24x7 Scheduler™ 7.0 Multi-platform Edition

24x7 Enterprise Java Bean Reference

Table of Contents

About This Guide

This reference describes 24x7 Scheduler Enterprise Java Bean component and interface supported in 24x7 Scheduler Multi-platform Edition, an advanced cross-platform job scheduling and automation system. Information in this reference applies to the 24x7 Scheduler v5.1 running on all supported operating systems. This reference contains information for experienced users of the 24x7 Scheduler and assumes that you have a working knowledge of Java and EJB technologies and also understand basic concepts of your operation system.

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 FTP, HTTP, RAS, SQL, DBMS, SSH and other.

Trademarks

24x7 Automation Suite, 24x7 Scheduler, 24x7 Event Server, 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, Java 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.

24x7 Scheduler Enterprise Java Bean

Deployment

24x7 Scheduler Enterprise Java Bean has been developed to provide easy access to 24x7 Scheduler Multi-platform Edition from enterprise applications using Sun EJB standards. The EJB bean is provided ready-for-deployment to the following application servers:

- JBoss AS
- BEA Weblogic
- IBM WebSphere

The package for deployment can be found in the <installation dir>/ejb/ directory and named ejb24x7.jar. It has the following structure:

- com/softtreetech/jscheduler/ejb/* - compiled java classes for the Session EJB
- cscheduler/jdpb24x7.jar, cscheduler/jgl3.1.0.jar – java libraries for connecting to 24x7 Scheduler Multi-platform Edition
- META-INF/ejb-jar.xml – common standard deployment descriptor for all the application servers
- META-INF/jboss.xml, META-INF/ejb-borland.xml, META-INF/sun-ejb-jar.xml, META-INF/weblogic-ejb-jar – deployment descriptors specific for different application servers.

Minimal System Requirements

1. 24x7 Scheduler Multi-platform Edition v7.0 or later
2. 1.2 Mbytes free disk space
3. JDK 1.8 or later

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.

Interface J24x7EJB

24x7 Scheduler is a Stateful Session EJB bean. Its interface named 'J24x7EJB' provides all the functionality needed to work with a remote 24x7 Scheduler. The usual scenario can look like the following:

- Check that 24x7 Scheduler, you are connecting to, is started and has remote control enabled in its network options.
- Startup application server.
- Deploy ejb24x7.jar and check that it was deployed successfully.
- Create InitialContext in your client program to connect to application server.

- Lookup 24x7 Scheduler object and cast it to J24x7EJBHome interface.

- Create a remote interface J24x7EJB with the help of the home interface.

- Call openSession() business method to connect to 24x7 Scheduler.

- Call needed business methods to work with jobs/folders/agents etc.

- Call closeSession() business method to disconnect from 24x7 Scheduler.

- Destroy EJB with the help of the remove method.

Important Note: All the examples in this documentation are written for JBoss application server. However, you can easy adapt them for your own application server. Here is an example how to connect to application server and obtain J24x7EJB interface.

Example:

```
// Properties to connect to application server
Properties props = new Properties();

props.put(Context.INITIAL_CONTEXT_FACTORY,
  "org.jnp.interfaces.NamingContextFactory");
props.put(Context.PROVIDER_URL, "localhost:1099");
props.put(Context.URL_PKG_PREFIXES, "org.jboss.naming:org.jnp.interfaces");

// Lookup home interface
InitialContext jndiContext = new InitialContext(props);
Object ref = jndiContext.lookup("24x7Scheduler");

J24x7EJBHome home = (J24x7EJBHome) PortableRemoteObject.narrow(ref,
J24x7EJBHome.class);

// Create EJB bean and obtain remote interface
J24x7EJB j24x7EJB = home.create();
```

Exceptions and Error Handling

All methods can throw J24x7EJBException. Two error codes are supported:

Error codes:

REMOTE_OPERATION_ ERROR	Returned if RemoteException has occurred during remote call to 24x7 Scheduler
INVALID_DATA	Returned if the operation can't be executed successfully because of incorrect method parameters.

Exception's method `getMessage()` can be used to obtain the error message. See code examples in the following topics for more information on how to use this method.

Constants

The following constants can be use in the code

Data Type	Name	Value	Comments
Integer	OPERATION_SUCCESSED	Integer(1)	Method call return code
Integer	OPERATION_FAILED	Integer(-1)	Method call return code
Integer	NOT_FOUND	Integer(-4)	Method call return code
Integer	POSTPONED_SHUTDOWN	Integer(-100)	Method call return code
int	SECURITY_OFF	-1	User's security role assignment.
int	USER_ADMIN	0	User's security role assignment. Synonym ROLE_ADMIN
int	USER_POWER	1	User's security role assignment. Synonym ROLE_STANDARD
int	USER_OPER	2	User's security role assignment. Synonym ROLE_RESTRICTED
int	USER_GUEST	3	User's security role assignment. Synonym ROLE_GUEST
int	RC_INFO	0	Job event message severity
int	RC_WARNING	1	Job event message severity
int	RC_ERROR	2	Job event message severity
int	REMOTE_OPERATION_ERROR	1	Error code returned by J24x7EJBException
int	INVALID_DATA	2	Error code returned by J24x7EJBException

J24x7EJB Public Methods

addAgentProfile

public void addAgentProfile (String profileName, String comMethod, String location, String port, String options) throws J24x7EJBException

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

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
location	The location of the Agent. 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)
port	Specify 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.
options	Reserved for future use.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

updateAgentProfile

deleteAgentProfile

getAgentList

Example:

```
try
{
    j24x7EJB.addAgentProfile("AgentProfile Test", "WinSock",
                        "localhost", "1092", "");
```

```
}
catch (J24x7EJBException e)
{
    System.out.println("J24x7EJBException " + e.toString());
    System.out.println(e.getCode() + " " + e.getMessage());
}
```

addASGroup

public Integer addASGroup(String name, String agents, IntegerHolder groupId) throws J24x7EJBException

The **addASGroup** method creates new Auto-scaling group.

Parameters:

Name	Name of the new Auto-scaling Group. This name must be unique.
agents	Comma-separated list of Remote Agent identifiers.
groupId	Reference variable for receiving unique identifier of the new auto-scaling group

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> addAgentProfile
>
> deleteAgentProfile
>
> deleteASGroup
>
> getAgentList

Example:

```
try
{
    IntegerHolder id = new IntegerHolder(0);

    j24x7EJB.addASGroup("Group 1", "190,191,198", id);
}
catch (J24x7EJBException e)
{
    System.out.println("J24x7EJBException " + e.toString());
    System.out.println(e.getCode() + " " + e.getMessage());
}
```

addCalendar

public Integer addCalendar(String calendarName, Boolean exclusive) throws J24x7EJBException;

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

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

Return: Returns Id of the new calendar.

