != 1)
      /* Print error */
      System.out.println( obj.getLastError() );
```

killJob

Integer killJob (Long JobQueueID) throws RemoteException

The **killJob** method deletes job from the queue. If the job is already running it forcedly terminates the running process. If the job type is JavaScript the script and the job is run as a non-detached process, killJob will attempt to interrupt the script. If the job is SQL Script and the job is run as a non-detached process, killJob will attempt to execute database-specific "Cancel Query" request.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobQueueID	Run-time job number in queue.

See also:

deleteJob
disableJob
holdJob

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.killJob( 22465 );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

killShellCommand

Integer killShellCommand (Long ProcessID) throws RemoteException

The **killShellCommand** method forcedly terminates operation system processes. Although this method can be used with any processes it is recommended that this method is only used for processes started using runShellCommand.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

ProcessID	System process ID.

See also:

runShellCommand
getProcessList
killJob

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.killShellCommand( 254 );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

logMessage

Integer logMessage (Integer SourceID,
String SourceName,
Integer EventSeverity,
String Message,
Boolean ForceReportUpdate) throws RemoteException

The **logMessage** method adds new record to the 24x7 Scheduler event log. If parallel logging to the Windows NT event log is enabled, **logMessage** also writes an entry at the end of the Windows NT application event log.

Note: 24x7 Scheduler does not verify SourceID and SourceName values. Normally you should use them to specify job ID and job name that the logged message belongs to.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

SourceID	ID of the job for which you add the new message. To add a job independent message specify 0 for the **SourceID**.
SourceName	Descriptive name of the message source.
EventSeverity	The event severity. It can be one of the following: • RC_ERROR • RC_WARNING • RC_INFO For more information about event severity values see Constants topic.
Message	The message that you want to add to the 24x7 event log.
ForceReportUpdate	A Boolean whose value indicates whether 24x7 Scheduler must perform an immediate update of the **24x7 Status Report** or perform a deferred update. **Value** **Meaning** True update immediately False don't update immediately **Important Notes:** • If the **24x7 Status Report** is not enabled, **ForceReportUpdate** parameter is ignored. • "ForceReportUpdate = True" ensures the report is updated immediately so other people who monitor that report can see this change in a real-time. However, immediate updates have some impact on the performance of the 24x7 Scheduler.

See also:

getJobLog
getStatusReport

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.logMessage( "34", "Some job name",
             obj.RC_INFO, "Ready to run job #34",
             new Boolean(true) );
if (retCode.intValue() != 1)
       System.out.println( obj.getLastError() ); // Print error
```

openSession

Integer openSession (String ServerLocation,
Integer ServerPort,
String User,
String Password,
String Serial) throws RemoteException

The **openSession** method establishes new connection between **24x7 Remote Control Java** control and the specified target 24x7 Scheduler running in either Master or Agent mode. The target 24x7 Scheduler can run on the same computer or run on another networked computer.

Notes:

- You must open a session using the **openSession** method before you call other **24x7 Remote Control Java** methods.
- Use the **closeSession** method to close the session after you don't need it anymore.

If you destroy j24x7 class and a session is still open, the j24x7 class automatically calls **closeSession**. In web based applications object clean up and destruction is often left to the web server. In such applications you usually do not need to call **closeSession** explicitly.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

User	The user ID that you want to use for the connection.
Password	The password that you want to use for the connection.
`Serial`	`The Serial Number given for the` **`24x7 Remote control Java interface`**
ServerLocation	The location of the 24x7 Scheduler server. Specify either of the following values: • The IP address (for example, 199.99.99.91) • The host name of the remote computer (network computer name in workgroup) • LocalHost (this indicates that the 24x7 Scheduler resides on the local machine)
ServerPort	The port number for the 24x7 Scheduler server (for example, 1096).

See also:

closeSession
Using 24x7 Java API in Your Program

Example:

```
// create j24x7 class
j24x7 obj = new j24x7();
// open session
obj.openSession( "LocalHost", new Integer(1096), "john_doe",
              "password", "00000-00000-00000");

// ... do something here ...

// close session
obj.closeSession();
```

protectJob

Integer protectJob (String JobID,
String ProtectionType,
String JobPassword) throws RemoteException

The **protectJob** method can be used to change protection type for an already protected job or setup protection on an unprotected job. If the job is already protected you must specify valid JobPassword that matches current job password.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name
ProtectionType	Job protection code. The following values are supported: **F** – Full protection – job definition cannot be seen, changed or deleted and the job cannot be run manually. **R** – Read only protection – job definition can be seen and the job can be run manually, but the job cannot be changed or deleted. **E** – Execute only – job definition cannot be seen, changed or deleted, but the job can be run manually.
JobPassword	Job password

See also:

unprotectJobob
disableJob
createJob

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.protectJob( "34", "R", "some password" );
if (retCode.intValue() != 1)
      System.out.println( obj.getLastError() ); // Print error
```

releaseJob

Integer releaseJob (Long JobQueueID) throws RemoteException

The **releaseJob** method releases held jobs job and puts it back to queue according to job priority. The job state changes to "awaiting start". Do not confuse releaseJob with enableJob. The latest method can be used to enable previously disabled job and allow it to be scheduled again, while the **releaseJob** method can be used to unlock a queued job instance and allow that specific instance to run. Note that this affects only single job instance.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobQueueID	Run-time job number in queue.

See also:

holdJob
enableJob

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.releaseJob( 22465 );
if (retCode.intValue() != 1)
      /* Print error */
      System.out.println( obj.getLastError() );
```

queueJob

public Long queueJob(String jobID) throws J24x7EJBException

The **queueJob** method submits the specified job to the associated job queue. If the queue is free, the specified job starts running immediately; otherwise, it waits for the queue to become free. As compared to **runJob** method with synchronous parameter set, the submitter job does NOT wait for the submitted child job to complete.

