

# JDF: JAMA Distribution Framework

Author: Marc Klaver  
Date: 1 April 2011  
Document version: 1.0.0  
Framework version: 0.9.24

# Contents

1	Introduction .....	4
1.1	Basics .....	4
1.2	Prerequisites.....	4
1.2.1	Secure copy server .....	4
1.2.2	Secure copy client.....	5
1.2.3	Key file generator .....	5
1.2.4	Unzip.exe and Zip.exe .....	5
1.2.5	Framework.....	5
2	Retrieving and installing the framework .....	6
2.1	BaseDistribution .....	7
2.2	include_binary .....	7
2.3	include_text.....	7
2.4	MP.....	7
2.5	output.....	7
2.6	tools .....	7
3	Configuring and building the framework.....	8
3.1	Configuring the framework .....	8
3.2	Changing build scripts .....	11
3.3	Changing the jdf.jdp file .....	12
3.3.1	FRAMEWORK_VERSION .....	12
3.3.2	VARs.....	13
3.3.3	REGISTRATION.....	13
3.3.4	PACKAGE_FILES .....	14
3.4	Default private key .....	15
3.5	Building the distribution MP .....	15
3.5.1	JDF.BaseDistribution.xml .....	16
3.5.2	jdf_v0.9.24.zip .....	16
4	Secure Copy Server.....	17
4.1	Required accounts .....	17
4.2	Private keys.....	17
4.2.1	jdfBaseDistribution.....	17
4.2.2	jdfCustomer_<name> .....	17
5	Importing the framework .....	18
5.1	Possible errors .....	19
5.2	JDF: Base distribution. Script failed to initialize (event id 200) .....	19
5.3	JDF: Base distribution. Failed to create at least one file (event id 400) .....	20
6	Using the framework .....	22
6.1	Loading the framework .....	22
6.2	Debugging.....	23
7	Framework reference.....	24
7.1	Properties.....	24
7.2	Methods .....	25
7.2.1	AddToPackage .....	25
7.2.2	CreateDirectory .....	25
7.2.3	CreatePackage .....	25
7.2.4	DeletePackage.....	26
7.2.5	DownloadPackage .....	26
7.2.6	ExpandString .....	26
7.2.7	ExtractFromPackage.....	27
7.2.8	ExtractPackage .....	27
7.2.9	GetFile.....	27

7.2.10	GetInfoString .....	28
7.2.11	GetVar .....	28
7.2.12	InitializeFramework .....	29
7.2.13	PutFile .....	30
7.2.14	RegisterFramework .....	30
7.2.15	ScheduleTaskIn .....	31
7.2.16	UnRegisterFramework.....	31
7.2.17	UploadPackage .....	31
8	Template script file template.vbs .....	32

## 1 Introduction

This document describes the use of the Jama Distribution Framework (JDF) the prerequisites and the configuration of the framework. The framework is intended to exchange files between a computer with a SCOM agent and a central location. As the name implies, it is a framework. Which means it does not do anything, with the exception of allowing others scripts to exchange files.

### 1.1 Basics

The framework is basically a secure copy implementation, which can be used with the SCOM agent through scripting. The framework supplies the functions to retrieve from and send files to a central location.

Like the SCOM agent itself, the framework can only be used as a client and connections should always be initiated from the computer holding the framework. This means that the central location cannot initiate a connection.

On a fixed location on the client computer, the framework file will be distributed. After registering the framework, other scripts will be capable of using the framework for file distribution.

The framework only uses secure copy to retrieve and sent files. It does this by using public/private SSH keys, no username and password combinations are used.

The framework itself is based on customer information. This means that a customer must be defined for the framework (either in the registry or at initialization of the framework). Every customer defined must use a different SSH key file, which will allow to separate the data on a customer basis at the central location and make it impossible for customer A to see or change data from customer B.

On the central location a secure copy server must be available that supports public key authentication and can be used using the putty secure copy client (pscp.exe).

### 1.2 Prerequisites

Before you actually can start using the framework you need:

- A secure copy server
- A secure copy client
- Key file generator
- Unzip.exe and zip.exe
- The JDF framework

#### 1.2.1 Secure copy server

The secure copy server must be available on a central location, which must be reachable from all computers on which you want to use the framework. We use [WinSSHD](#) from bitvise and have configured the server to only accept secure copy connections using public key authentication.

### 1.2.2 Secure copy client

The secure copy client used in the framework is the putty secure copy client [pscp.exe](#). This file is part of the framework download.

### 1.2.3 Key file generator

The tool used for generating the SSH key files is [puttygen.exe](#). This file is part of the framework download.

### 1.2.4 Unzip.exe and Zip.exe

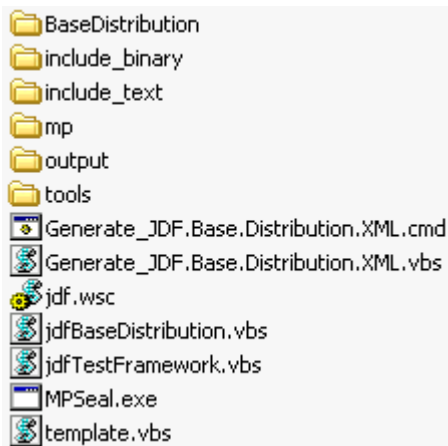
In order to reduce the required traffic, the data sent between the framework and the central server is compressed into a .zip file before use. The tools used for this purpose are the [info-zip](#) versions of unzip.exe and zip.exe. These files are part of the framework download.