Parameters:

calendarName	Name of the new calendar. This name must be unique.
exclusive	Type of the calendar. Two types of calendars are supported "Exclusive" and "Inclusive". "Exclusive" calendars define set of dates, which could be considered as exception dates or holidays. "Inclusive" calendars are the opposite.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> addCalendarDay

Example:

```
try
{
  Integer id = j24x7EJB.addCalendar("Calendar Test1", new Boolean(true)));
  System.out.println("Created calendar ID: " + id);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

addCalendarDay

public Integer addCalendarDay(Integer calendarId, Integer year, Integer month, Integer day, String description) throws J24x7EJBException

The **addCalendarDay** method adds a new day to the calendar.

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

Return: Returns Id of the created calendar day.

calendarId,	Id of the calendar
year	Year part of the day date. Supported range 1900-3000
month	Month part of the day date. Supported range 1-12
day	Day part of the day date. Supported range 1-31. Day must be a valid day for the specified Year and Month.
description	Description of the day limited by 50 characters

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

                addCalendar

Example:

```
try
{
  Integer id = j24x7EJB.addCalendarDay(new Integer(38), new Integer(2005),
      new Integer(10), new Integer(24), "Description");
  System.out.println("Created calendar day id: " + id);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

addDatabaseProfile

public void addDatabaseProfile(String profileName, String databaseDriver, String server, String databaseName, Boolean autoCommit, String user, String password) throws J24x7EJBException

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

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.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> updateDatabaseProfile
>
> getDatabaseProfile
>
> deleteDatabaseProfile
>
> getDatabaseList

Example:

```
try
{
  j24x7EJB.addDatabaseProfile("Database Profile Test",
                    "MS SQL Server 7.x and later", "Neptune",
                    "warehouse", new Boolean(true), "sa", "******");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

addHoliday

public void addHoliday(Integer year, Integer month, Integer day, String description) throws J24x7EJBException

WARNING: Method addHoliday is deprecated and provided for backward compatibility only. Don't use this method in new applications.

The **addHoliday** method adds a new holiday to the [default] calendar.

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.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> deleteHoliday
>
> getHolidays

Example:

```
try
{
j24x7EJB.addHoliday(new Integer(2005), new Integer(11), new Integer(27),
      "Thanksgiving Day");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

addJobQueue

public void addJobQueue(String queueName, Integer maxSize) throws J24x7EJBException

The **addJobQueue** method creates a 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.

Parameters:

queueName	Name of the new Job Queue. This name should be unique.
maxSize	Max queue size in Mbytes. 1 Mbytes should be sufficient in most cases.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> updateJobQueue
>
> getJobQueue
>
> deleteJobQueue
>
> getJobQueueList

Example:

```
try
{
  j24x7EJB.addJobQueueEx("Payroll jobs",

             new Integer(1), new Integer(200), new Boolean(true)

             "alerter@domain.com", null, "helpdesk@domain.com");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

addJobQueueEx

public void addJobQueueEx(String queueName, Integer maxSize, Integer maxJobs, Boolean nearCapacityAlerts, String emailSender, String emailPassword, String emailRecipients) throws J24x7EJBException

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

Note: In 24x7 Scheduler for Windows adding 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 new queue is available immediately to run the jobs.

Parameters:

queueName	Name of the Job Queue. This name should be unique.
maxSize	Max queue size in Mbytes. 1 Mbytes should be sufficient in most cases.
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.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> updateJobQueueEx
>
> getJobQueue
>
> deleteJobQueue
>
> getJobQueueList

Example:

```
try
{
  j24x7EJB.addJobQueueEx("Payroll jobs",
            new Integer(1), new Integer(200), new Boolean(true)
            "alerter@domain.com", null, "helpdesk@domain.com");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

addMessage

public void addMessage(String description, String defaultActionType, String messageType, Boolean enabled, Integer ownerJobId, String defaultMessage, String subscribers, IntegerHolder messageId) throws J24x7EJBException

The addMessage method creates new message definition in the system settings.

Parameters:

description	Message definition description. Use it as a message name reference.
defaultActionType	One of the following one character codes: **D** – do nothing, meaning do not automatically delete the message from the message queue after it's received. The message will expire and get purged automatically later. **B** – delete received message from the message queue before executing jobs listed in the message subscriber array.. **A** – delete received message from the message queue after executing jobs listed in the message subscriber array.
messageType	This parameter is reserved for future use. In the current version always specify **T** or null value.
enabled	Message state, enabled/disabled. Disabled messages are ignored and not processed automatically when they are received.
ownerJobId	The numeric identifier of the job that owns the message. This an optional property mostly for documentation purpose to make it easier to see when job is expected to send this type of messages.
defaultMessage	The default data sent with the message. The data format is user-defined, could be a plan text, XML, JSON, or anything else required.
subscribers	Comma-separated list of job identifiers or job names. The listed jobs get launched when the message is received.
messageId	The holder variable populated with the unique message identifier of the created message definition

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

      deleteMessage

Example:

```
try
{
  IntegerHolder id = new IntegerHolder(0);
  j24x7EJB.addMessage("New message", "B", "T", true,
    "{ \"name\" : \"George\" }", "185,186,187", id);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

addTemplate

public void addTemplate(String section, String name, String template, String text) throws J24x7EJBException

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

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.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

deleteTemplate

setTemplate

getTemplate

getJobTemplateData

getTemplateCatalog

setJobTemplateData

Example:

```
try
{
  j24x7EJB.addTemplate("FTP jobs",
                       "Upload Monthly Reports",
                       "%HOME%\\Templates\\ftp_month_reports.ini",
                       "TEMPLATE_CODE");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

changeFolder

public void changeFolder(String jobID, String targetFolderID) throws J24x7EJBException

The **changeFolder** method moves a job with the specified **JobID** to a job folder specified by the **TargetFolderID** value.

