FUJITSU Software
Interstage Studio

J Business Kit User's Guide

B1WD-3187-01ENZ0(00)
November 2013

Preface

Purpose of this Document

J Business Kit (hereinafter abbreviated as JBK) is an application system development kit containing application support libraries, application support tools, and operation support tools, all of which are useful for writing a variety of applications in Java.

J Business Kit is composed of the JBK Development Kit, which is an environment for Java program development, and the JBK Runtime Environment, which is an environment for Java program execution.

This manual explains how to use JBK to develop programs in the JBK Development Kit and provides notes on using JBK. It also provides other information, such as information on the JBK Runtime Environment.

Structure of This Document

The structure of this manual is as follows:

- Chapter 1 Overview of JBK

  This section gives an overview of JBK.

- Chapter 2 JBK Plugin

  This section gives an JBK Plugin.

- Chapter 3 GUI Library

  This section gives an GUI Library.

- Chapter 4 JBK Download Installer

  This section gives an JBK Download Installer.

- Chapter 5 Notes

  This section provides common notes on use of JBK.

Trademarks

Microsoft, Active Directory, ActiveX, Excel, Internet Explorer, MS-DOS, MSDN, Visual Basic, Visual C++, Visual Studio, Windows, Windows NT, Windows Server, Win32 are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
Oracle and Java are registered trademarks of Oracle and/or its affiliates.
Other company and product names in this documentation are trademarks or registered trademarks of their respective owners.

The symbols (R) and TM are omitted throughout this document.

October 2013

Revision history
First Edition (November 2012)
Second Edition (November 2013)

Contents

Chapter 1 Overview of JBK

JBK provides libraries and tools to reuse your legacy code when developing Java applications.

1.1 JBK

JBK includes the JBK Development Kit, which is an environment for Java program development, and the JBK Runtime Environment, which is an environment for Java program execution.

You can develop Java programs on a machine on which the JBK Development Kit is installed, and you can execute Java program on a machine on which the JBK Runtime Environment is installed.

JBK Development Kit

The JBK Development Kit is an environment for developing Java programs that are to be executed and operated in the JBK Runtime Environment. JBK includes GUI and sorting class libraries.

- Products

  Interstage Studio Standard-J Edition

- Function

  - GUI Library
  - JBK Plugin
  - JBK Download Installer

JBK Runtime Environment

The JBK Runtime Environment is an environment in which you can run Java programs developed with the JBK Development Kit. JBK includes the runtime environment which helps you run Java application.

- Products

  Interstage Studio Client Runtime

- Function

  - GUI Library
  - JBK Plugin

1.2 Target Versions of JavaSE

JBK requires one of the following versions of JavaSE:

- JBK V6.3

  JBK works with JavaSE 6 included in this product.

- JBK V7.1

  JBK works with JavaSE 7 included in this product.

JavaSE included in this product

The JavaSE included in this product is based on the one licensed and provided by Oracle Corporation, and it has feature enhancement and bug fixes by Fujitsu.

JBK works with only the JavaSE included in this product. The JavaSE provided by Oracle Corporation will not be supposed to work with JBK.

1.3 Installation composition

Folder tree

The folder tree of the JBK Development Kit is as follows:

```
(Product-installation-folder)
   +-IDE\1101\JBK
       +-bin       ... Contains Windows components.
       +-classes   ... Contains jar files for libraries and tools.
       +-gui6      ... Contains jar files for GUI libraries(for JRE6).
       +-gui7      ... Contains jar files for GUI libraries(for JRE7).
       +-examples  ... Contains sample programs of libraries.
```

Environment Variables

The following environment variables must be specified in order to run JBK Development Kit:

- CLASSPATH:Product-installation-folder\IDE\1101\JBK\gui6\jbkgui.jar

  These environment variables are set in the registry by the product installer.
  CLASSPATH is set in the environment of the system side.

Configuration of the java.policy File

The installer adds the following information to (Product-installation-folder)\IDE\jre\lib\security\java.policy:

```
grant codeBase "file:/(Product-installation-folder)/IDE/1101/JBK/gui6/jbkgui.jar " {
    permission java.security.AllPermission;
};
grant codeBase "file:/( Product-installation-folder) /IDE/1101/JBK/examples/-" {
    permission java.security.AllPermission;
    };
```

Chapter 2 JBK Plugin

This section describes JBK Plugin.