Return: Returns unique run-time job id for the submitted job. This run-time id can be referenced in **killJob**, **holdJob** and other methods dealing with run-time job instances. Return -1, if an error occurres

Parameters:

jobID	Job ID converted to string or job name.

See also:

createJob
runJob

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.queueJob( "test job" );
if (retCode.intValue() == -1)
       System.out.println( obj.getLastError() ); // Print error
```

readFile

Integer readFile(String FileName,
StringHolder OutputBuffer) throws RemoteException

This method can be used to remotely read small text files and logs.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

FileName	A string whose value is the name of the target file.
Buffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the content of the file is copied to the **OutputBuffer**.

See also:

writeFile
runJob
runShellCommand
runScript

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// Read remote log file
StringHolder buffer = new StringHolder("");
Integer retCode = obj.readFile( "c:\\log\\somelog.log", buffer );
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
       // Print log content
       System.out.println( "Log content: " + buffer.getValue() );
```

runJob

Integer runJob (String JobID,
Boolean Detached) throws RemoteException

The **runJob** method starts the specified job. The job is executed on the computer where the target 24x7 Scheduler is running.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
Detached	Controls how the job is processed. **Value** **Meaning** True Start the job using a separate instance of 24x7 Scheduler and then immediately return control back to the caller False Start the job using the main instance of 24x7 Scheduler and wait for the job to complete before returning control back to the caller. **Note:** Each method has some advantages and disadvantages. If you want to wait for a job to complete before starting next job use **False** for the **Detached** property. This will ensure that jobs will run sequentially. Otherwise, if you run the job **Detached** (Detached = True) and then run another job, both of these jobs will run concurrently. On the other hand if you run a job non-**Detached** (Detached=False) and that job takes a long time to run your session may expire before the job completes and this may cause the job to abort prematurely. Always use **True** for the **Detached** property for jobs taking a long time to run.

See also:

createJob
queueJob
runScript
runShellCommand

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.runJob( "34", new Boolean(true) );
if (retCode.intValue() != 1)
       System.out.println( obj.getLastError() ); // Print error
```

runScript

Integer runScript (String Script) throws RemoteException

Important Notes: This method is only valid when executed in a client session connected to 24x7 Scheduler Windows Edition. The method is ignored if executing in a client session connected to 24x7 Scheduler Multi-platform Edition.

The **runScript** method executes the specified JAL script. The effect of this method is similar to running a job with the same script. The script is executed on the computer running 24x7 Scheduler to which the API client session is connected to..

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Script	JAL script that you want to execute.

See also:

runJob
runShellCommand
utilRunScript

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
// Run dynamic script to copy .LOG files on the server
String script =
    "Dim( count, number )\r\n" +
    "FileCopyEx( \"c:\\\\pathA\\\\*.log\", \"c:\\\\pathB\", count )";
Integer retCode = obj.runScript( script );
if (retCode.intValue() != 1)
      /* Print error */
      System.out.println( obj.getLastError() );
```

runShellCommand

Integer runShellCommand (String CommandLine,
String Dir,
Boolean Async,
Integer Timeout) throws RemoteException

The **runShellCommand** method executes the specified Operation System command or program. The command is executed on the computer where the target 24x7 Scheduler is running.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

CommandLine	The full or partial path and filename of the module to execute. The module may be a binary program or a command line script. **Notes:** • You should always include the full path - don't rely on the PATH environment variable, because this may be different at the time the program runs, depending on what account the program is being run under. Also, keep in minds that other programs can modify the PATH environment variable as well. • Program name can be followed by command line parameters.
Dir	The directory you want to set as a working directory for the started program. Most programs do not require this so that you can specify an empty string (""). Occasionally however, a program will refuse to run properly unless it is specifically told where to find other program files. The 24x7 Scheduler will change to this directory when running the program or document.
Async	This controls how the 24x7 Scheduler executes the process specified in the **CommandLine**. **Value** — **Meaning** True — Start the specified command and return control back to the 24x7 Scheduler. The **Timeout** argument is ignored in this case. False — Start the specified command and wait for that command to complete before returning control back to the 24x7 Scheduler. **Note:** When running the command synchronously (**Async=False**) 24x7 Scheduler enters an efficient wait state until this command finishes or the **Timeout** interval elapses. In the latter case, the 24x7 Scheduler forcedly terminates the process created by the specified **CommandLine**.
Timeout	The maximum time interval (in seconds) within which you allow the

	process (as specified in the **CommandLine**) to run. Use 0 **Timeout** to allow infinite waiting. The **Timeout** parameter is ignored when **Async** parameter is set to **True**.

See also:

runJob
runScript
utilRunScript

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.runShellCommand(
                    "h:\\sales\\reports\\month_end.exe",
                    "", new Boolean(false), new Integer(30) );
if (retCode.intValue() != 1)
       System.out.println( obj.getLastError() ); // Print error
else
       // Reports ran successfully, now redirect to the monthly
       // report menu
       document.location = "../reports/monthly/menu.jsp";
```

setFolderProperty

Integer setFolderProperty (String FolderID,
String PropertyName,
String NewValue) throws RemoteException

The **setFolderProperty** method changes value of the specified folder **PropertyName** for the specified **FolderID**. The **PropertyName** must be a valid JDL job property name. The following folder properties are currently supported: "FOLDER_NAME", "DESCRIPTION".

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

FolderID	Folder ID converted to string or folder name.
PropertyName	The folder property name.
NewValue	Then new property value.

See also:

createFolder
getFolderProperty

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// Change description of folder "Rep12"
Integer retCode = obj.setFolderPropery( "Rep12", "DESCRIPTION",
             "This folder is now used for ad-hoc reports" );

if (retCode.intValue() != 1)
       System.out.println( obj.getLastError() ); // Print error
```

setGlobalVariable

Integer setGlobalVariable (String VariableName,
String NewValue) throws RemoteException

The **setGlobalVariable** method changes the value of the global variable on the target 24x7 Remote Agent or 24x7 Master Scheduler.