### 1.2.5 Framework

The framework itself is a windows scripting component, called jdf.wsc. All functions are placed into this file. A separate distribution script is written and packed into a SCOM management pack. When importing this management pack, all agents will receive the framework. But before you can use the framework, it needs to be configured for your environment.

## 2 Retrieving and installing the framework

After downloading the framework and extracting the files you will have a directory structure as shown:



These files are present in the root.

### **Generate\_JDF.BaseDistribution.XML.cmd**

This file is used to build the management pack that holds and distributes the framework.

### **Generate\_JDF.BaseDistribution.XML.vbs**

This file combines the different files to build the requested management pack.

### **jdf.wsc**

The framework file.

### **jdfBaseDistribution.vbs**

The script used in the distribution rule to distribute the framework itself.

### **jdfTestFramework.vbs**

Example file for use with the framework.

### **MPSeal.exe**

Sealing application from Microsoft to seal the framework. This file is included for convenience.

### **template.vbs**

Template file on how to use the framework

[You can download the framework here](#)

Beside these files, 6 sub directories exist in the framework download.

### **2.1 BaseDistribution**

The generated framework package file is placed into this sub directory.

### **2.2 include\_binary**

Binaries that need to be converted to get included in the base distribution rule. At present this is the pscp.exe only.

### **2.3 include\_text**

Text files that need to be converted to get included in the base distribution rule. You should place your default private key file into this directory.

All other files will be generated during the build of the management pack.

### **2.4 MP**

All depended management packs (.mp files) are placed in this directory. These files are required when building a sealed management pack from the generated .xml file and are included for convenience.

### **2.5 output**

Output directory for both the test script and the resulting management pack JDF.BaseDistribution.xml. The files in this directory are always created and can be safely deleted.

### **2.6 tools**

At present holds only puttygen.exe which is used for generating the ssh key pairs.

### 3 Configuring and building the framework

This chapter will explain how you need to configure the JDF framework for your environment. In order to use the framework for your environment, you have to configure the settings and build a distribution management pack. The following need to be configured:

1. Disk location for the framework. The framework needs to be placed on disk. Select the location where you want the framework be placed. The default location will be set to  
%SystemDrive%\management\jdf.

At this moment it is not possible to place the framework on another disk. It will always be placed on the %SystemDrive% disk of the system.

2. Registry location for the framework. This is the location where the framework will keep its configuration after it has been registered. Default this location will be set to  
SOFTWARE\Company\jama\jdf

#### 3.1 Configuring the framework

The disk location for the framework needs to be set in the framework distribution script (jdfBaseDistribution.vbs).

**Figure 1 - jdfBaseDistribution.vbs**

```
#####  
##### User configurable values #####  
#####  
-----  
' Framework information. This information is required for correct functioning  
' of the framework.  
'  
' STR_BASE_DIR = Base directory to use for the location of the framework.  
' STR_JDF_DIR = Sub directory used for the location of the framework  
'  
' The files will be stored in this location:  
' %SystemDrive%\STR_BASE_DIR\STR_JDF_DIR  
'  
' STR_JDF_KEY = The value in the registry where all framework settings are  
' stored. This value MUST be equal to the one defined in the  
' framework file (jdf.wsc).  
'  
' Note: The directories can not contain whitespaces!!  
-----  
const STR_BASE_DIR = "management"  
const STR_JDF_DIR = "jdf"  
const STR_JDF_KEY = "SOFTWARE\Company\jama\jdf"  
#####  
##### End of user configurable values #####  
#####
```

---

**Note 1:** Whitespaces in the directory names is not supported.

---

The registry location for the framework needs to be set in the framework distribution script (jdfBaseDistribution.vbs), but also in the framework itself (jdf.wsc)

**Figure 2 - jdf.wsc**

```
#####
##### User configurable values #####
#####
-----
The following entries can be adjusted for your environment. Do not alter any
other values in this script.

STR_FRAMEWORK_VERSION - Version number of this framework. This number should
                        be upgraded with every change of the framework. The
                        version number is a 3 part string, containing only
                        numbers and seperated by dots (e.g. 1.0.0)

STR_JDF_KEY            - This is the registry key, where the framework stores
                        its configuration. If the key does not exist, it will
                        be created during registration.

STR_JDF_CUSTOMER_KEY  - The registry key were to find the customer id for the
                        computer running the framework.

STR_JDF_CUSTOMER_ID   - The registry value name for the customer id, which
                        is stored at STR_JDF_CUSTOMER_KEY.

STR_CUSTOMER_PREFIX   - The prefix used for customer accounts. The prefix
                        combined with the customer id, will be the generated
                        user account.

Note: If you do not have a registry key where to find the customer id, you
      MUST make sure you always specify the customer id when loading the
      framework. If not, initialization will fail.
-----

const STR_FRAMEWORK_VERSION = "0.9.24"
const STR_JDF_KEY           = "SOFTWARE\Company\jama\jdf"
const STR_JDF_CUSTOMER_KEY  = "SOFTWARE\Company\jama\OpsMgr2007\attributes"
const STR_JDF_CUSTOMER_ID   = "customer_id"
const STR_CUSTOMER_PREFIX   = "jdfCustomer_"

#####
##### End of user configurable values #####
#####
```