Parameters:

jobID	Job ID converted to string or job name.
targetFolderID	Folder ID converted to string or folder name.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> createJob
>
> setJobProperty
>
> createFolder

Example:

```
try
{
  j24x7EJB.changeFolder("folder ejb", "folder ejb name change");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

closeSession

public void closeSession( ) throws J24x7EJBException

The **closeSession** method closes work session with the J24x7EJB and terminates the connection. The session must be previously opened using the **openSession** method.

The **closeSession** method does not destroy **J24x7EJB.**

Parameters: None

Throws:

Throws J24x7EJBException exception. The error code is REMOTE_OPERATION_ERROR. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> openSession

Example:

```
try
{
  // insert valid variable values here
  j24x7EJB.openSession(portNo, hostName, userID, userPass, serial);
  j24x7EJB.closeSession();
  System.out.println("session closed");
}
catch (J24x7EJBException e)
{
  System.out.println("InteractionException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

createFolder

public Integer createFolder(String name, String description) throws J24x7EJBException

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

Return: Returns 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.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> deleteFolder
>
> getFolderProperty
>
> setFolderProperty

Example:

```
try
{
  Integer id = j24x7EJB.createFolder("jobFolder 1 ", "Description 1"));
  System.out.println("Created folder Id: " + id);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

createJob

public Integer createJob(String jobDefinition) throws J24x7EJBException

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

Return: Job ID of the created job

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.
	For supported job properties and their JDL names see Job Properties in JDL Format topic.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the

operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> setJobProperty
>
> setJobTemplateData
>
> changeFolder
>
> updateJob
>
> disableJob
>
> protectJob
>
> runJob

Example:

```
try
{
  String jobDefinition = "NAME=My Job 1\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]";
  Integer id = j24x7EJB.createJob(jobDefinition);
  System.out.println("Created job Id: " + id);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```


deleteAgentProfile

public void deleteAgentProfile(String profileName) throws J24x7EJBException

The **deleteAgentProfile** method deletes specified profile of 24x7 Remote Agent.

Parameters:

profileName	Name of an existing Remote Agent Profile.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> addAgentProfile
>
> updateAgentProfile

Example:

```
try
{
  j24x7EJB.deleteAgentProfile("AgentProfileTest");
  System.out.println("agent profile deleted.");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

deleteASGroup

public void deleteASGroup(Integer groupId) throws J24x7EJBException

The **deleteASGroup**method deletes specified auto-scaling group.

Parameters:

groupId	Auto-scaling group identifier

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> addASGroup
>
> addAgentProfile
>
> updateAgentProfile

Example:

```
try
{
  j24x7EJB.deleteASGroup(190);
  System.out.println("group deleted.");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

deleteCalendar

public void deleteCalendar(Integer calendarId) throws J24x7EJBException

The **deleteCalendar** method deletes specified calendar.

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

Parameters:

calendarId	Id of deleted calendar

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

Example:

```
try
{
  j24x7EJB.deleteCalendar(new Integer(35));
  System.out.println("calendar deleted. ");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

deleteCalendarDay

public void deleteCalendarDay(Integer calendarId, Integer year, Integer month, Integer day) throws J24x7EJBException

The **deleteCalendar** method deletes specified calendar.

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

Parameters:

calendarId,	Id of the calendar
year	Year part of the day date. Supported range 1900-3000
month	Month part of the day date. Supported range 1-12
day	Day part of the day date. Supported range 1-31. Day must be a valid day for the specified Year and Month.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

Example:

```
try
{
  j24x7EJB.deleteCalendarDay(new Integer(33), new Integer(2005),
                        new Integer(9), new Integer(27));
  System.out.println("calendar day deleted.");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

deleteDatabaseProfile

public void deleteDatabaseProfile(String profileName) throws J24x7EJBException

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

Parameters:

profileName	Name of an existing Database Profile.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> addDatabaseProfile
>
> updateDatabaseProfile

Example:

```
try
{
  j24x7EJB.deleteDatabaseProfile("DatabaseProfileTest");
  System.out.println("database profile deleted.");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

deleteFolder

public void deleteFolder(String folderID)

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

Parameters:

folderID	Folder ID converted to string or folder name.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> createFolder
>
> setFolderProperty

Example:

```
try
{
```

```
  j24x7EJB.deleteFolder("folder1");
  System.out.println("folder deleted.");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

deleteHoliday

public void deleteHoliday(Integer year, Integer month, Integer day) throws J24x7EJBException

WARNING: Method deleteHoliday is deprecated and provided for backward compatibility only. Don't use this method in new applications.

The **deleteHoliday** method deletes holiday record from the 24x7 Holiday Table.

Parameters:

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

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

>   addHoliday
>
>   getHolidays

Example:
```
try
{
  j24x7EJB.deleteHoliday(new Integer(2005), new Integer(10), new Integer(27));
  System.out.println(" holiday deleted.");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

deleteJob

public void deleteJob(String jobID) throws J24x7EJBException

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

Parameters:

jobID	Job ID converted to string or job name.

Throws:

J24x7EJBException if RemoteException has occurred (code= REMOTE_OPERATION_ERROR) or if the operation can't be executed successfully, cause invalid data (code= INVALID_DATA). Use the getMessage() method of Throwable class to obtain the error message.

See also:

> disableJob
>
> createJob
>
> deleteFolder

Example:

```
try
{
  j24x7EJB.deleteJob("11");
  System.out.println("Job 11 deleted.");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

deleteJobQueue

public void deleteJobQueue(String queueName) throws J24x7EJBException

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

Note: In 24x7 Scheduler for Windows 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.

Parameters:

queueName	Name of an existing Job Queue.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

addJobQueue

updateJobQueue

getJobQueue

getJobQueueList

Example:

```
try
{
  j24x7EJB.deleteJobQueue("JobQueue1");
  System.out.println("Job queue deleted. ");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

deleteMessage

public void deleteMessage(String messageName) throws J24x7EJBException

The **deleteMessage** method deletes specified message definition.

Parameters:

messageName	Name of an existing Message.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

addMessage

Example:
```
try
{
  j24x7EJB.deleteMessage("Pathing Trigger");
  System.out.println("Message deleted. ");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

deleteTemplate

public void deleteTemplate(String section, String name, Boolean deleteFile) throws J24x7EJBException

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

and optionally deletes the template file.

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.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> addTemplate
>
> getTemplate
>
> getTemplateCatalog
>
> setTemplate
>
> setJobTemplateData

Example:
```
try
{
  j24x7EJB.deleteTemplate("Web reports", "FTP upload", new Boolean(false));
```

```
   System.out.println("template deleted.");
}
catch (J24x7EJBException e)
{
   System.out.println("J24x7EJBException " + e.toString());
   System.out.println(e.getCode() + " " + e.getMessage());
}
```

disableJob

public void disableJob(String jobID) throws J24x7EJBException

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.

Parameters:

JobID	Job ID converted to string or job name

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> enableJob
>
> deleteJob

Example:

```
try
{
   j24x7EJB.disableJob("11");
   System.out.println("Job 11 disabled.");
}
catch (J24x7EJBException e)
{
   System.out.println("J24x7EJBException " + e.toString());
   System.out.println(e.getCode() + " " + e.getMessage());
}
```

enableJob

public void enableJob(String jobID) throws J24x7EJBException

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

Parameters:

jobID	Job ID converted to string or job name.