Important Note: This method is only valid for 24x7 Scheduler Windows Edition.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

VariableName	The name of the global variable that must exist on the target 24x7 Agent or Master Scheduler. Variable names are case-insensitive.
NewValue	The new value converted to string format.

See also:

getGlobalVariable
runScript

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.setGlobalVariable( "my_variable", "125" );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

setJobProperty

Integer setJobProperty (String JobID,
String PropertyName,
String NewValue) throws RemoteException

The **setJobProperty** method changes value of the specified job **PropertyName** for the specified **JobID**. The **PropertyName** must be a valid JDL job property name. For more details, see "Job Property Names" topic in the 24x7 on-line help file.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
PropertyName	The job property name. For supported job properties and their JDL names see 24x7 User's Guide. As a quick reference, see Job Properties in JDL Format topic in this manual.
NewValue	Then new value converted to string format.

See also:

getJobProperty
setJobPropertyEx
updateJob
setFolderProperty
setJobTemplateData

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
// Change value of QUEUE property for job #235
retCode = obj.setJobProperty( "235", "QUEUE", "Reports" );
if (retCode.intValue() != 1)
       System.out.println( obj.getLastError() );  // Print error
```

setJobPropertyEx

Integer setJobProperty (String JobID,
String PropertyNameList,
String NewValueList) throws RemoteException

The **setJobPropertyEx** method changes values of job properties specified in the **PropertyNameList** parameter for the job specified by **JobID** parameter. The **PropertyNameList** must contain comma-separated list of valid JDL job property names. For more details, see "Job Property Names" topic in the 24x7 on-line help file.

Note: setJobPropertyEx method is an extended version of **setJobProperty** method. **setJobPropertyEx** is capable of updating multiple properties at once while **setJobProperty** can update only one property at a time. **setJobPropertyEx** provides more efficient and faster method for updating multiple properties as it makes just one round-trip to the 24x7 Scheduler server. On the other hand, because new values are passed as a comma-separated list, **setJobPropertyEx** cannot be used to update values that contain commas

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
PropertyNameList	Comma-separated list of job property names. For supported job properties and their JDL names see 24x7 User's Guide. As a quick reference, see Job Properties in JDL Format topic in this manual.
NewValueList	Tab-separated list of new property values converted to string format.

See also:

setJobProperty
getJobPropertyEx
updateJob
setFolderProperty
setJobTemplateData

Example:

```
      // ... create j24x7 class (obj) and open session here ...
      // ... For more details, see openSession topic ...

      // Change job #235 schedule to daily job to be run on Mondays
      // and Wednesdays (SCHEDULE_TYPE=D, MONDAY=Y, WEDNESDAY=Y, all other
      // days set to N)
      Integer retCode = obj.setJobPropertyEx( "235",
"SCHEDULE_TYPE,SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY",
"D,N,Y,N,Y,N,N,N" );
```

```
if (retCode.intValue() != 1)
       System.out.println( obj.getLastError() );  // Print error
```

setJobTemplateData

Integer setJobTemplateData (String JobID,
String TemplateName,
String TemplateFile,
String Data) throws RemoteException

The **setJobTemplateData** method saves template properties and data used to create or modify the specified job. This data can be later retrieved using the **getJobTemplateData** method. If the method succeeds the values of **TemplateName, TemplateFile**, and **Data** parameters are saved in a file. The **Data** value is a free text that could be of virtually any size. It is up to the developer to decide which internal format to use for the **Data**.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
TemplateName	Name of the template used for the specified job.
TemplateFile	Name of the template file used for the specified job.
Data	Data that you want to save for future references.

See also:

getJobTemplateData
setJobProperty
setTemplate

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

String templateName = "Venus FTP Download";
String templateFile = "venus_ftp.ini";
String templateData = "SRC_DIR=\\daily\\cars\r\n" +
                      "SRC_FILE=122003car.zip\r\n" +
                      "DEST_DIR=f:\\carinfo";
Integer retCode = obj.setJobTemplateData( "235", templateName,
              templateFile, templateData  );
if (retCode.intValue() != 1)
       // Print error message
       System.out.println( "setJobTemplateData error: " +
                            obj.getLastError() );
```

setTemplate

Integer setTemplate (String TemplateFile,
String Buffer) throws RemoteException

The **setTemplate** method updates contents of the specified template file.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

TemplateFile	Template file name.
Buffer	A String variable whose value replaces contents of the specified template file.

See also:

addTemplate
getTemplate
getJobTemplateData
getTemplateCatalog
setJobTemplateData

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// Obtain template code from the submitted form
String templateCode = request.getParameter("TEMPLATE_CODE");
// Update ftp_reports.ini template
Integer retCode = obj.setTemplate(
                            "%HOME%\\Templates\\ftp_reports.ini",
                             templateCode );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

test

Integer test (Integer x) throws RemoteException

The **test** method is provided exclusively for your convenience so you can test whether you can load and call **24x7 Java API** methods. The **test** method does not communicate with the 24x7 servers and performs simple computations on the client side only.

Return: The returned value must be the same as the **x** value specified for the method argument.