2.1 Overview of JBK Plugin

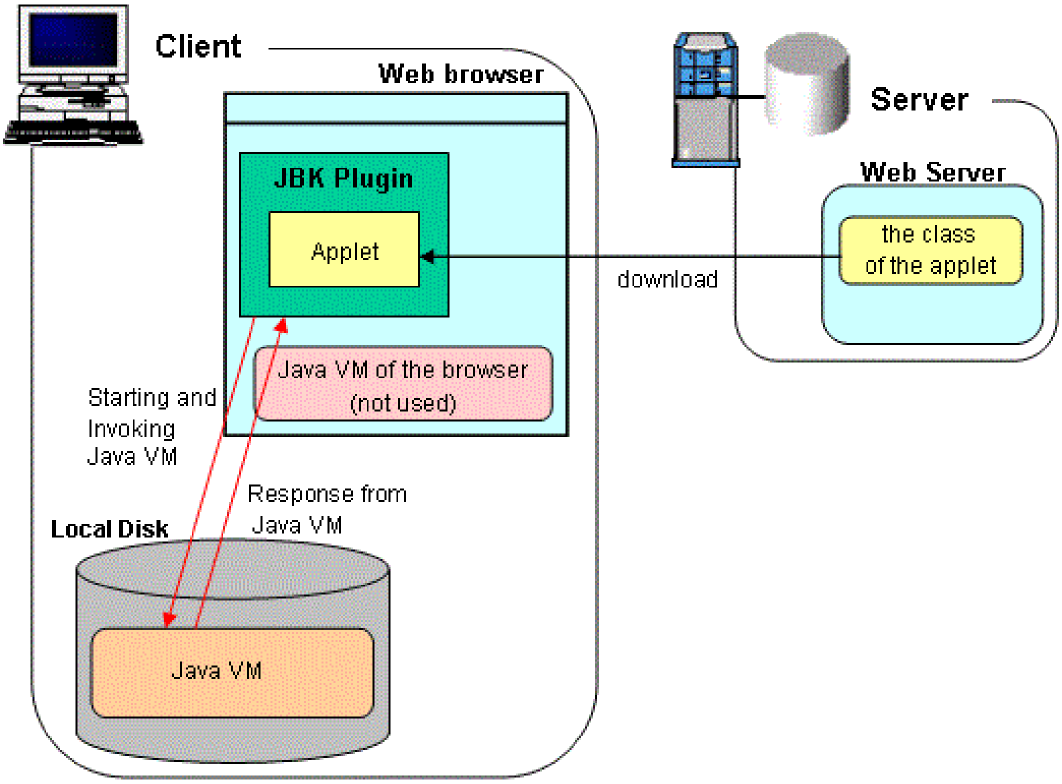
JBK Plugin provides functions to run an applet on a Web browser.

To use JavaSE included in this product on a Web browser, either JBK Plugin or Java Plug-in must be installed.

JBK Plugin is recommended.

By using JBK Plugin, you can run an applet with JavaSE included in this product rather than the Java VM provided by the browser. It means you do not need to bother the Java VM version of your browser or the incompatibility of the behavior.

Figure 2.1 Running an applet with JBK Plugin



Related Software

JBK Plugin works with the following Web browsers:

- Internet Explorer 7 - 10

  All of them are Windows products. Check your browser's documentation for the Windows operating systems on which your browser will run.

Note

JBK Plugins do not work on the Windows UI version of Internet Explorer in Windows 8.

JBK Plugins do not work on the Windows UI version of Internet Explorer in Windows 8.

If you try to open an HTML page in which a JBK plugin has been embedded in the Windows UI version of Internet Explorer, nothing will be displayed for the relevant parts.

You must change to the desktop version of Internet Explorer 10 by performing the following action:

1. On the apps bar, click the [Page Tools] button, and then select [View on the desktop].

2.1.1 Java Plug-In of Oracle Corporation

Java Plug-In of Oracle Corporation is also a tool that supports the execution of an applet on a Web browser.