See also:

> disableJob
>
> deleteJob

Example:

```
try
{
  j24x7EJB.enableJob("15");
  System.out.println("Job 15 enabled.");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getAgentList

public String getAgentList(Boolean htmlFormat) throws J24x7EJBException

The **getAgentList** method obtains list of names of configured 24x7 Remote Agent Profiles. If the method succeeds, returned variable is populated with the agent list. If **htmlFormat** is true, the returned 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 the name list. If **htmlFormat** is true, name list is a text formatted as a HTML table; otherwise, it is populated with a plain text containing each entry on a new line.

Parameters:

htmlFormat	The output format in which you want to obtain the list of 24x7 Remote Agent Profiles.

Value Meaning:

true - format output as HTML text

false - format output as plain text

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

>   addAgentProfile
>
>   deleteAgentProfile
>
>   getAgentProfile

Example:

```
try
{
  String agentList = j24x7EJB.getAgentList(new Boolean(true));
  System.out.println(agentList);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getAgentProfile

public AgentProfileContent getAgentProfile(String profileName) throws J24x7EJBException

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

Return: Returns the object of **AgentProfileContent** type, that has following fields: comMethod (name of the communication driver), location (location of the agent), port (the communication port number), options(additional communications options)

Parameters:

profileName	Remote Agent profile name.

Throws:

J24x7EJBException if RemoteException has occurred (code= REMOTE_OPERATION_ERROR) or if the operation can't be executed successfully, cause invalid data (code= INVALID_DATA). Use the getMessage() method of Throwable class to obtain the error message.

See also:

>   addAgentProfile
>
>   updateAgentProfile
>
>   deleteAgentProfile
>
>   getAgentList

Example:

```
try
{
  AgentProfileContent agentProfileContent =
       j24x7EJB.getAgentProfile("AgentProfile Test");
  System.out.println(" ComMethod " + agentProfileContent.getComMethod());
  System.out.println(" Location " + agentProfileContent.getLocation());
  System.out.println(" Port " + agentProfileContent.getPort());
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getCalendar

public CalendarContent getCalendar(Integer calendared) throws J24x7EJBException

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

Return: Returns the object of **CalendarContent** type, that has following fields: name (name of the calendar) and exclusive (type of calendar)

Parameters:

calendarID	Numeric ID of the business calendar whose properties to return.

Throws:

J24x7EJBException if RemoteException has occurred (code= REMOTE_OPERATION_ERROR) or if the operation can't be executed successfully, cause invalid data (code= INVALID_DATA). Use the getMessage() method of Throwable class to obtain the error message.

See also:

addCalendar

getCalendarList

deleteCalendar

getCalendarDays

Example:

```
try
{
  CalendarContent calendarContent =
       j24x7EJB.getCalendar( new Integer(55) );
  System.out.println(" Name " + calendarContent.getName());
  System.out.println(" Is Exclusive " + calendarContent.getExclusive());
}
catch (J24x7EJBException e)
```

```
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getCalendarDays

public String getCalendarDays(Integer calendarId, Integer year, Boolean htmlFormat) throws J24x7EJBException

The **getHolidays** method obtains list of calendar days for the specified **year**. If the method succeeds, it returns the list of days. If **htmlFormat** is true, the returned 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 yes, month, day, and description separated by tab characters.

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

Note: You can use split( ) or getToken( ) method to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **getToken** method.

Return: Returns the list of holidays.

Parameters:

calendarID	Numeric ID of the business calendar whose days to return.
year	The year for which you want to obtain the list of days.
htmlFormat	The output format in which you want to obtain the list of days.

Value Meaning:

true - format output as HTML text

false - format output as plain text

Example:

```
try
{
  String list = j24x7EJB.getCalendarDays(new Integer(21), new Integer(2005),
                                          new Boolean(true));
  System.out.println(list);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
```

```
}
```

getCalendarList

public String getCalendarList(Boolean htmlFormat) throws J24x7EJBException

The **getCalendarList** method obtains list of names of configured Business Calendars. If the method succeeds, returned variable is populated with the list of calendars. If **htmlFormat** is true, the returned 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.

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

Note: You can use split() or getToken() method to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **getToken** method.

Return: Returns the list of database profile names.

Parameters:

htmlFormat	The output format in which you want to obtain the list of Business Calendars

Value Meaning:

true - format output as HTML text

false - format output as plain text

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> addCalendar
>
> deleteCalendar
>
> getCalendar
>
> getCalendarDays

Example:

```
try
{
  String list = j24x7EJB.getCalendarList(new Boolean(false));
  System.out.println(list);
```

```
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getDatabaseList

public String getDatabaseList(Boolean htmlFormat) throws J24x7EJBException

The **getDatabaseList** method obtains list of names of configured Database Profiles. If the method succeeds, returned variable is populated with the list of database profiles. If **htmlFormat** is true, the returned 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 split() or getToken() method to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **getToken** method.

Return: Returns the list of database profile names.

Parameters:

htmlFormat	The output format in which you want to obtain the list of Database Profiles.

Value Meaning:

true - format output as HTML text

false - format output as plain text

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> addDatabaseProfile
>
> deleteDatabaseProfile
>
> getDatabaseProfile

Example:

```
try
{
  String list = j24x7EJB.getDatabaseList(new Boolean(false));
  System.out.println(list);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getDatabaseProfile

public DatabaseProfileContent getDatabaseProfile(String profileName) throws J24x7EJBException

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

Return: Returns the object of **DatabaseProfileContent** type. It has following fields: databaseDriver (name of the database driver), server (name of the database server), databaseName (name of the database), autoCommit (state of the AutoCommit mode), user (name of the user to logon), password (password used to logon to the database)

Parameters:

profileName	Database profile name.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

       addDatabaseProfile

       updateDatabaseProfile

       deleteDatabaseProfile

Example:

```
try
{
  DatabaseProfileContent databaseProfileContent =
      j24x7EJB.getDatabaseProfile("Database Profile Test");
  System.out.println(" AutoCommit " + databaseProfileContent.getAutoCommit());
  System.out.println(" DatabaseDriver " +
      databaseProfileContent.getDatabaseDriver());
  System.out.println(" DatabaseName " +
      databaseProfileContent.getDatabaseName());
```

```
  System.out.println(" Password " + databaseProfileContent.getPassword());
  System.out.println(" Server " + databaseProfileContent.getServer());
  System.out.println(" User " + databaseProfileContent.getUser());
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getFolderList

public String getFolderList(Boolean htmlFormat) throws J24x7EJBException

The **getFolderList** method obtains list of all job folders. If the method succeeds, returned variable is populated with the list of job folder IDs and names.

Note: You can use split( ) or getToken( ) method to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **getToken** method.

Return: Returns with the list of job folder IDs and names. If **htmlFormat** is true, the list of job folder IDs is a text formatted as a HTML table; otherwise it a plain text containing each entry on a new line. Each line contains folder ID and name separated by a tab character.

Parameters:

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

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

createFolder

deleteFolder

setFolderProperty

getFolderProperty

Example:

```
try
{
  String list = j24x7EJB.getFolderList(new Boolean(false));
  System.out.println(list);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getFolderProperty

public String getFolderProperty(String folderID, String propertyName) throws J24x7EJBException

The **getFolderProperty** method obtains value of the specified job **propertyName** for the specified **folderID**. 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, it returns the value of the folder property.