Example:

```
// Create j24x7 class
j24x7 obj = new j24x7( );

// test it
if (obj.test(5) != 5)
       System.out.println("Error") // Print error
```

unprotectJob

Integer unprotectJobob (String JobID,
String JobPassword) throws RemoteException

The **unprotectJobob** method can be used to remove protection from a protected job. You must specify valid JobPassword that matches the current job password.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name
JobPassword	Job password

See also:

protectJob
enableJob
createJob

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.unprotectJob( "34", "some password" );
if (retCode.intValue() != 1)
       System.out.println( obj.getLastError() ); // Print error
```

updateAgentProfile

Integer updateAgentProfile(String ProfileName,
String ComMethod,
String Location,
String Port,
String Options) throws RemoteException

The **updateAgentProfile** method updates properties of an existing 24x7 Remote Agent or 24x7 Master Scheduler profile.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

For description of **updateAgentProfile** parameters see addAgentProfile method.

See also:

addAgentProfile
deleteAgentProfile
getAgentProfile

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.updateAgentProfile( "Print Server", "Winsock",
                                          "192.168.100.1", "1096", "");
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

updateCalendar

Integer updateCalendar (Integer CalendarID,
String Name,
Boolean Excelusive) throws RemoteException

The **updateCalendar** method updates properties of an existing business calendar.

Important Note: This method is only valid for 24x7 Scheduler Multi-platform Edition.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

For description of **updateCalendar** parameters see addCalendar method.

See also:

addCalendar
getCalendarList
deleteCalendar

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// update name for calendar #25
Integer retCode = obj.updateDatabaseProfile( 25,
        "Bi-monthly Calendar", new Boolean(false) );
if (retCode.intValue() != 1)
        /* Print error */
        System.out.println( obj.getLastError() );
```

updateDatabaseProfile

Integer updateDatabaseProfile (String ProfileName,
String DatabaseDriver,
String Server,
String DatabaseName,
Boolean AutoCommit,
String User,
String Password) throws RemoteException

The **updateDatabaseProfile** method updates properties of an existing database profile.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

For description of **updateDatabaseProfile** parameters see addDatabaseProfile method.

See also:

addDatabaseProfile
getDatabaseProfile
deleteDatabaseProfile

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

Integer retCode = obj.updateDatabaseProfile( "Prod Server",
                  "SQL Server", "Neptune",
                  "warehouse", True, "sa", "******");
if (retCode.intValue() != 1)
        /* Print error */
        System.out.println( obj.getLastError() );
```

updateJob

Integer updateJob (String JobID, String JobDefinition) throws RemoteException

The **UpdatesJob** method updates properties of an existing job.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Note: **JobDefinition** parameter may contain any combination of job properties and scripts. Only values of referenced properties are updated. Values of all other properties are preserved as is.

Parameters:

JobID	Job ID converted to string or job name
JobDefinition	The Job definition in JDL format. For a wide variety of examples see job templates available in the [24x7 install directory]\Template subdirectory. The default path is *C:\Program Files\24x7 Automation 3\Template*. For supported job properties and JDL properties see 24x7 User's Guide.

See also:

createJob
setJobProperty
setJobTemplateData
changeFolder
disableJob
protectJob
runJob

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// Get job definition from the submitted form
String jobDefinition = request.getParameter("JOB_DEF");
// Update job #234
Integer retCode = obj.updateJob( "234", jobDefinition );
if (retCode < 0)
       // Print error
       System.out.println( obj.getLastError() );
else
       // Print confirmation
       System.out.println("Job updated succesfully");
```

updateJobQueue

Integer updateJobQueue (String QueueName,
Integer MaxSize) throws RemoteException

The **updateJobQueue** method updates properties of an existing job queue.

Note: This method has been deprecated. Use updateJobQueueEx method.

Note: In 24x7 Scheduler Windows Edition, updating job queue properties does not have an immediate effect. The changes are effective only after the 24x7 Scheduler is restarted. In 24x7 Scheduler Multi-platform Edition, the changes are effective immediately.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

QueueName	Name of the Job Queue whose properties will be modified.
MaxSize	Maximum queue size in Mbytes. 1 Mbytes should be sufficient in most cases.

See also:

addJobQueueEx
updateJobQueueEx
deleteJobQueue

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.updateJobQueue( "Payroll jobs",
                                      new Integer(5));
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

updateJobQueueEx

Integer updateJobQueue (String QueueName,
Integer MaxSize,
Integer MaxJobs,
Boolean NearCapacityAlerts,
String EmailSender,
String EmailPassword,
String EmailRecipients) throws RemoteException

The **updateJobQueue** method updates properties of an existing job queue.

Note: In 24x7 Scheduler Windows Edition, updating job queue properties does not have an immediate effect. The changes are effective only after the 24x7 Scheduler is restarted. In 24x7 Scheduler Multi-platform Edition, the changes are effective immediately.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

QueueName	Name of the Job Queue whose properties will be modified.
MaxSize	Maximum allowed disk space usage in Mbytes. 1 Mbytes should be sufficient in most cases. As of 24x7 Scheduler Multi-platform Edition v4.3, this queue parameter is reserved for future use. The parameter is used in 24x7 Scheduler Windows Edition only. Specify zero value for unlimited disk space usage.
MaxJobs	Maximum number of jobs allowed in a queue at any moment in time. The count includes both running and queued jobs, including jobs placed on hold. Specify zero value for unlimited number of jobs.
NearCapacityAlerts	Enables sending email alerts in case a queue is at or near its maximum capacity. The capacity is controlled by **MaxSize** and **MaxJobs** parameters
EmailSender	Email account (for MAPI email protocol) or email address (for SMTP email protocol) to use for sending "near capacity" and "over capacity" email alerts.
EmailPassword	Email password to use for authenticating email sender to the email server. Specify an empty string or null value if password is not required.
EmailRecipients	Comma-separated list of email recipient addresses for "near capacity" and "over capacity" email alerts.
QueueName	Name of the new Job Queue. This name should be unique.
MaxSize	Maximum queue size in Mbytes. 1 Mbytes should be sufficient in most

	cases.