Using Together with Java Plug-In

- Java Plug-In and JBK Plugin cannot mutually be used together in a single browser process. If an attempt is made to use both of them, the two may malfunction. Use just any of the two.

Difference between JBK Plugin and Java Plug-In of Oracle Corporation

JBK Plugin, however, has the following functionalities that Java Plug-In does not have:

- Capability to specify additional policy files

  JavaSE provides new security setting files called policy files. Because JBK Plugin allows you to use additional policy files, you can use JBK Plugin policy files to switch to JavaSE or reinstall JavaSE and continue to use your settings without setting up new default policy files.

  For information about how to set a security level, see '2.3.3 Security Setting'.

- Startup of a Java VM in advance

  To initially execute an applet in a browser, you need to start the Java VM. In general, Java VM needs from several to tens of seconds to start up. In this startup sequence, the browser may be unable to accept any user instructions.

  When you use JBK Plugin and you code HTML files appropriately, you can start just the Java VM without running any applets. Because the Java VM starts in the background, the browser is never in a state in which it is unable to accept user instructions during Java VM startup. This functionality enables you to handle HTML files as follows:

  a. Create an HTML file dedicated to starting the Java VM (that is, a file that is different from HTML files in which applets are executed). Write the HTML file so that, before an applet is executed, the file handles permitted preliminary operations (for example, displays the application's title page, provides the latest information, or makes a request to the user) while the Java VM is starting up.

  b. Start the browser, then load the HTML file created in step a. The browser handles the preliminary operations coded in the HTML file in its window. Since, during operation, JBK Plugin starts the Java VM in the background, the Java VM startup sequence does not interrupt any operations on the browser window.

  c. After completion of the preliminary operations, load an HTML file in which applets are to be executed. When the Java VM has been completely started by JBK Plugin, the Java VM immediately starts downloading the applets. This method of starting the Java VM does not require users to wait until Java VM startup has been completed.

     For information about how to initially start only the Java VM, see '2.3.9 Starting a Java VM in Advance'.

- Reporting of applet activation and deactivation

  You can place several applets in one browser window and allow them to run. If a window is active, some applets in the window are active and some are not.

  JBK Plugin can identify active and inactive applets using its own functionality and notify applets when they are to be active or inactive. This functionality is useful for switching the operation environment by applet in a system that includes multiple applets.

  For example, you can set a pop-up window or dialog box and make an applet active when the window or box is closed.

  For information about how to use notifications reporting whether applets are active or inactive, see '2.3.10 Notifying Applet Activation and Inactivation'.

- Confirmation of applet termination

  JBK Plugin can ask applets running in a window whether they can be terminated or not when the user is closing the window. When applets associated with this feature receive an inquiry from JBK Plugin, the applets can inform JBK Plugin of whether they can be terminated or not.

  For information about how to ask an applet about termination, see '2.3.11 Confirming Applet Termination'.

- Property file switching

  You can operate JBK Plugin with different property files on one client by preparing multiple JBK Plugin property files and calling a different JBK Plugin property file in an HTML file.

  With selectable JBK Plugin property files, for example, you can load property files with different JREs specified into separate browsers to run applets with different VM versions.

  For information about how to switch between property files, see '2.3.12 Switching the Property Files'.

If you need these functionalities, we recommend you to use JBK Plugin.

See

For details on Java Plug-in, refer to "JAVA PLUG-IN TECHNOLOGY" (http://www.oracle.com/technetwork/java/index-jsp-141438.html) from Oracle Corporation.

2.2 Preparation

This section describes preparations for using JBK Plugin.

2.2.1 JBK Plugin Setup

This section describes how to set up an environment for using JBK Plugin.

To execute an applet with JBK Plugin, check the following points:

Software Requirements

JBK Plugin requires the following software products:

- Web browser
- JDK or JRE
- JBK runtime

For the method of setting an environment for executing each product, see the manual included in the product.