Return: Returns the value of the folder property.

Parameters:

FolderID	Folder ID converted to string or Folder name.
PropertyName	The folder property name.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

createFolder

setFolderProperty

Example:

```
try
{
  String value = j24x7EJB.getFolderProperty("1", "DESCRIPTION");
  System.out.println(value);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
```

```
    System.out.println(e.getCode() + " " + e.getMessage());
}
```

getForecast

public String getForecast(Boolean htmlFormat) throws J24x7EJBException

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

Return: Returns String with job forecast report. If **htmlFormat** is true, returned 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.

Parameters:

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

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

getMonitor

getJobQueueMonitor

Example:

```
try
{
  String value = j24x7EJB.getForecast(new Boolean(true));
  System.out.println(value);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getGlobalVariable

public String getGlobalVariable(String variableName) throws J24x7EJBException

The **getGlobalVariable** method obtains 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 value of the specified global variable converted to String format.

Parameters:

variableName	The name of the global variable that must exist on the target 24x7 Agent or Master Scheduler. Variable names are case-insensitive.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> setGlobalVariable

Example:

```
try
{
  String value = j24x7EJB.getGlobalVariable("test_var");
  System.out.println(value);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getHolidays

public String getHolidays(Integer year, Boolean htmlFormat) throws J24x7EJBException

WARNING: Method getHolidays is deprecated and provided for backward compatibility only. Don't use this method in new applications.

The **getHolidays** method obtains list of holidays for the specified **year**. If the method succeeds, it returns the list of holidays. If **htmlFormat** is true, the returned 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 split( ) or getToken( ) method to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **getToken** method.

Return: Returns the list of holidays.

Parameters:

Year	The year for which you want to obtain the list of holidays.
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

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

>   addHoliday

>   deleteHoliday

Example:

```
try
{
  String list = j24x7EJB.getHolidays(new Integer(2005), new Boolean(true));
  System.out.println(list);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getJobDefinition

public String getJobDefinition(String jobID, Boolean htmlFormat) throws J24x7EJBException

The **getJobDefinition** method obtains values of all job properties for the specified **jobID**. If the method succeeds, returned variable is populated with the job definition. If **htmlFormat** is true, the returned 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 returned 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 job definition.

Parameters:

jobID	Job ID converted to string or job name.
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

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> createJob
>
> getJobProperty
>
> getFolderProperty
>
> getJobTemplateData
>
> updateJob

Example:

```
try
{
  String value = j24x7EJB.getJobDefinition("2", new Boolean(true));
  System.out.println(value);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getJobList

public String getJobList(Boolean htmlFormat) throws J24x7EJBException

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

If the **htmlFormat** is true, returned variable is populated with a text formatted as an HTML table; otherwise the returned 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 split( ) or getToken( ) method to parse the returned list and extract individual elements or convert it into an array of elements. For 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 list of all jobs.

Parameters:

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

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

getJobListEx

getFolderList

Example:

```
try
{
  String list = j24x7EJB.getJobList(new Boolean(false));
  System.out.println(list);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getJobListEx

public String getJobListEx(Boolean htmlFormat, String propertyList) throws J24x7EJBException

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". For a list of supported properties see Job Properties in JDL Format topic.

If the **htmlFormat,** is true returned variable is populated with a text formatted as an HTML table; otherwise the returned 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 split( ) or getToken( ) method to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **getToken** method.

Below is a 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

Return: Returns of all jobs and optionally their properties.

Parameters:

htmlFormat	The output format in which you want to obtain the result.
propertyList	parameter is used to specify additional JDL properties. Separate multiple properties by commas. For example: "FOLDER_NAME,DISABLED,MODIFY_TIME". For a list of supported

	properties see <u>Job Properties in JDL Format</u> topic.

Value Meaning:

24x7 Remote Control Java Interface - 47 -

true - format output as HTML text

false - format output as plain text

propertyList - A comma separated list of job property names.

Throws:

Throws <u>J24x7EJBException</u> exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> getJobList
>
> getFolderList
>
> getJobProperty
>
> getJobPropertyEx

Example:

```
try
{
  String list = j24x7EJB.getJobListEx(new Boolean(true), "JOB_TYPE,DISABLED");
  System.out.println(list);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getJobLog

public String getJobLog(String jobID, Boolean htmlFormat) throws J24x7EJBException

The **getJobLog** method obtains all log records for the specified **jobID**. If the **htmlFormat** is true, returned 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 returned 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 split( ) or getToken( ) method to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **getToken** method.

Return: Returns all log records.

Parameters:

jobID	Job ID converted to string or job name
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

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

>
> getForecast
>
> getMonitor

Example:

```
try
{
  String value = j24x7EJB.getJobLog("61", new Boolean(true));
  System.out.println(value);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getJobQueueSize

public Integer getJobQueueSize(String queueName) throws J24x7EJBException

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

Return: Returns size of an existing job queue.

Parameters:

queueName	Job Queue name.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> addJobQueue
>
> updateJobQueue
>
> deleteJobQueue

Example:

```
try
{
  Integer value = j24x7EJB.getJobQueueSize("Job Queue Test");
  System.out.println(value);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getJobQueueList

public String getJobQueueList(Boolean htmlFormat) throws J24x7EJBException

The **getJobQueueList** method obtains list of names of configured Job Queues. If **htmlFormat** is true, returned 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 split( ) or getToken( ) method to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **getToken** method.

Return: Returns the list of names of configured Job Queues.

Parameters:

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

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> getJobQueue
>
> getJobQueueMonitor
>
> addJobQueue

Example:

```
try
{
  String list = j24x7EJB.getJobQueueList(new Boolean(false));
  System.out.println(list);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getJobQueueMonitor

public String getJobQueueMonitor(String queueName, Boolean htmlFormat) throws J24x7EJBException

The **getJobQueueMonitor** method obtains the list of running and waiting jobs in the specified job queue. If **htmlFormat** is true, the returned 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 split( ) or getToken( ) method to parse the returned list and extract individual elements or convert it into an array of elements. For 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.
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

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

getForecast

getMonitor

Example:

```
try
{
    String value = j24x7EJB.getJobQueueMonitor("Job Queue Test",
                                                new Boolean(true));
  System.out.println(value);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getJobProperty

public String getJobProperty(String jobID, String propertyName) throws J24x7EJBException

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 Properties in JDL Format topic.