See also:

AddJobQueueEx
deleteJobQueue

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
Integer retCode = obj.updateJobQueue(( "Payroll jobs",
       new Integer(1), new Integer(200), new Boolean(true)
       "alerter@domain.com", null, "helpdesk@domain.com" );
if (retCode.intValue() != 1)
       /* Print error */
       System.out.println( obj.getLastError() );
```

utilRunScript

Integer utilRunScript (String Script,
StringHolder OutputBuffer) throws RemoteException

Important Note: This method is only valid for 24x7 Scheduler Windows Edition.

The **utilRunScript** method executes the specified JAL script and returns the script output. The script is executed on the computer running 24x7 Scheduler to which the API client session is connected to.
This method internally calls the runScript method. Before calling **runScript** method to execute the **Script**, **utilRunScript** creates a unique temporary file on the remote system running 24x7 Scheduler, and alters text of the **Script,** adding an additional variable named **output**. This variable value is set to the name the temporary file. This is how the name of the temporary file is passed to the **Script**. In the **Script** you can then use this file to save any script output. After completion of the **runScript** method, **utilRunScript** reads contents of the remote temporary file into the **OutputBuffer** parameter and deletes the file. That's how the output saved on the remote system is returned to the caller.

Note: Do not use **utilRunScript** if you do not need to return any output from the script. Use runScript method directly.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

Script	JAL script that you want to execute.
OutputBuffer	A StringHolder class variable used as an output parameter. In case of successful method execution, the **Script** output is copied to the **OutputBuffer**.

See also:

runJob
runShellCommand
runScript

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...
// Run dynamic script to copy .LOG files on the server
StringHolder buffer = new StringHolder("");
String script = "Dim( count, number )\r\n" +
"FileCopyEx( \"c:\\\\pathA\\\\*.log\", \"c:\\\\pathB\", count )\r\n"+
"FileSave( output, count )";
Integer retCode = obj.utilRunScript( script, buffer );
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
       // Print results
```

```
            System.out.println("Total files copied: " +
                               buffer.getValue());
```

writeFile

Integer writeFile(String FileName,
String Buffer) throws RemoteException

This method can be used to remotely create and update small batch and configuration files which can later be scheduled using createJob and similar methods.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the getLastError method to obtain the error message.

Parameters:

FileName	A string whose value is the name of the target file.
Buffer	A string whose value is the text to write to the target file.

See also:

createJob
runShellCommand
runScript
readFile

Example:

```
// ... create j24x7 class (obj) and open session here ...
// ... For more details, see openSession topic ...

// Create batch job on the server to count all .CSV files
String batchSccript = "dir /S /B c:\\data\\*.csv > files.log\r\n" +
       "c:\\bin\\linecount files.log | c:\\bin\\tail –n 2 | " +
       "c:\\bin\\cut –f2 \":\" > filecount.txt";
Integer retCode = obj.writeFile( "c:\\jobs\\csv_count.bat", script );
if (retCode.intValue() != 1)
       // Print error
       System.out.println( obj.getLastError() );
else
       // Print results
       System.out.println( "File c:\\jobs\\csv_count.bat has been " +
                           "written to the remote system" );
```

Additional Classes

The *com.sybase.dpb* package contains additional Java classes required to implement applications that use the **24x7 Remote Control Java** interface.

Class index

BooleanHolder -- Use this class for passing Boolean values for output parameters.

IntegerHolder -- Use this class for passing Integer values for output parameters.

StringHolder -- Use this class for passing String values for output parameters.

Exception index

RemoteException exception

BooleanHolder Class

Description

Holder class to contain the Boolean class. Use this class to pass Boolean values by reference. This class is part of com.sybase.dpb package.

Specification

public class BooleanHolder extends Holder

Dependencies

java.lang.Boolean

Constructors

public BooleanHolder(Boolean f) -- Creates an instance containing the specified Boolean value.

Methods

public void setValue(Boolean f) -- Sets the boolean instance contained in the holder class.

Inherited methods

public Object getValue -- Retrieves the instance contained in the holder class.

IntegerHolder Class

Description

Holder class to contain the Integer class. Use this class to pass Integer values by reference. This class is part of com.sybase.dpb package.

Specification

public class IntegerHolder extends Holder

Dependencies

java.lang.Integer

Constructors

public IntegerHolder(Integer f) -- Creates an instance containing the specified Integer instance.

Methods

public void setValue(Integer f) -- Sets the Integer instance contained in the holder class.

Inherited methods

public Object getValue -- Retrieves the instance contained in the holder class.

StringHolder Class

Description

Holder class to contain the String class. Use this class to pass String values by reference. This class is part of com.sybase.dpb package.

Specification

public class StringHolder extends Holder

Dependencies

java.lang.String

Constructors

public StringHolder(String f) -- Creates an instance containing the specified String instance.

Methods

public void setValue(String f) -- Sets the String instance contained in the holder class.

Inherited methods

public Object getValue -- Retrieves the instance contained in the holder class.

RemoteException Class

Description

All 24x7 Java API methods throw the RemoteException exception. This class is part of com.sybase.dpb package.

Specification

public class RemoteException extends java.lang.Exception

Dependencies

java.lang.Exception

Constructors

public RemoteException( ) -- Creates an exception.
public RemoteException(String s) -- Creates an exception using the specified message text. Call the ancestor **getMessage** method to retrieve the message text.