Locations of the Applet Class and the HTML File

The applet class and HTML file for executing the applet can be stored in the same way as when they are stored to execute a usual applet by a browser. Choose either of the following depending on the method of applet execution:

- Store the applet class and the HTML file in the Web server. The applet is downloaded into the client machine and executed.
- Store the applet class and the HTML file in the client machine. The applet is executed in the local environment of the client machine.

For executing an applet with JBK Plugin, an HTML file must have a tag for JBK Plugin in it, instead of a usual <APPLET> tag. See '2.2.3 How to Create an HTML File for JBK Plugin' for the detail.

Setting of a browser

The following settings are needed according to a browser used.

Control module of JBK Plugin

Up to now,the single thread model made control module (f5cxwpie.ocx) of the JBK Plugin. However, the problem that the applet is not correctly displayed when it opens an empty window when using it in the environment since Internet Explorer 5 and the applet is displayed there might occur. Therefore, f5cxwpie.ocx was changed to the apartment thread model.

Please do the following work when you use f5cxwpie.ocx of the single thread model:

Please replace "[JBK installation folder]\bin\single\f5cxwpie.ocx" with "[JBK installation folder]\bin\f5cxwpie.ocx".(Please take the backup in "JBK installation folder \bin\f5cxwpie.ocx" to restore it to the apartment thread model.)

Please execute the following command.

```
regsvr32 /u [JBK installation folder]\bin\f5cxwpie.ocx
regsvr32 [JBK installation folder]\bin\f5cxwpie.ocx
```

Please work by using the backup of "JBK installation folder \bin\f5cxwpie.ocx" according to the same procedure when you restore f5cxwpie.ocx from the single thread model to the apartment thread model.

Setting Environment Variables

JBK plugins use the following environment variables on the client machine when in use:

If the setting has already been made, the setting need not be changed.

CLASSPATH

If some classes are installed in the client machine and the applet uses the classes, set the path to the classes into the CLASSPATH. Unlike Java Plug-in, JBK Plugin by default uses the CLASSPATH environment variable when executing applets.

Class paths when Java VM used

The Java VM uses two class paths: a boot class path required for Java VM startup and a user class path for execution of applets. JBK Plugin internally sets a specific boot class path at runtime.

For information about the specific boot class path, see the following:

1. JAR file containing the JBK Plugin classes, i.e.,
   "(JBK installation folder)\classes\jbkplugin2.jar" or "(JBK installation folder)\classes\jbkplugin4.jar"

2. (when "-Xbootclasspath" is specified for the Java VM startup option)
   Boot class path specified for the startup option

   (when "-Xbootclasspath" is not specified for the Java VM startup option)
   Default boot class path of the JavaSE used for the Java VM

   The JavaSE default boot class path includes the paths to "rt.jar", "i18n.jar", "jce.jar", "jsse.jar" and "charsets.jar" used by JavaSE.

When the user class path is used,

- (when "-classpath" or "-cp" is specified for the Java VM startup option)
  the class path specified in the startup option, or

- (when "-classpath" or "-cp" is not specified for the Java VM startup option)
  the class path specified in the CLASSPATH environment variable

is used. When Java Plug-in is used or if the "CLASSPATH" environment variable does not need to be used, setting the following property in the jbkplugin.properties file disables the CLASSPATH environment variable settings:

```
jbk.plugin.sw.classpath.use_env=<use-CLASSPATH-environment-variable>
```

In <use-CLASSPATH-environment-variable>, specify either of the following:

true

When the CLASSPATH environment variable settings will be used to start the Java VM.

false

When the CLASSPATH environment variable settings will not be used to start the Java VM.

Note that the jbk.plugin.sw.classpath.use_env default value is true (use the environment variable).

Note

In the "-Xbootclasspath" startup option, do not include any boot class paths of JavaSEs whose version is different from the one that you intend to use.

If you do so, an incorrect version may be called for Java VM execution, and an incorrect operation or Java VM execution error may result. In addition, downloading classes existing in the CLASSPATH environment variable for applet execution from a Web server may cause a security exception. To guard against such errors, set the class path by adding the -classpath option to the Java VM startup option or set the property above to "false."

2.2.2 Getting Started with JBK Plugin

The following steps describe how to execute an applet on a browser by using JBK Plugin:

1. Create an applet class.

   Follow the usual procedure in creating an applet class.