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 Job Properties in JDL Format topic.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> setJobProperty
>
> getJobPropertyEx
>
> getJobDefinition
>
> getJobTemplateData

Example:

```
try
{
  String value = j24x7EJB.getJobProperty("61", "DESCRIPTION");
  System.out.println(value);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getJobPropertyEx

public String getJobPropertyEx(String jobID, String propertyNameList) throws J24x7EJBException

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 Properties in JDL Format topic.

 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 split( ) or getToken( ) method to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **getToken** method.

Return: Returns value of the property.

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 Job Properties in JDL Format topic.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> getJobProperty
>
> getJobDefinition
>
> setJobProperty
>
> setJobPropertyEx

getJobTemplateData

Example:

```
try
{
  String value = j24x7EJB.getJobPropertyEx("61", "DESCRIPTION");
  System.out.println(value);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getJobStatus

public Integer getJobStatus(String jobID) throws J24x7EJBException

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.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

runJob

getForecast

getJobQueueMonitor

getMonitor

Example:

```
try
{
  Integer value = j24x7EJB.getJobStatus("61");
  System.out.println(value);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getJobTemplateData

public JobTemplateContent getJobTemplateData(String jobID) throws J24x7EJBException

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 method returns the object of **JobTemplateContent** type.

Return: Object of **JobTemplateContent** type, that has properties and data used to create or modify the specified job. This type has following fields:

> templateName - saved job template.

> templateFile - saved job template file name.

> data - saved job data.

Parameters:

jobID	Job ID converted to string or job name.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> setJobTemplateData

> getJobDefinition

> getJobProperty

> getTemplate

Example:

```
try
{
  JobTemplateContent value = j24x7EJB.getJobTemplateData("1");
  System.out.println(" TemplateFile " + value.getTemplateFile());
  System.out.println(" TemplateName " + value.getTemplateName());
  System.out.println(" Data " + value.getData());
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getMonitor

public String getMonitor(Integer interval, Boolean htmlFormat) throws J24x7EJBException

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, returned variable is populated with job monitor report. If **htmlFormat** is true, the returned 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 split( ) or getToken( ) method to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **getToken** method.

Below is a 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 job monitor report.

Parameters:

Interval	The time interval within which the monitor report is created. The interval is specified in hours.
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

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

getForecast

getJobQueueMonitor

Example:

```
try
{
  String value = j24x7EJB.getMonitor(new Integer(2), new Boolean(true));
  System.out.println(value);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getTemplate

public String getTemplate(String templateFile) throws J24x7EJBException

The **getTemplate** method obtains contents of the specified template file. If the method succeeds, returned variable is populated with the template text.

Return: Returns template.

Parameters:

templateFile	Template file name.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

setTemplate

getTemplateCatalog

Example:

```
try
{
  String value = j24x7EJB.getTemplate("%HOME%\\Template\\ftp.ini");
  System.out.println(value);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getTemplateCatalog

public String getTemplateCatalog() throws J24x7EJBException

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, returned variable is populated with the template catalog.

Return: Returns template catalog.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> setTemplate
>
> getTemplate
>
> addTemplate

Example:

```
try
{
  String value = j24x7EJB.getTemplateCatalog();
  System.out.println(value);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getToken

public String getToken(String holder, String separator) throws J24x7EJBException

The **getToken** method parses the source string (**holder**) and obtains first string 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 **holder**.

Parameters:

holder	Source string from which you want to remove a token.
separator	A string whose value is searched within the source string. The **Separator** value must consist of one or more characters.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

Note: getToken can be used with many methods that return value lists and table-like multi-value results. Standard `split()` method of the `String` class can be also used for this purpose. The main difference between these methods is that the `split()` method operates on arrays while getToken method operates on simple Strings.

Warning: It recommended that you use the `split()` method whenever possible as getToken method is going to be deprecated in future versions.

See also:

> getAgentList
>
> getJobList
>
> getFolderList
>
> getJobPropertyEx
>
> getMonitor
>
> getJobQueueList
>
> getJobQueueMonitor

Example:

```
try
{
  String value = j24x7EJB.getToken("eter  ytryr", "r");
  System.out.println(value);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getStatusReport

public void getStatusReport(String destinDir) throws J24x7EJBException

The **getStatusReport** method obtains **HTML Status Reports** from the 24x7 Master Scheduler and copies them to the destination directory specified in the **DestinDir** parameter.

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. The process executing this method must have write permissions for the specified directory.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> getJobLog

Example:

```
try
{
  System.out.println(" getStatusReport");
  j24x7EJB.getStatusReport("c:\\reports");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

getUserRole

public Integer getUserRole(String userName) throws J24x7EJBException

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** (value = -1) – the security system is turned off. All users have unlimited privileges.

- **ROLE_ADMIN** (value = 0) – user is a member of Administrators group.

- **ROLE_STANDARD** (value = 1) - user is a member of Standard Privileges group. In 24x7 Scheduler Multi-platform Edition the corresponding group name is Power Users.

- **ROLE_RESTRICTED** (value = 2) - user is a member of Restricted Privileges group. In 24x7 Scheduler Multi-platform Edition the corresponding group name is Operators.

- **ROLE_GUEST** (value = 3) - user is a member of Guests group

For more information about returned values see the Constants topic.

Parameters:

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

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

openSession

Example:

```
try
{
  Integer value = j24x7EJB.getUserRole("scott");
  System.out.println(value);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

logMessage

public void logMessage(Integer sourceID, String sourceName, Integer eventSeverity, String message, Boolean forceReportUpdate) throws J24x7EJBException

The **logMessage** method adds new record to the 24x7 Scheduler job 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.

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 **HTML Status Report** or perform a deferred update.

Value Meaning:

true - update immediately and force immediate rebuild and refresh of HTML Status Reports and the job log.

false – add the message to log without forcing the rebuild of HTML Status Reports.

Important Notes:

- If the **HTML Status Report** option is not enabled, **forceReportUpdate** parameter is ignored.

- "forceReportUpdate = true" ensures the reports are updated immediately so people who monitor that report can see this change in a real-time. However you should remember that immediate updates impacts the performance of the 24x7 Scheduler.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> getJobLog

> getStatusReport

Example:

```
try
{
  j24x7EJB.logMessage(new Integer(15), "Test",
                 new Integer(RC_INFO), "test message",
                 new Boolean(false));
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

holdJob

Public void 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.

Parameters:

JobQueueID	Run-time job number in queue.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> deleteJob
>
> disableJob
>
> killJob
>
> RELEASEJOB

Example:

```
try
{
  j24x7EJB. holdJob( 22465 );
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

killJob

public void 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.