The framework file also holds some other configurable values (see also previous screenshot):

<b>STR_FRAMEWORK_VERSION</b>	This is the version number of the framework. Initialization and registration functions depend on this version number. Whenever you make a change to the framework, you should update this number accordingly. If you do not update the version a new version of the framework may not be distributed correctly.
<b>STR_JDF_CUSTOMER_KEY</b>	The framework is customer based. At this registry location the information about the customer can be found.
<b>STR_JDF_CUSTOMER_ID</b>	This is the registry value, which holds the name of the customer as known to the framework <sup>1</sup>
<b>STR_CUSTOMER_PREFIX</b>	<p>The prefix used to generate the account name to use when logging on to the secure copy server and to generate the name of the private key file to use.</p> <p>Account Name: STR_CUSTOMER_PREFIXSTR_CUSTOMER_ID</p> <p>Private Key File: STR_CUSTOMER_PREFIXSTR_CUSTOMER_ID.private</p> <hr/> <p><b>Note 2:</b> When generating and distributing the private key files, they must comply with the above naming schema.</p> <hr/>

If you ever need to change the version number of the framework, you should also change the version information in the registration part at the top of the jdf.wsc file:

```
<registration
  description = "Jama Distribution Framework"
  progid      = "jdf.wsc"
  version     = "0.9.24"
  classid    = "{17c3330c-981d-4544-a363-148bffd96a75}">
</registration>
```

Both version numbers should be equal. The build scripts will verify those numbers and generate an error if they are not correct.

---

**Note 3:** No other items should be changed when configuring the framework for your environment. All other changes to files are related to building the correct management pack.

---

---

<sup>1</sup> If a registry location is not available, you must give the customer name during initialization of the framework.

### 3.2 Changing build scripts

You need to configure the build scripts for your environment. Edit the file called "Generate\_JDF.BaseDistribution.XML.cmd":

Figure 3 - Generate\_JDF.BaseDistribution.XML.cmd

```
rem -----
rem Version of the framework and the generated MP.
rem -----
set STR_VERSION=0.9.24
set STR_MP_VERSION=0.9.24.0

rem -----
rem STR_TARGET_DIR
rem
rem This directory is used for testing purposes only. It should however be the
rem same directory as defined in the framework.
rem -----
set STR_TARGET_DIR=%SystemDrive%\management\jdf

rem -----
rem Variables
rem -----
set STR_PORT=5723
set STR_REMOTE_OTA_SERVER=172.24.11.20
set STR_REMOTE_PROD_SERVER=winsshd.production.local
```

<b>STR_VERSION</b>	Version of the framework. This version <b>must</b> be equal to both the version strings of the framework. The build script will check the version numbers to be equal.
<b>STR_MP_VERSION</b>	Version of the management pack being generated. If you need to update a sealed version of the management pack, make sure this version number is higher than the current management pack version (otherwise an upgrade will fail).
<b>STR_TARGET_DIR</b>	Target directory of the framework. This value is used for testing the generated files on your local computer. This target directory should be equal to the target directory given in <code>jdfBaseDistribution.vbs</code>
<b>STR_PORT</b>	The port number used by the secure copy server.
<b>STR_REMOTE_OTA_SERVER</b>	The secure copy server for the DTA build. This can be an IP address or an FQDN name. When using an FQDN name, it must be resolvable by the client.

<b>STR_REMOTE_PROD_SERVER</b>	The secure copy server for the production build. This can be an IP address or an FQDN name. When using an FQDN name, it must be resolvable by the client.
-------------------------------	---

### 3.3 Changing the jdf.jdp file

There is one special file, called jdf.jdp. This file is normally generated by the build scripts, but can also be manually created. This file needs to be placed in the "include\_text" directory (this is done default by the build script).

This file determines a handful of characteristics of the framework. The function block `:GenerateJdpFile` in the build script (`Generate_JDF.Base.Distribution.XML.cmd`) generates the requested file. If you ever need to change this file, you should either do it in this function block, or remove this function block to prevent the build script from overriding your own version.

```

:GenerateJdpFile
  echo ^^^>>>^^>>>^^>>>^^>>> Generating JDP file

  echo ;-----> %STR_JDP_FILE%
  echo ; This file is generated by Generate_JDF.BaseDistribution.XML.cmd>> %STR_JDP_FILE%
  echo ; Generated at: %TIME%0%DATE%>> %STR_JDP_FILE%
  echo ;----->>> %STR_JDP_FILE%
  echo [FRAMEWORK_VERSION]>> %STR_JDP_FILE%
  echo %STR_VERSION%>> %STR_JDP_FILE%
  echo.>> %STR_JDP_FILE%
  echo [VARS]>> %STR_JDP_FILE%

  rem -----
  rem Custom variables to add to the jdf.jdp file.
  rem -----
  echo UPLOAD      = /upload/>> %STR_JDP_FILE%
  echo DOWNLOAD    = /download/>> %STR_JDP_FILE%
  echo FILES       = /files/>> %STR_JDP_FILE%
  echo PACKAGES    = /packages/>> %STR_JDP_FILE%

  echo.>> %STR_JDP_FILE%
  echo [REGISTRATION]>> %STR_JDP_FILE%
  echo RemoteServer = %STR_REMOTE_SERVER%>> %STR_JDP_FILE%
  echo Port         = %STR_PORT%>> %STR_JDP_FILE%
  echo.>> %STR_JDP_FILE%
  echo [PACKAGE_FILES]>> %STR_JDP_FILE%
  echo zip.exe     = \jdf\>> %STR_JDP_FILE%
  echo.>> %STR_JDP_FILE%

goto :EOF