2. Create an HTML file for executing an applet.

   To execute an applet by JBK Plugin, the HTML file for the applet needs a special tag for JBK Plugin in it, instead of a usual <APPLET> tag. See '2.2.3 How to Create an HTML File for JBK Plugin' for the detail.

3. Store the applet class and the HTML file in a Web server.

   Follow the usual storing procedure.

4. Start the browser on a client machine, and load the HTML file from the Web server. To load the HTML file, specify the URL indicating its storage location in the browser.

   JBK Plugin automatically starts when the browser reads the HTML file with tags for JBK Plugin. JBK Plugin then downloads applet classes, and it executes applets in the browser by using the Java VM that is installed in another folder rather than the Java VM provided by the browser.

2.2.3 How to Create an HTML File for JBK Plugin

This section describes how to create an HTML file for using JBK Plugin.

To execute an applet in a browser, a <APPLET> tag is used in an HTML file. To execute the applet in the browser with JBK Plugin, however, a tag for JBK Plugin is used instead of the <APPLET> tag. Create an HTML file for JBK Plugin, as described below.

For the method of specifying an applet with the <APPLET> tag, see a JDK document. For the attribute of the <APPLET> tag that can be used with JBK Plugin, see 'Supported Attributes in the <APPLET> Tag' below.

Creating an HTML file for Internet Explorer

To use JBK Plugin on Internet Explorer, create an HTML file for executing an applet, following these steps:

1. Use the <OBJECT>...</OBJECT> tags instead of the <APPLET>...</APPLET> tags.

2. Specify "CLSID:BEA62964-C40B-11D1-AACA-00A0C9216A67" as the CLASSID attribute of the <OBJECT> tag. With this setting, Internet Explorer automatically executes JBK Plugin.

3. Code the following <PARAM> tag between the <OBJECT>...</OBJECT> tags.

   ```
   <PARAM NAME="TYPE" VALUE="application/x-JBK-Plugin">
   ```

4. Code the following attributes that would be specified in the <APPLET> tag, directly into the <OBJECT> tag.

   - WIDTH, HEIGHT (required attributes)

   - ALIGN, HSPACE, VSPACE (optional attributes)

5. Code the other attributes that would be specified in the <APPLET> tag between <OBJECT>...</OBJECT>, using the <PARAM> tag. Code this <PARAM> tag in this format: <PARAM NAME="attribute-name" VALUE="attribute-value">.

6. Code the parameters to be passed to the applet (parameters that would be specified by the <PARAM> tag between <APPLET>...</APPLET>) directly between <OBJECT>...</OBJECT>.

The shortcut keys of a Web browser are disabled when an applet has the focus.

When [Ctrl]+[N] is pressed to open a new window from an Internet Explorer window that contains an applet, the new window may not accept shortcut keys because its applet has the focus.

To avoid focusing on the applet, specify the event handler onload="window.focus()" in the <BODY> tag. To inhibit focus movement by tab to the applet, specify the property "tabIndex=-1" in the <OBJECT> tag.

Example

Sample coding of an HTML file for Internet Explorer

To use JBK Plugin with Internet Explorer, the following coding of the <APPLET> tag is changed as shown.

- Sample coding of the <APPLET> tag (when JBK Plugin is not used)

```
<APPLET NAME="sample" CODE="Sample.class" ARCHIVE="sample.jar" WIDTH=100 HEIGHT=100>
  <PARAM NAME="color" VALUE="blue">
  <PARAM NAME="useDefault" VALUE="true">
</APPLET>
```

- Sample coding for using JBK Plugin (for Internet Explorer)

```
<OBJECT CLASSID="CLSID:BEA62964-C40B-11D1-AACA-00A0C9216A67"
  WIDTH=100 HEIGHT=100>
  <PARAM NAME="TYPE" VALUE="application/x-JBK-Plugin">
  <PARAM NAME="NAME" VALUE="sample">
  <PARAM NAME="CODE" VALUE="Sample.class">
  <PARAM NAME="ARCHIVE" VALUE="sample.jar">
  <PARAM NAME="color" VALUE="blue">
  <PARAM NAME="useDefault" VALUE="true">
</OBJECT>
```

Using Hidden Applets

There are two methods to hide the <object> element in HTML.