Job Properties in JDL Format

All job properties are documented in the 24x7 Scheduler User's Guide. This topic can be used as a quick reference for supported job properties and their JDL names.

Job Definition Language (JDL) supports the following property names:

Property Name	Meaning
ACCOUNT	E-mail Account such as user ID, profile, or e-mail address (e-mail watch job). The actual value may differ for different e-mail interfaces. For a MAPI interface you should use the name of the MAPI profile you use when logging on to the e-mail system. For Lotus Notes you should use the name of the user (or ID) you use when logging on to the Lotus Notes. For SMTP you should use your e-mail address.
AGENT	Same as Host (see Host description)
ALL_DAY_TYPE	All Day Schedule Type, one of the following: **R**, **L** (**R** - recursive, repeat at specified intervals; **L** - fixed time list)
ARCHIVE_OUTPUT	Capture and archive job output, one of the following: **Y**, **N** (yes, no) **Note**: This property is not used in 24x7 Scheduler Windows-only Edition.
AS_GROUP	Name of Auto-scaling group assigned to the job
ASYNC	Asynchronous Process, one of the following: **Y**, **N** (yes, no)
BACKUP_AGENT	Same as Backup Host (see Backup Host description)
BACKUP_HOST	Backup Remote Host (e.g. Backup Remote Agent name)
CALENDAR	Name of the Calendar object assigned to the job
COMMAND	Program Command Line
DAY_END_TIME	Daily End Time for "all day" jobs with limited run-time interval
DAY_LIST	Monthly Schedule List of fixed Day Numbers, numbers must be in 1..31 range. Example: 1,3,5,7,14. This property is shared with TIME_LIST property for All Day Schedule.
DAY_NAME	Monthly Schedule Day Name, one of the following: **Monday , Tuesday, Wednesday , Thursday , Friday , Saturday , Sunday , Weekend, Weekday**
DAY_NUMBER	Monthly Schedule Day Number, a number from 1 - 31 range
DAY_START_TIME	Daily Start Time for "all day" jobs with limited run-time interval
DELAY	Allowed Job Delay Interval (minutes)
DELETE_RULE	Delete, Move, and Rename Semaphore File Rules, one of the following: D, **A, B, M, E, R, C, F, G** (**D** - do not delete, move, rename; **A** - delete after job run; **B** - delete before job run; **M** - move before job run; **E** - move after job run; **R** - rename before job run; **C** - rename after job run; **F** -move and rename before job run; **G** - move and rename after job run)
DESCRIPTION	Job Description
DISABLE_ON_ERROR	Disable Job on Error, one of the following: **Y**, **N** (yes, no)
DISABLED	Job Disabled Status, one of the following: **Y**, **N** (yes, no)

DETACHED	Detached Job, one of the following: **Y**, **N** (yes, no) **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
END_DATE	Last Job Start Date
END_TIME	Last Job Start Date
EXIT_CODE	Exit Code Condition (as a string expression)
FILE	Semaphore File Names(s) for file-watch jobs; Module Name for process-watch job
FOLDER	Job Folder ID. This is read-only property. It may not be changed using **SET** command. It can be retrieved using **GET** command
FOLDER_NAME	Job Folder Name. This is read-only property. It may not be changed using **SET** command. It can be retrieved using **GET** command
FRIDAY	Execute Job On Fridays, one of the following: **Y**, **N** (yes, no)
HOST	Remote Host (Remote Agent Name)
ID	Job ID, This is read-only property. It may not be changed using **SET** command. It can be retrieved using **GET** command with Job Name parameter.
IGNORE_ERRORS	Ignore Errors, one of the following: **Y**, **N** (yes, no)
INIT_TIMEOUT	Initial Timeout before sending keystroke (seconds)
INTERVAL	Repeat Interval for Job having Schedule Type **T**
JOB_PASSWORD	Job Protection State and Password. Sets or removes job protection state and password. This is a write-only property. It can be changed using **SET** command, but it cannot be retrieved using **GET** command. The value in this property must be specified in the following format : [old password][tab character][new password][tab character][protection state] If the job is not protected, the [old password] is ignored, otherwise a valid password must be specified in order to remove or change job password or protection state. If the protection exists and the new protection state is specified as an empty string the protection will be removed. The protection code must one of the following: **F**, **E**, **R**, an empty string (**F** -full protection; **E** - execute only; **R** - read only; an empty string indicates that a job is not protected).
JOB_TYPE	Job Type, one of the following: **P**, **D**, **S** (program, database, script)
KEYSTROKE	Keystroke to send to the launched program. **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
LOG	Log Job Execution, one of the following: **Y**, **N** (yes, no)
MESSAGE	E-mail Message Text (e-mail watch job). **Note**: This is used in email-watch jobs only. Do not confuse this with Message triggers. The legacy property named Message is preserved for backward compatibility reasons.
MESSAGE_ACTION_TYPE	Type of Action that the job should perform after being triggered by a Message, one of the following **D**, **B**, **A** (do nothing, delete message before starting job execution,