```

The jdf.jdp file is a windows ini file. The supported tags are listed in the next chapters.

#### 3.3.1 FRAMEWORK\_VERSION

This is a required tag. It should match the version number in the framework windows component script file.

Example:

```

[FRAMEWORK_VERSION]
1.0.0

```

### 3.3.2 VARS

This is an optional tag and defines the JDF variables. All variables you define in this sections are available for use in the framework. Variable values are stored in the registry, during registration. These values can be retrieved using the GetVar() method of the framework.

The following variables are default generated by the registration function:

JDF_DIR	JDF Installation path
JDF_IN_DIR	JDF packages in path
JDF_KEYFILE_DIR	JDF keyfiles path
JDF_OUT_DIR	JDF packages out path
JDF_PACKAGE_DIR	JDF base distribution path
JDF_TEMP_DIR	JDF temp path

Example:

```
[VARS]
UPLOAD      = /upload/
DOWNLOAD    = /download/
FILES       = /files/
PACKAGES    = /packages/
```

### 3.3.3 REGISTRATION

This is a required tag and is used for registering the framework. Two values must exist:

RemoteServer	The name or IP of the server running the secure copy service.
Port	Port number used by the secure copy service on the remote server.

Example:

```
[REGISTRATION]
RemoteServer = 172.24.11.20
Port         = 5723
```

### 3.3.4 PACKAGE\_FILES

A required tag. It defines the end location of the files in the package. Each file in the distribution package must be noted in this section. The format of this section is:

source = target

source	The name of the file in the package
target	Directory where the file will be placed after extraction.

Use \ for \$JDF\$ itself.

The following files are generated by the base distribution script or retrieved if missing and should not be part of the distribution package:

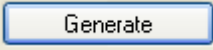
jdf.jdp	generated
pscp.exe	generated
jdfBaseDistribution.private	generated
unzip.exe	retrieved (we can't extract without)

Example:

```
[PACKAGE_FILES]
zip.exe = \jdf\
```

### 3.4 Default private key

The last thing to do is generating a private/public key pair for use with the distribution script. The distribution script uses the same mechanism as the framework to retrieve the initial required files.

Start puttygen.exe from the tools subdirectory and press the  button to generate an ssh key pair. You should use the default settings for generating the key pair:



Parameters

Type of key to generate:

SSH-1 (RSA)       SSH-2 RSA       SSH-2 DSA

Number of bits in a generated key:

- The private key must be called jdfBaseDistribution.private
- The private key cannot have a password phrase for protection.
- Finally you should place the private key file in the sub-directory “include\_text”

### 3.5 Building the distribution MP

When all values are set and the private key file is generated, you can generate a new version of the distribution management pack. Execute the following command:

```
Generate_JDF.BaseDistribution.XML.cmd FALSE OFF
```

The above command will generate the production version of the distribution management pack.

If everything runs as required, the output of the command will look like this:

```
>>>>>>>> Checking version numbers (0.9.24)
>>>>>>>> Clearing local target directory
>>>>>>>> Generating JDP file
>>>>>>>> Copying C:\jamaDistributionFramework\jdfBaseDistribution\jdf.wsc
>>>>>>>> Creating script argument string
>>>>>>>> Creating distribution package
    adding: zip.exe (172 bytes security) (deflated 52%)
>>>>>>>> Creating base distribution management pack
    jdf.jdp
    jdf.wsc
    jdfBaseDistribution.private
    pscp.exe
>>>>>>>> Generating files
>>>>>>>> Comparing text files
>>>>>>>> Comparing binary files

Done.
DON'T FORGET TO MAKE THE NEW PACKAGE AVAILABLE!
Press any key to continue . . . _
```

When the build is finished, two important files exist:

.\output\JDF.BaseDistribution.xml

.\BaseDistribution\jdf\_v0.9.24.zip

### 3.5.1 JDF.BaseDistribution.xml

This is the resulting management pack in .xml format. You can use this version for your environment or you can seal this management pack. If you need to seal the management pack, you will find all dependent management packs in the "mp" sub directory.

### 3.5.2 jdf\_v0.9.24.zip

This is the base distribution package generated by the script. This file must be available on the secure copy server. It must be reachable by the jdfBaseDistribution account at its /packages/ path.