Method 1

Use width=0,height=0 to create an object Tag.

In this case, the window disappears, but sometimes the window can get focus when the applet starts, and sometimes the focus can be set on the applet by Tab switch(Tab pressing) on HTML.

Therefore, please specify the "tabindex=-1" in the object, to prevent the focus from moving to the window.

Method 2

Prepare a <div>Tab that includes an <object>Tab and set its display style as hidden (display=none or visibility=hidden).

Note that JBK Plugin's control cannot be created and the applet cannot be started if the display style is hidden at the beginning. Therefore, make the display style visible when creating it, and set it to hidden after creation.

Note

Sometimes the applet cannot be started if the applet is hidden at startup.

The Applet cannot be started under the following conditions.

- The <object>Tab is inserted through the script; and
- The display style is set to hidden in the same event handler.

The JBK Plugin creates an applet after the window size is confirmed. However, under the above conditions, the applet is hidden before the object's window size is notified, which leads to a failure in starting the creation of the applet's child window. In this case, to set the display style as hidden, use the setTimeout() function of script as the next event.

Supported Attributes in the <APPLET> Tag

For JBK Plugin, you can use the following attributes of the <APPLET> tag.

- Required attributes

  These attributes must be specified to execute an applet:

Attribute name	Description
CODE	Specifies an applet class name.
WIDTH	Specifies an applet width. *Do not specify the WIDTH attribute in the style attribute
HEIGHT	Specifies an applet height. *Do not specify the HEIGHT attribute in the style attribute.

- Optional attributes

  These attributes are specified if needed. The optional attributes can be omitted.

Attribute name	Description
ALIGN	Specifies the location of an applet. The following can be specified. - top Aligns the top end of the applet with the top end of text. - bottom Aligns the bottom end of the applet with the bottom end of text. - left Aligns the end of the applet with the left end of the browser and displays text to the right. - right Aligns the end of the applet with the right end of the browser and displays text to the left.
ARCHIVE	Specifies a JAR file including the applet class.
CODEBASE	Specifies the URL from which the applet class is downloaded. This attribute is specified when the HTML file and the applet class are downloaded from different locations.
HSPACE	Specifies the horizontal space of the applet.
NAME	Specifies the name of the applet.
VSPACE	Specifies the vertical space of the applet.

Note

Uppercase and lowercase letters in attribute names and parameter names

- When coding the attributes of the <APPLET> tag described above in an HTML file for JBK Plugin, the attribute names can be coded in uppercase or lowercase in the <APPLET> tag. For instance, the attribute name for specifying an applet class name can be "CODE" or "code."
- When an applet is executed with JBK Plugin, the parameter names passed to the applet are case-insensitive.

Point

Compatibility with an HTML file for Java Plug-in

An HTML file for JBK Plugin has compatibility with that for Java Plug-in of Oracle Corporation, however, except a few differences. An HTML file for using Java Plug-in can be used with JBK Plugin after the following changes were made:

- Change the CLASSID attribute of the <OBJECT> tag coded in the HTML file for using Java Plug-in to "CLSID:BEA62964-C40B-11D1-AACA-00A0C9216A67."
- If the <OBJECT> tag contains the coding of the CODEBASE attribute as shown below, eliminate the coding.

```
codebase="plugin-download-URL"
```

"plugin-download-URL" specifies the URL from which Java Plug-in can be downloaded. For Java Plug-in 1.5.0, as an example, "plugin-download-URL" specifies the following URL (which may be changed when Java Plug-in is upgraded).

```
http://java.sun.com/update/1.5.0/jinstall-1_5_0-windows-i586.cab#Version=1,5,0,0
```

- Check whether the <PARAM> tag is coded as shown below between the <OBJECT>...</OBJECT> tags.

```
<PARAM NAME="type" VALUE="plugin-type-information">
```

As "plugin-type-information," the type information used internally by Java Plug-in is coded. The followings are the examples of the type names. (The type names may change when Java Plug-in is upgraded.)

a. application/x-java-applet

b. application/x-java-applet;version=nnn

c. application/x-java-bean

d. application/x-java-bean;version=nnn

('nnn' is a version number of Java Plug-in.)