	delete message after job executions completes.
MESSAGE_ID	Message ID of the Message to monitor. This is used in Message-watch job triggers. Do not confuse this with email messages.
MODIFY_TERMINAL	Network name of the computer from which the job was last modified. This is a read-only property. It cannot be changed using **SET** command. It can be retrieved using **GET** command. **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
MODIFY_TIME	Date and time when the job was modified. This is a read-only property. It cannot be changed using **SET** command. It can be retrieved using **GET** command. **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
MODIFY_USER	Name of the user who last modified the job. This is a read-only property. It cannot be changed using **SET** command. It can be retrieved using **GET** command. **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition
MONDAY	Execute Job On Mondays, one of the following: **Y**, **N** (yes, no)
MONTHLY_TYPE	Monthly Schedule Type, one of the following: **D**, **T**, **L** (**D** - by day number; **T** - by day name; **L** - fixed day list)
MOVE_DIR	Name of the destination directory for semaphore file move and rename operations.
MSG_ACCOUNT	E-mail Account for Notification Action of E-mail Type E-mail (user ID, profile, or e-mail address). The actual value may differ for different e-mail interfaces. For the MAPI interface you should use the name of the MAPI profile you use when logging on to the e-mail system. For Lotus Notes you should use the name of the user (or ID) you use when logging on to Lotus Notes. For SMTP you should use your e-mail address.
MSG_ACTIONS	Map of Notification Actions and Events in text format. The map is represented as a comma-separated list of 2-character values where in every list item the first character represents Notification Event Type and the second character represents Notification Action Type. The following characters can be used for the event type: **S** - job start, **F** - job finish, **E** - job error, **N** - job file not found, **L** - job is late. The following characters can be used for the action type: **E** - send email, **P** - send page, **N** - send network popup message, **D** - execute database commands, **F** - create semaphore files, **M** – send Message; **T** - send SNMP trap, **J** - run job, **R** - run program, **S** -run script. Example map: SE,FE,EE,FD This example map represents the following Notification Events and Events: 1. Send notification email on job start. 2. Send notification email on job finish. 3. Send notification email on job error.

	4. Execute database commands on job finish.
MSG_DATABASE	Execute Notification Action of Database Type, one of the following: **Y**, **N** (yes, no)
MSG_EMAIL	Execute Notification Action of E-mail Type, one of the following: **Y**, **N** (yes, no)
MSG_ERROR	Execute Notification Action on Job Execution Error, one of the following: **Y**, **N** (yes, no)
MSG_FILE	Execute Notification Action of Semaphore File Type, one of the following: **Y**, **N** (yes, no)
MSG_FILE_NAME	File name(s) for Notification Action of Semaphore File Type
MSG_FINISH	Execute Notification Action on Job Finish, one of the following: **Y**, **N** (yes, no)
MSG_JOB	Execute Notification Action of Run Job Type, one of the following: **Y**, **N** (yes, no)
MSG_JOB_ID	Job name or job id for Notification Action of Run Job Type
MSG_MESSAGE	Execute Notification Action of Message Type, one of the following: **Y**, **N** (yes, no)
MSG_MESSAGE_DATA	The data to send in Notification Action of Message Type. The data format is user defined. The size of the data shouldn't exceed 1 MByte
MSG_MESSAGE_ ID	Message ID of the Message sent in Notification Action of Message Type
MSG_LATE	Execute Notification Action on Job Late Start, one of the following: **Y**, **N** (yes, no)
MSG_NET	Execute Notification Action of Network Message Type, one of the following: **Y**, **N** (yes, no) **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
MSG_NET_RECIPIENT	Message Recipient for Notification Action of Network Message Type **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
MSG_NOTFOUND	Execute Notification Action on Job Executable Not Found Error, one of the following: **Y**, **N** (yes, no)
MSG_PAGE	Execute Notification Action of Page Type, one of the following: **Y**, **N** (yes, no) **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
MSG_PAGER	Page Recipient's Pager Number **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
MSG_PASSWORD	E-mail Password for Notification Action of E-mail Type
MSG_PROFILE	Database Profile for Notification Action of Database Type
MSG_PROGRAM	Execute Notification Action of Run Program Type, one of the following: **Y**, **N** (yes, no)
MSG_PROGRAM_NAME	Program name for Notification Action of Run Program Type
MSG_PROGRAM_TIMEOUT	Process Timeout (seconds) for Notification Action of Run Program Type

MSG_RECIPIENT	E-mail Recipient for Notification Action of E-mail Type
MSG_SCRIPT	Execute Notification Action of Run Script Type, one of the following: **Y**, **N** (yes, no)
MSG_SCRIPT_CODE	JavaScript script for Notification Action of Script Type
MSG_SCRIPT_ TYPE	Type of Script for Notification Action of Script Type, Only **JS** (JavaScript) is supported in 24x7 Scheduler Multi-platform Edition for notification actions.
MSG_START	Execute Notification Action on Job Start, one of the following: **Y**, **N** (yes, no)
MSG_TRAP	Execute Notification Action of Send SNMP Trap Type, one of the following: **Y**, **N** (yes, no) NAME Job Name
MSG_SQL	SQL script for Notification Action of Database Type
MQ_PROFILE	MQ broker profile
MQ_SAVE_DIR	The output directory in which to save messages downloaded from MQ broker
MULTI_INSTANCE_CONTROL	Job queuing rule for handling multiple job instances, one of the following: **R** – always queue and run, **T** – terminate already running and queued job instances if any, and add new instance, **S** – if there are already running or queued job instances, skip new instance and do nothing, **E** – if there are already running or queued job instances, skip new instance and raise an error.
NAME	Job Name
OUTPUT_PATTERN	Job Output evaluation pattern This property is not used in 24x7 Scheduler Windows-only Edition.
PASSWORD	E-mail Password (e-mail watch job)
POLLING_INTERVAL	Polling Interval (minutes)
PRIORITY	Job Priority, one of the following: **-1** – low, **0** – normal, **1** – high
PROFILE	Database Profile
PROTECTION	Job Protection State, one of the following: **F**, **E**, **R**, an empty string (**F** – full protection; **E** – execute only; **R** – read only; an empty string indicates that a job is not protected). This is a read-only property. It cannot be changed using **SET** command. It can be retrieved using **GET** command.
QUEUE	Job Queue
REBOOT	Reboot Computer After Job Finished, one of the following: **Y**, **N** (yes, no) **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
RETRY_INTERVAL	Retry Interval (seconds)
RENAME_SUFFIX	Retry Interval (seconds)
RETRY_ON_ERROR	The name suffix used in semaphore file names for rename operations.
RUNAS_DOMAIN	Domain Name for authentication of jobs to be run using another user account. **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition. In that version the RUNAS_USER must contain both the domain and user names in