In other words, the scp client must be able to copy "/packages/jdf\_v0.9.24.zip". You should make this file available before you import the management pack. Otherwise a lot of failures will be generated as the distribution script is not capable of retrieving the correct file.

---

**Note 4:** The framework will not work until this file is available on the secure copy server.

---

**Note 5:** When generating a framework with another version number, the filename will be adjusted by the scripts to comply with: jdf\_v<version>.zip.

---

## 4 Secure Copy Server

Before you can start with the framework, you must make sure that your secure copy server is up and running. Although not described in this document, the secure copy server is required for the framework to work. Without a working and accessible secure copy server, the framework is useless.

### 4.1 Required accounts

The framework always requires at least two accounts. One general account used for distributing the framework and general packages and files and one customer account which is only valid for that customer. For all customers you need to support, you should create a separate secure copy account. So at least two accounts are required:

jdfBaseDistribution	This is the account used for distributing the framework itself and for general available packages. This should be a read-only account.
jdfCustomer_<name>	This is the account for the first (or only) customer that you define when loading the framework. This account can have write access.

You should generate a private/public key pair for those two accounts.

### 4.2 Private keys

The private keys for all accounts used on a computer, must be made available in the keyfiles subdirectory on the computer with the SCOM agent. During initialization, the framework will check if the private key for the customer name given during initialization exists. If the key does not exist, the framework will fail to initialize.

#### 4.2.1 jdfBaseDistribution

This private key is part of the generated management pack. No separate distribution method is required. The distribution script will place this file on the correct location.

#### 4.2.2 jdfCustomer\_<name>

This private key is not part of the generated management pack and it is the responsibility of the user to get this key on the right location. You must make sure that these customer keys are only available at the right customer. Note that the distribution script will not remove any files from the keyfiles subdirectory, so it should be a onetime action, which could be done with SCOM ☺


Sharing the keys with different customers, will make it possible for customer A to read data retrieved from customer B. So make sure you don't mix customers.

We ourselves created a management pack, which would distribute a private key to all servers of a single customer. So for all customers we created a single management pack, only targeted at that customer (this prevents the mp to be distributed to other customers).

For more information about how to implement the secure copy server, see the information on our blog.






## 5 Importing the framework

After you have imported the management pack, the “JDF: Base distribution” management pack will be available in your management group.






 JDF: Base distribution

The following rules will be available in your management group:

### Type: Windows Computer (5)

 JDF: Base distribution. Files distributed successfully (event id 300)	Windows Computer	JDF: Base distribution
 JDF: Base distribution. Script failed to initialize (event id 200)	Windows Computer	JDF: Base distribution
 JDF: Base distribution. All required files are available (event id 100)	Windows Computer	JDF: Base distribution
 JDF: Base distribution. Failed to create at least one file (event id 400)	Windows Computer	JDF: Base distribution
 JDF: Base distribution	Windows Computer	JDF: Base distribution

The rules have these functions:

Rule name	Function	Enabled
 JDF: Base distribution	Distribution rule. This rule will default run once a day and if required will distribute the framework on the target computer.	No
 JDF: Base distribution. All required files are available (event id 100)	Alert rule. The framework files are already available on the correct location. No action is taken and no action is required.	No
 JDF: Base distribution. Script failed to initialize (event id 200)	Alert rule. The distribution script could not initialize. No files are distributed.	Yes
 JDF: Base distribution. Files distributed successfully (event id 300)	Alert rule. The framework is distributed successfully. No action is required.	No
 JDF: Base distribution. Failed to create at least one file (event id 400)	Alert rule. The distribution script failed to create at least one required file.	Yes

---

**Note 6:** The actual file generation rule is default disabled and should be enabled by creating an override (for a sealed MP) or changed to enabled (for a non sealed MP).

---

When the files are generated by the agents, we can start using the framework.

## 5.1 Possible errors

There are two rules that will generate an alert when an error is encountered during the distribution of the framework.

## 5.2 JDF: Base distribution. Script failed to initialize (event id 200)

 JDF: Base distribution. Script failed to initialize (event id 200)

### Summary

The script failed to initialize properly.

### Causes


Possible causes:

- An invalid framework directory is given.
- A system directory is part of the framework directory.
- Environment values can not be retrieved.
- A general scripting error occurred.

### Resolutions

To identify the exact problem, you can enable debugging to get debugging information about the distribution script and the framework. See the JDF documentation on how to enable debugging for the framework and the distribution rule.

## 5.3 JDF: Base distribution. Failed to create at least one file (event id 400)

 JDF: Base distribution. Failed to create at least one file (event id 400)

### Summary

The script was not able to create at least one of the required target files.