If the tag is coded, replace the <PARAM> tag with the following coding. If the tag is not coded, add the <PARAM> tag as shown above between the <OBJECT>...</OBJECT> tags.

```
<PARAM NAME="TYPE" VALUE="application/x-JBK-Plugin">
```

2.3 How to Use

The setting related to JBK Plugin is coded in a file named jbkplugin.properties. The jbkplugin.properties file is stored in the "classes" subdirectory of the directory where JBK runtime is installed. To change the setting of JBK Plugin, edit the contents of the jbkplugin.properties file by a text editor.

Table 2.1 Sample setting of jbkplugin.properties

```
##**************************************************************************
##* J Business Kit: Copyright (c) FUJITSU LIMITED  1999 - 2011
##*                      All Rights Reserved
##*
##* jbkplugin.properties
##**************************************************************************
# A line which starts with '#' is a comment line.
#
# Notice:
# - Maximum length in each line is 1024 bytes.
# - For Windows Vista/Windows 7, When you will open an applet on the web site
#   within protection mode, please specify the stored folder as under the
#   AppData\LocalLow in the user directory (represented as ${user.home}).
# - If the application installed folder name contains white space,
#   please don't use "${jbk.home}" for the jbk.plugin.policy.url property.
#   If set, the policy file fails to be read.

# the Java VM used in JBK Plugin (optional)
# Example:
#jbk.plugin.javahome=C:\jdk6
# or
#jbk.plugin.javahome=C:\jdk7

# Java VM startup options (optional)
jbk.plugin.vmoption=-Dsun.java2d.noddraw=true
```

```
# JBK Plugin policy file
jbk.plugin.policy.url=file:${jbk.home}/classes/jbkplugin.policy

# Proxy setting (optional)
# Example:
#jbk.plugin.proxy.enable=true
#jbk.plugin.proxy.http.host=proxy.fujitsu.com
#jbk.plugin.proxy.http.port=8080
#jbk.plugin.proxy.override=*.fujitsu.com

# The Java console
jbk.plugin.console.visible=false

# The download method of applets
# The following are available:
# - java:
#   Uses the Java networking class.
# - native: (for Internet Explorer only)
#   Uses the network functionality of the browser.
# - hybrid:
#   For class files and JAR files: performs as 'native.'
#   For other files: performs as 'java.'
#
jbk.plugin.protocol.http=hybrid

# The HTTPS download method of applets
# The following are available:
# - java:
#   Uses the Java networking class.
# - native: (for Internet Explorer only)
#   Uses the network functionality of the browser.
# - hybrid:
#   For class files and JAR files: performs as 'native.'
#   For other files: cannot be downloaded.
#
jbk.plugin.protocol.https=hybrid

# Applet caching
jbk.plugin.www.plugin_cache.enable=false
jbk.plugin.www.plugin_cache.dir=${jbk.home}\applet_cache
jbk.plugin.www.plugin_cache.dir.freespace=1%
jbk.plugin.www.plugin_cache.update=newer
# Accelerate mode. this property is avaliable in 'newer' mode only.
#jbk.plugin.www.plugin_cache.accelerate=true
# 2 pass mode. cannot set true with accelarate.
#jbk.plugin.www.plugin_cache.twopass=true

# Shows the download status messages
jbk.plugin.applet.showmessage=true

# Output Java VM messages
#jbk.plugin.debug.showvmmsg=true
# Java VM messages stored folder.
#jbk.plugin.debug.tracedir=C:\WINDOWS\TEMP

# Allows calling applet's methods from JavaScript
#jbk.plugin.sw.script.enable=true

# Logging Java console output
jbk.plugin.debug.console_log=false
# Stored folder for Java console output log
#jbk.plugin.debug.console_log.dir=${jbk.home}/console_log
```

```
# Maximum file numbers for Java console output log
#jbk.plugin.debug.console_log.history=5
# Maximum file sizes for Java console output log
#jbk.plugin.debug.console_log.size=100K
# Adding header for each line
#jbk.plugin.debug.console_log.header=false
```

Note

When the contents of the jbkplugin.properties file are changed, the change is not reflected in active JBK Plugin. To reflect the new setting in JBK Plugin, restart the browser and re-read the HTML file using JBK Plugin by the browser.