	domain\user format.
RUNAS_PASSWORD	Password for authentication of jobs to be run using another user account.
RUNAS_USER	User Name for authentication of jobs to be run using another user account.
SATURDAY	Execute Job On Saturdays, one of the following: **Y**, **N** (yes, no)
SAVE_ATTACHMENT	Save E-mail Attachments (e-mail watch job), one of the following: **Y**, **N** (yes, no)
SCHEDULE_TYPE	Schedule Type, one of the following: **O**, **D**, **T**, **M**, **F**, **P**, **A**, **E**, **I**, **L**, **S** (Time trigger: **O** – run once, **D** – repeat daily, **T** – repeat at specified time interval, **M** – repeat monthly; File trigger: **F** – check semaphore files; Process trigger: **P** – check process presence, **A** – check process absence; **E**-mail trigger: **E** - check e-mail message; User trigger: **I** – wake up on "user idle" event, **L**- wake up on log-off event, **S** – wake up on shutdown event)
SCRIPT	Job Script
SCRIPT_TYPE	Job Script Type, one of the following: **JAL**, **VBS**, **JS** (Job Automation Language, Visual Basic Script, JavaScript) **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
SEND_KEYSTROKE	Send Keystroke, one of the following: **Y**, **N** (yes, no) **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
SIZE_CHECK_INTERVAL	File Size Stability Check Interval (used in File-watch jobs)
SKIP	Skip Late Job, one of the following: **Y**, **N** (yes, no)
SKIP_HOLIDAY	Skip Job on Holiday, one of the following: **Y**, **N** (yes, no)
SLIDE_HOLIDAY	Slide Job Execution Time on the next non-holiday if it falls on holiday, one of the following: **Y**, **N** (yes, no)
SQL	SQL Command(s)
START_DATE	First Start Date
START_IN	Program Start-up Directory
START_TIME	First Start Time
SUBJECT	E-mail Message Subject for (e-mail watch job)
SUNDAY	Execute Job On Sundays, one of the following: **Y**, **N** (yes, no)
TIME_LIST	All Day Schedule List of fixed Times, values must be in valid 24-hour time format. Example: 11:10,12:10,17:10,18:10. This property is shared with DAY_LIST property for Monthly Schedule.
THURSDAY	Execute Job On Thursdays, one of the following: **Y**, **N** (yes, no)
TIMEOUT	Timeout (seconds)
TUESDAY	Execute Job On Tuesdays, one of the following: **Y**, **N** (yes, no)
WEDNESDAY	Execute Job On Tuesdays, one of the following: **Y**, **N** (yes, no)
WINDOW	Window Style, one of the following: **N**, **M**, **I**, **H** (normal, maximized, iconic, hidden) **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.

Installation

Minimal System Requirements

1. 24x7 Scheduler Multi-platform Edition v7.0 or later
2. 1.8 Mbytes free disk space
3. JDK 1.8 or later

Installation Steps

Run 24x7 Scheduler installation program. Follow instructions provided by the installation program.

Post-installation Steps

In order to start using 24x7 Java API you must add *jdpb24x7.jar* and *jdl3.1.0.jar* file names to the ClassPath option. You must also add import statements for *com.sybase.dpb* and *com.softtreetech.scheduler* as needed in your Java classes.

```
import com.sybase.dpb.*;
import com.softtreetech.scheduler.*;
```

Security Issues

To secure access to the remote 24x7 Scheduler servers enable security options on these servers. For more information please see "Security" topic in the 24x7 Automation Suite User's Guide and on-line help files.

Licensing

1. **Single installation license:** A separate single installation license is required for every 24x7 Remote Control Java installation. That's it, you need at least 2 single licenses (or a site license) for using 24x7 Remote Control Java to control and manipulate 24x7 Scheduler or 24x7 Remote Agent. In case of 2 single licenses, one license is applied to the 24x7 Scheduler server or 24x7 Remote Agent and another license is applied to the 24x7 Remote Control Java.
2. **Site license:** 24x7 Scheduler site license covers unlimited usage of 24x7 Remote Control Java. Site license users can use 24x7 Remote Control Java just as they use other components of the 24x7 Automation Suite. The usage of 24x7 Scheduler, 24x7 Remote Agents, 24x7 Remote Control, 24x7 Remote Control Java and 24x7 Remote Control COM is governed by their site license agreement. The site license also allows installing and using 24x7 Remote Control Java on **one** Web server. You may not install or run it on multiple servers using the same license.

3. **Web server usage:** You must obtain 24x7 Scheduler site license before you can use 24x7 Remote Control Java on your server. A separate site license is required for every web server utilizing 24x7 Remote Control Java.
4. **Redistribution:** You must obtain 24x7 Remote Control Java redistribution license before you can redistribute it with your program.

Important note about third party packages provided with the 24x7 Java API:
24x7 Java API is license includes installation and usage for *com.sybase.dpb* and *com.objectspace.jdl* packages. You may not use or redistribute these packages separately from 24x7 Java API.

The LICENSE.TXT file can be found in the 24x7 Scheduler installation directory. If you would like to obtain the redistribution license, please email your request to sales@softtreetech.com.