### Causes

The error code gives an indication of the error encountered by the rule.:

Error Code	Description
1	A required registry value could not be set.
2	A required registry key could not be created.
3	A registry object could not be created by the script.
4	A target file could not be created. The script writes the file on location \$PATH\$\
5	A write error occurred while writing the target file.
6	A target file could not be opened/created.
7	The source file with the target definitions could not be opened for reading.
10	The definition file was created, but could not be moved to its final location. The final destination is: \$PATH\$\jdf\
11	The pscp.exe file was created, but could not be moved to its final location. The final destination is: \$PATH\$\jdf\
12	The script failed to register the framework.
13	No framework object could be created.
14	The key file was created, but could not be moved to its final location. The final destination is: \$PATH\$\keyfiles\
15	The target directory (\$PATH\$) could not be cleaned.
16	The connection parameters could not be read.
17	The script could not retrieve the base package from the central secure copy server.
18	The script could not retrieve the required unzip.exe from the central secure copy server.
19	At least one file could not be extracted from the base package.

**Note:** \$PATH\$ is the value that can be found in the registry of the local server at and is default set to: %SystemDrive%\management\jdf. See the JDF documentation for more information about this registry setting.

## Resolutions

To identify the exact problem, you can enable debugging to get debugging information about the distribution script and the framework. See the JDF documentation on how to enable debugging for the framework and the distribution rule.

## 6 Using the framework

After the framework is distributed, you can start using the framework. In order to use the framework, it must be loaded first to be available in your script. The template.vbs file is an example on how to do this. After the framework is loaded, the running script can use the framework for creating, sending and retrieving files and packages.

### 6.1 Loading the framework

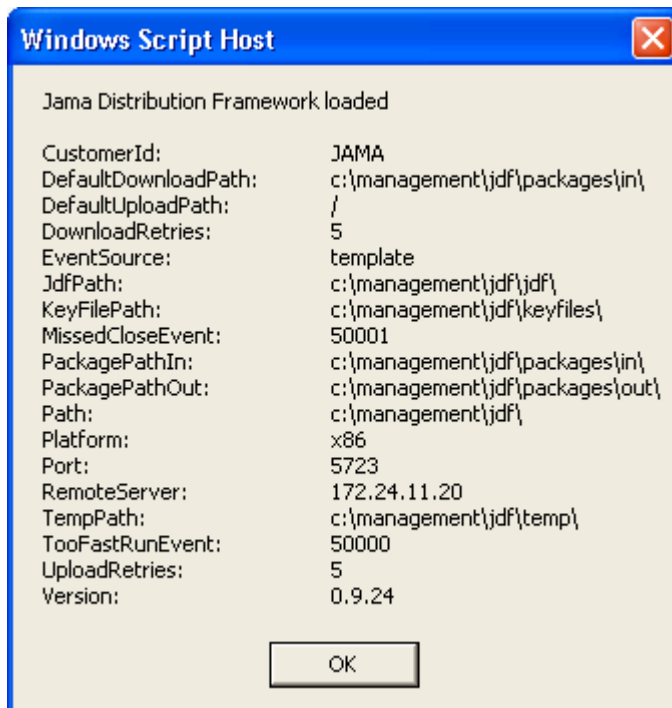
First you need to create a variable, which will hold the object for the framework.

```
dim jdf
```

Now use the supplied function in the template.vbs file to load the framework. After the framework is loaded successfully, you can start using it.

```
if(not jdfLoadFramework(jdf, null, null, null, null, null)) then
    wscript.echo "Jama Distribution Framework not loaded."
    wscript.quit(1)
else
    wscript.echo "Jama Distribution Framework loaded" & vbNewLine & vbNewLine & jdf.GetInfoString(wscript.fullname)
    '-----
    ' Your code here!
    '-----
end if
```

From the template file, when the framework loaded successfully, you will see output similar to this:



---

#### Note 7: Common errors are:


No connection to the secure copy server during distribution or the jdf\_v<vesion>.zip file is missing (registration will fail).



















No customer private key to be found in the keyfiles directory or registry.

---

## 6.2 Debugging

When you need to troubleshoot what is happening with the framework, you can enable debugging messages in the framework by adding a debug string value to the "HKEY\_LOCAL\_MACHINE\SOFTWARE\Company\jama\jdf" registry key.

When setting this value to "true" ( debug REG\_SZ true), the framework will log a lot of extra information into the application eventlog:

Type	Date	Time	Source	Category	Event ID
 Information	3/30/2011	13:43:18	template	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900
 Information	3/30/2011	13:43:18	Jama Distribution Framework	None	900

## 7 Framework reference

This chapter describes the available methods and properties of the framework.

### 7.1 Properties

The following properties are defined in the framework.

Name	Type	Description
CustomerId	Read	The name of the customer as found in the registry or given during initialization.
DefaultUploadPath	Read / Write	The default upload path to use when no upload path is given.
DefaultDownloadPath	Read / Write	The default download path to use when no download path is given.
DownloadRetries	Read / Write	Number of retries to perform when a download fails. After this number of retries has passed, the framework will give up.
EventSource	Read	The name of the event source name used by the framework to log entries in the application eventlog.
JdfPath	Read	Full path to the jdf sub directory. This sub directory holds all default distributed files, with the exception of the jdf.wsc file.
KeyFilePath	Read	Full path to the key file sub directory. This directory should hold all private keys defined for the customer.
MissedCloseEvent	Read / Write	Number which can be returned if a command finished too fast on the computer. This number will be returned if we missed a process close event.
PackageName	Read	Name of the current active package.
PackagePathIn	Read / Write	Full path of the directory where to store a downloaded package.
PackagePathOut	Read / Write	Full path of the directory where to store a new package.
Path	Read	Full path to the base directory (default: %SystemDrive%\management\jdf)
Platform	Read	x86, x64 or ia64
Port	Read	Port number of the remote secure copy server.
RemoteServer	Read	FQDN or IP address of the remote secure copy server.
TempPath	Read / Write	Full path to the temp directory for the framework.
TooFastRunEvent	Read / Write	Number which can be returned if a command finished too fast on the computer. This number will be returned if we cannot find the process id of a process after we have successfully created the process (for really fast processes).
UploadRetries	Read / Write	Number of retries to perform when an upload fails. After this number of retries has passed, the framework will give up.
Version	Read	Version of the framework.