Important Notes:

The **killJob** method can be used with all types of jobs both jobs launched using runJob command and jobs started automatically by the scheduler

Parameters:

JobQueueID	Run-time job number in queue.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> deleteJob
>
> disableJob
>
> holdJob

Example:

```
try
{
   j24x7EJB.killJob( 22465 );
}
catch (J24x7EJBException e)
{
   System.out.println("J24x7EJBException " + e.toString());
   System.out.println(e.getCode() + " " + e.getMessage());
}
```

killShellCommand

Public void killShellCommand (Long ProcessID) throws RemoteException

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

Parameters:

ProcessID	System process ID.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

runShellCommand

getProcessList

killJob

Example:

```
try
{
    j24x7EJB.killShellCommand( 254 );
}
catch (J24x7EJBException e)
{
    System.out.println("J24x7EJBException " + e.toString());
    System.out.println(e.getCode() + " " + e.getMessage());
}
```

openSession

public void openSession(Integer portNo, String hostName, String userID, String userPass, String serial) throws J24x7EJBException

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 j24x7EJB class and a session is still open, the j24x7EJB 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.

Parameters:

portNo	The port number for the 24x7 Scheduler server (for example, 1096).
hostName	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)
userID	The user ID that you want to use for the connection.
userPass	The password that you want to use for the connection.
serial	The Serial Number given for the **24x7 Remote Control Java interface**

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> closeSession

Example:

```
System.out.println("connect");
Integer portNo = new Integer(1097);
String hostName = new String("localhost");
String userID = new String("");
String userPass = new String("");
String serial = new String("00000-9999999-55555");

try
{
  System.out.println("openSession");
  j24x7EJB.openSession(portNo, hostName, userID, userPass, serial);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

protectJob

public void protectJob(String jobID, String protectionType, String jobPassword) throws J24x7EJBException

The **protectJob** method can be used to change protection level for an already protected job, define protection level for an unprotected job. If the job is already protected you must specify valid **jobPassword** that matches the current job password.

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

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

>　　unprotectJobob

>　　disableJob

>　　createJob

Example:
```
try
{
  j24x7EJB.protectJob("67", "R", "1");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

releaseJob

Public void 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.

Parameters:

JobQueueID	Run-time job number in queue.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> holdJob
>
> enableJob

Example:

```
try
{
  j24x7EJB. releaseJob( 22465 );
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

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.

Parameters:

jobID	Job ID converted to string or job name.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> createJob
>
> runJob

Example:

```
try
{
  j24x7EJB.queueJob("10");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

runJob

public void runJob(String jobID, Boolean asynchronous) throws J24x7EJBException

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

Parameters:

jobID	Job ID converted to string or job name.
asynchronous	Controls how the job is processed.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

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 the job to complete before starting another job set **asynchronous** parameter to **false**. This will ensure that jobs run sequentially. If you start the job **asynchronous** but don't wait for it to complete set run **asynchronous**

parameter to **true**. Keep in mind that if you run a job **synchronous** (asynchronous=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 **asynchronous** property for jobs taking long time to run.

See also:

> createJob
>
> queueJob
>
> runScript
>
> runShellCommand

Example:

```
try
{
  j24x7EJB.runJob("10", new Boolean(true));
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

runScript

public void runScript(String script) throws J24x7EJBException

The **runScript** method executes the specified JavaScript script. The effect of this method is similar to running a job with the same script. The script is executed on the computer where the target 24x7 Scheduler is running.

Important Notes: This method is not currently supported.

Parameters:

script	The script that you want to execute.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> runJob
>
> runShellCommand
>
> utilRunScript

Example:

```
try
{
  j24x7EJB.runScript("Scheduler.pause( 10 )\n");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

runShellCommand

public void runShellCommand(String commandLine, String dir, Boolean async, Integer timeout) throws J24x7EJBException

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.

Parameters:

commandLine	The full or partial path and filename of the module to execute. **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. You can use 24x7 supported macro-parameters inside program name and command line parameters to pass dynamic information (such as the current month) to the scheduled program.
dir	The directory where the program executable finds associated files that are required to run the program. Most programs do not require an entry in this field so that you can specify an empty string (""). Occasionally 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 command
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 icon] **Note:** When running the command synchronously (a**sync=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**.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> runJob
>
> runScript
>
> utilRunScript

Example:

```
try
{
  j24x7EJB.runShellCommand("h:\\sales\\reports\\month_end.exe",
                           "", new Boolean(false), new Integer(30));
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

setGlobalVariable

public void setGlobalVariable(String variableName, String newValue) throws J24x7EJBException

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

![important note icon] **Important Note:** This method is only valid for 24x7 Scheduler Windows Edition.

Return: Returns value of the specified global variable converted to String format.

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.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

getGlobalVariable

Example:

```
try
{
  String value = j24x7EJB.getGlobalVariable("test_var", "55");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

setFolderProperty

public void setFolderProperty(String folderID, String propertyName, String newValue) throws J24x7EJBException

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".

Parameters:

folderID	Folder ID converted to string or folder name.
propertyName	The folder property name.
newValue	Then new property value.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> createFolder

> getFolderProperty

Example:

```
try
{
  j24x7EJB.setFolderProperty("56", "DESCRIPTION",
                        "New Folder description test");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

setJobProperty

public void setJobProperty(String jobID, String propertyName, String newValue) throws J24x7EJBException

The **setJobProperty** method changes value of the specified job **propertyName** for the specified job. The **propertyName** must be a valid JDL job property name. For more details see Job Properties in JDL Format topic.

Parameters:

jobID	Job ID converted to string or job name.
propertyName	The job property name. For supported job properties and their JDL names see Job Properties in JDL Format topic.
newValue	Then new value converted to string format.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> getJobProperty

> setJobPropertyEx

> updateJob

> setFolderProperty

> setJobTemplateData

Example:

```
try
{
  j24x7EJB.setJobProperty("69", "DESCRIPTION",
                          "new test job description test");

}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

setJobPropertyEx

public void setJobPropertyEx(String jobID, String propertyName, String newValue) throws J24x7EJBException

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 Properties in JDL Format topic.