## 7.2 Methods

The next chapters will describe all public available methods of jdf.wsc. All examples given, assume that the framework is initialized and the framework can be used with an object variable called jdf and method return values are stored in bResult (booleans), iResult (integers) or strResult (strings).

---

**Note 8:** Most methods will return false, -1 or "" if the framework is not initialized.

---

### 7.2.1 AddToPackage

AddToPackage(strFile)

Use: Adds a file to the current package.  
Input: strFile string - full path to the file to add.  
Returns: integer 0 on success, any other value otherwise.

Example: `iResult = jdf.AddToPackage("d:\test\file7.txt")`

### 7.2.2 CreateDirectory

CreateDirectory(strPath)

Use: Creates the target directory.  
Input: strPath string - Full path of the target directory.  
Returns: Boolean TRUE - Success.  
FALSE - An error occurred.

Example: `bResult = jdf.CreateDirectory("e:\files\logging\week\12")`

### 7.2.3 CreatePackage

CreatePackage(strPackage)

Use : Creates a new package.  
Input: strPackageName string - The name of the package  
Returns: integer 0 on success, 1 otherwise.  
Note(s): Only one package can be active at the same time. You cannot access an existing package.

Example: `iResult = jdf.CreatePackage("ReportsWeek14")`

## 7.2.4 DeletePackage

DeletePackage()

Use: Deletes the current package.

Input: ---

Returns: integer 0 on success or -3 if no package is available.

Example: `iResult = jdf.DeletePackage()`

## 7.2.5 DownloadPackage

DownloadPackage(strPackage, strDestination)

Use: Downloads a package from the central server.

Input: strPackage string - Full path to the package.

strDestination string - Full path to the destination directory or null

Returns: integer 0 on success

1 if the package path contains a backslash.

-3 if the package path does not contain a forward slash or does not end with .zip

All other values indicate an error condition from copying.

Note(s): If strDestination is set to null or an empty string (""), the default download location will be used.

Example: `iResult = jdf.DownloadPackage("/packages/jdfTest.zip", null)`

## 7.2.6 ExpandString

ExpandString(strString)

Use: Expands all variables (both JDF and Windows) in a string.

Input: strString string - String to parse.

Returns: string Expanded string

Note(s): Not all windows variables are supported. Always test the use of windows variables.

Example: `strResult = jdf.ExpandString("This is the jdf dir: $JDF_DIR$")`

## 7.2.7 ExtractFromPackage

ExtractFromPackage(strFile, strDestination)

Use: Extracts a single file from the current package.  
Input: strFile string - File name as available in the package.  
strDestination string - Destination directory of the file to extract.  
Returns: integer 0 on success. Any other value indicates an error.

Example: `iResult = jdf.ExtractFromPackage("zip.exe", "c:\test\phase1")`

## 7.2.8 ExtractPackage

ExtractPackage(strDestination)

Use: Extract all files from the current package.  
Input: strDestination string - Destination directory for all files.  
Returns: integer 0 on success. Any other value indicates an error.  
Note(s): If any directory structure exists in the package, it will be kept in the new destination directory.

Example: `iResult = jdf.ExtractPackage("c:\test\phase2")`

## 7.2.9 GetFile

GetFile(strUser, strSource, strTarget)

Use: Downloads a single file.  
Input: strUserName string - Username used to connect.  
strSource string - Source file  
strTarget string - Target file  
Returns: integer exit code of pscp.exe (0 = success, <> 0 = failure)  
Note(s): No retries are performed. This must be handled by the calling function.

A private key file must exist for the user name given.

You must give the full path to both the source and target files.

Example: `iResult = jdf.GetFile("jdfBaseDistribution",  
"/packages/jdf_v1.0.0.zip", "c:\test\phase3\jdf_v1.0.0.zip")`

## 7.2.10 GetInfoString

GetInfoString(strScriptHost)

Use: Returns a formatted string with jdf information.  
Input: strScriptHost string - Full name of the scripting host.  
Returns: string Formatted string with jdf information.

Example: `strResult = jdf.GetInfoString(wscript.fullname)`

## 7.2.11 GetVar

GetVar(strVar)

Use: Retrieves a JDF variable from the registry  
Input: strVar string - Name of the variable  
Returns: string Expanded variable or "" if not found.  
Note(s): The name should **not** be enclosed by the dollar sign (\$). Normally you should use ExpandString(), which can expand a whole string instead of retrieving a single variable.

Example: `strResult = jdf.GetVar("JDF_TEMP_DIR")`

## 7.2.12 InitializeFramework

InitializeFramework( strMinimumVersion, bForceVersion, strCallingScript,  
strCustomerId, strDefaultUploadPath)

Use: Initializes the framework.

Input: strMinimumVersion string - Minimum framework version.  
bForceVersion bool - true or false.  
strCallingScript string - Name of the script calling this function.

strCustomerId string - Customer id or null.  
strDefaultUploadPath string - Default upload path.  
Returns: Boolean TRUE - Initialization succeeded.  
FALSE - Initialization failed.

Note(s): strMinimumVersion = null  
Any version of the framework will be accepted. The value of  
strForceVersion will be ignored.

strMinimumVersion = "x.x.x"  
Integer values, seperated by dots, e.g. : "3.5.11"  
The given version is a MINIMUM version required for the framework.

bForceVersion = true  
The framework version must EXACTLY match with the given version  
number.

strCustomerId = null  
The customer id will be retrieved from the registry.

strDefaultUploadPath = null  
The default upload path will be set to the root: "/". This argument can  
hold JDF variables (both default and custom provided in the jdf.jdp  
file).

Example: 

```
bResult = jdf.InitializeFramework(null, false, wscript.scriptname,  
null, null)
```

## 7.2.13 PutFile

PutFile(strUser, strFile, strDestination)

Use: Uploads a single file.

Input: strUserName string - Username used to connect.

strSource string - Source file

strTarget string - Target file

Returns: integer exit code of pscp.exe (0 = success, <> 0 = failure)

Note(s): No retries are performed. This must be handled by the calling function.

A private key file must exist for the user name given.

You must give the full path to both the source and target files.

Example: 

```
iResult = jdf.PutFile("jdfBaseDistribution",
"c:\test\phase3\jdf_v1.0.0.zip", "/packages/jdf_v1.0.0.zip")
```

## 7.2.14 RegisterFramework

RegisterFramework(strFrameworkFile, strJdpFile, strPackageInDir, strPackageOutDir, strTempDir)

Use: Register this framework file in the registry.

Input: strFrameworkFile string - Full path to the framework file.

strJdpFile string - Full path to the jdf.jdp file.

strPackageInDir string - Full path to the package in directory.

strPackageOutDir string - Full path to the package out directory.

strTempDir string - Full path to the temp directory.

Returns: Boolean TRUE - Registration succeeded.

FALSE - Registration failed.

Note(s): You cannot register an initialized framework.

If the value null or "" is given for strPackageInDir, strPackageOutDir or strTempDir, the registration function will keep any existing value in the registry or use the default if none was found.

Normally the registration is done by the distribution script and you should not have to use this function.

Example: 

```
bResult = jdf.RegisterFramework( "c:\test\jdf\jdf.wsc",
"c:\test\jdf\jdf.jdp",
"", "", "")
```

## 7.2.15 ScheduleTaskIn

ScheduleTaskIn(strCommand, iMinutesToWait)

Use: Schedules a task x minutes in the future, but on the current day.

Input: strCommand string - command to execute.  
iMinutesToWait integer - number of minutes to schedule the task into the future.

Returns: boolean TRUE - Task is scheduled  
FALSE - Task is not scheduled.

Note(s): This function can only schedule a task on the current day. So if the current time + iMinutesToWait exceeds 23:59, the task will not be scheduled.

Example: `bResult = jdf.ScheduleTaskIn("c:\upgrade\upgrade.exe /silent", 120)`

## 7.2.16 UnRegisterFramework

UnRegisterFramework()

Use: Unregisters the framework.

Returns: bool TRUE - succeeded.  
FALSE - failed.

Example: `bResult = jdf.UnRegisterFramework()`

## 7.2.17 UploadPackage

UploadPackage(strDestination)

Use: Upload the current package to the remote server.

Input: strDestination string - Destination folder to copy the package to.

Returns: integer 0 on success, any other value otherwise.

Note(s): The destination folder must exist before starting the upload.

After a successful upload, the package will be deleted!

Example: `iResult = jdf.UploadPackage("$UPLOAD$")`

## 8 Template script file template.vbs

The framework has a file called template.vbs. This file is a template file you could use to load and use the framework.

Before you start using this template file, you should adjust the registry location for the framework settings (see red square)

```
function jdfLoadFramework(byref objJDF, byval bInitialize, byval strVersion, byval bForceVersi
    dim bResult, fso, strFrameworkFile, objReg, strResult

    bResult = false
    strResult = ""
    on error resume next
        err.clear
        set objReg=GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\default:StdRe
        if(err.number = 0) then
            objReg.GetStringValue &H80000002, "SOFTWARE\Company\jama\jdf", "path", strResult
            if((err.number = 0) and (strResult <> "") and (not isnull(strResult))) then
                strFrameworkFile = strResult & "jdf.wsc"
                set fso = CreateObject("Scripting.FileSystemObject")
                if(err.number = 0) then
                    if(fso.FileExists(strFrameworkFile)) then
                        set objJDF = GetObject("script:" & strFrameworkFile)
```

After you have changed this value to your correct value, you can start using the template file.