 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

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 Job Properties in JDL Format topic.
newValueList	Tab-separated list of new property values converted to string format.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> setJobProperty
>
> getJobPropertyEx

        updateJob

        setFolderProperty

        setJobTemplateData

Example:

```
try
{
  j24x7EJB.setJobPropertyEx("69",
       "SCHEDULE_TYPE,SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY",
       "D,N,Y,N,Y,N,N,N");
}
catch (J24x7EJBException e)
{
    System.out.println("J24x7EJBException " + e.toString());
    System.out.println(e.getCode() + " " + e.getMessage());
}
```

setJobTemplateData

public void setJobTemplateData(String jobID, String templateName, String templateFile, String data)

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 developers to decide which internal format to use for the **data**.

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.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

        getJobTemplateData

        setJobProperty

        setTemplate

Example:

```
try
{
  j24x7EJB.setJobTemplateData("69", "template_test", "template_file", "teampte_data");

}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

setTemplate

public void setTemplate(String templateFile, String buffer) throws J24x7EJBException

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

Parameters:

templateFile	Template file name.
buffer	A String variable whose value replaces contents of the specified template file.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

addTemplate

getTemplate

getJobTemplateData

getTemplateCatalog

setJobTemplateData

Example:

```
try
{
  j24x7EJB.setTemplate("Template_filename", "Template_test");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

test

public Integer test(Integer x) throws J24x7EJBException

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.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

Example:
```
try
{
  System.out.println(j24x7EJB.test(new Integer(67)));
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

unprotectJob

public void unprotectJob(String jobID, String jobPassword) throws J24x7EJBException

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

Parameters:

jobID	Job ID converted to string or job name
jobPassword	Job password

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

>protectJob

>enableJob

>createJob

Example:

```
try
{
  j24x7EJB.unprotectJob("69", "pass");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

updateAgentProfile

public void updateAgentProfile(String profileName, String comMethod, String location, String port, String options)

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

Parameters:

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

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

>addAgentProfile

>deleteAgentProfile

>getAgentProfile

Example:

```
try
{
  j24x7EJB.updateAgentProfile("profile_name", "Winsock",
                      "192.168.100.1", "1096", "");
}
```

```
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

updateCalendar

public void updateCalendar(Integer calendarId, String calendarName, Boolean exclusive) throws J24x7EJBException

The **updateCalendar** method updates properties of an existing Business Calendar.

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

Parameters:

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

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> addCalendar
>
> deleteCalendar
>
> getCalendar

Example:

```
try
{
  j24x7EJB.updateCalendar(new Integer(7), "Payroll Calendar",
                          new Boolean(false));
}
catch (J24x7EJBException e)
{
  System.out.println("InteractionException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

updateDatabaseProfile

public void updateDatabaseProfile(String profileName, String databaseDriver, String server, String databaseName, Boolean autoCommit, String user, String password)

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

Parameters:

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

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> addDatabaseProfile
>
> getDatabaseProfile
>
> deleteDatabaseProfile

Example:

```
try
{
  j24x7EJB.updateDatabaseProfile("profile_name", "My database", "Neptune",
        "warehouse", new Boolean(true), "sa", "******");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

updateJob

public void updateJob(String jobID, String jobDefinition) throws J24x7EJBException

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

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 see <u>Job Properties in JDL Format</u> topic.

Throws:

Throws <u>J24x7EJBException</u> exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> createJob
>
> setJobProperty
>
> setJobTemplateData
>
> changeFolder
>
> disableJob
>
> protectJob
>
> runJob

Example:

```
try
{
  String jobDefinition = "NAME=updated 4 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=401\r\n" +
                         "QUEUE=[default]";

  j24x7EJB.updateJob("4", jobDefinition);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

updateJobQueue

public void updateJobQueue(String queueName, Integer maxSize) throws J24x7EJBException

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.

Parameters:

queueName	Name of the new Job Queue. This name should be unique.
maxSize	Max queue size in Mbytes. 1 Mbytes should be sufficient in most cases.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> addJobQueue
>
> deleteJobQueue

Example:

```
try
{
  j24x7EJB.updateJobQueue("My queue", new Integer(20));
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
}
```

updateJobQueueEx

public void updateJobQueue Ex(String queueName, Integer maxSize, Integer maxJobs, Boolean nearCapacityAlerts, String emailSender, String emailPassword, String emailRecipients) throws J24x7EJBException

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.

Parameters:

queueName	Name of the Job Queue.
maxSize	Max queue size in Mbytes. 1 Mbytes should be sufficient in most cases.
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.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

> addJobQueue
>
> deleteJobQueue

Example:

```
try
{
  j24x7EJB.updateJobQueueEx("Payroll jobs",
            new Integer(1), new Integer(200), new Boolean(true)
            "alerter@domain.com", null, "helpdesk@domain.com");
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
```

```
   System.out.println(e.getCode() + " " + e.getMessage());
}
```

utilRunScript

public String utilRunScript(String script) throws J24x7EJBException

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.

Important Notes: This method is not currently supported.

Return: Returns contents of the temporary file.

Parameters:

script	The script that you want to execute.

Throws:

Throws J24x7EJBException exception. In case if an error has occurred on the server (RemoteException) the error code is REMOTE_OPERATION_ERROR. In case if an error has occurred on the client (the operation can't be executed successfully because of an invalid parameter or data) the error code is INVALID_DATA. Use the getMessage() method of Throwable class to obtain the error message.

See also:

>   runJob

>   runShellCommand

>   runScript

Example:

```
try
{
  String script = "var files = Directory.dir(\"/home/logs/*\");\n" +
                  "file.save( output, files )"
  String result = j24x7EJB.utilRunScript(script);
  System.out.println("Log directory contents on the server: " + result);
}
catch (J24x7EJBException e)
{
  System.out.println("J24x7EJBException " + e.toString());
  System.out.println(e.getCode() + " " + e.getMessage());
```

}

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_E-MAIL	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	Script for Notification Action of Script Type **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
MSG_SCRIPT_ TYPE	Type of Script for Notification Action of Script Type, one of the following: **JAL**, **VBS** (Job Automation Language, Visual Basic Script) **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
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
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.

Licensing

1. **Single installation license:** A separate single installation license is required for every 24x7 Scheduler client. That's it, you need at least 2 single licenses (or a site license) for using 24x7 EJB interface 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 EJB client program.

2. **Site license:** 24x7 Scheduler site license covers unlimited usage of 24x7 servers and clients. Site license users can use 24x7 EJB components just as they use other components of the 24x7 Automation Suite. The usage of 24x7 Scheduler, 24x7 Remote Agents, 24x7 Web Console and 24x7 EJB interface, 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 EJB interface on your server. A separate site license is required for every web server utilizing 24x7 Remote Control Java.

4. **Redistribution:** You must obtain 24x7 EJB interface redistribution license before you can redistribute it with your program.

Important note about third party packages provided with the 24x7 EJB interface:
24x7 EJB interface 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 EJB interface.

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