The contents of the jbkplugin.properties file

The contents of the jbkplugin.properties file are described in the following sections:

- 2.3.1 Specifying the Java VM Used in JBK Plugin
- 2.3.2 Specifying Java VM Startup Options
- 2.3.3 Security Setting
- 2.3.4 Proxy Setting
- 2.3.5 Using the Java Console
- 2.3.6 How JBK Plugin Downloads Class Files
- 2.3.7 USING THE HTTPS PROTOCOL
- 2.3.8 Displaying Download Status Messages
- 2.3.13 Applet Caching
- 2.3.14 Download status notification

Besides, JBK Plugin has the following functionalities:

- 2.3.9 Starting a Java VM in Advance

  Before an applet is executed, JBK Plugin can start a Java VM in advance.

- 2.3.10 Notifying Applet Activation and Inactivation

  The applet executed with JBK Plugin has its active and inactive status. JBK Plugin notifies the change of the status to the applet.

- 2.3.11 Confirming Applet Termination

  If the user is going to quit the browser while applets are running in it, JBK Plugin can ask each applet whether it can be terminated.

- 2.3.12 Switching the Property Files

  The user can switch the JBK Plugin property file (jbkplugin.properties) for each browser.

- 2.3.15 JavaScript Communication

  The applets can call JavaScript and JavaScript can call applet's public methods.

2.3.1 Specifying the Java VM Used in JBK Plugin

This section describes how to specify the Java VM, which JBK Plugin uses to execute an applet.

By default, JBK Plugin uses JavaSE selected at the installation of JBK, as a Java VM. If you use the default, you do not have to specify the Java VM.

Specifying the Java VM to be Used

To change the default Java VM to be used, code the following line in the jbkplugin.properties file.

```
jbk.plugin.javahome=<JavaSE-install-directory>
```

As <JavaSE-install-directory>, specify the install directory of JavaSE to be used as a Java VM.

Note

- In the default state, the line specifying the Java VM in the jbkplugin.properties file is a comment line (line starting with '#'). When specifying the Java VM, erase the '#' at the beginning of the line.
- If the name of the JavaSE install directory includes a blank character, enclose the JavaSE install directory name by double quotation (") marks.
- For Java VM to be used with JBK Plugin, specify a version of JavaSE suitable for the applets that will be executed.

Example

Sample specifications of the Java VM to be used

```
jbk.plugin.javahome="C:\Program Files\jdk1.6.0"
```

Point

Search sequence for finding the Java VM to be used

JBK Plugin searches for the Java VM to be used to execute an applet in the following sequence:

1. If the Java VM to be used is coded in the jbkplugin.properties file, JBK Plugin uses it.
2. If the Java VM to be used is not coded in the jbkplugin.properties file, JavaSE selected at the installation of JBK is used as a Java VM (default operation).
3. If the information of JavaSE selected at the installation of JBK cannot be obtained, JavaSE specified in the JAVA_HOME environment variable is used as a Java VM. This situation occurs if the registry including the JBK installation information cannot be referenced.

2.3.2 Specifying Java VM Startup Options

This section describes how to specify Java VM startup options.

The user can specify Java VM startup options for executing a Java VM with JBK Plugin. Command line options of "java" command can be used as Java VM startup options.

Specifying Java VM Startup Options

To specify Java VM startup options, code the following line in the jbkplugin.properties file.

```
jbk.plugin.vmoption=<java-vm-startup-options>
```

As <java-vm-startup-options>, specify the following options: See the "java" command reference manual for the detail of each option.

Startup Options for the Java VM of the JavaSE

Options must be separated by one or more space characters (' ').

-agentlib :

Specifies the native agent library.

-classpath (or -cp) :

Specifies the class path.

-D :

Defines a property value.

-Xbootclasspath :

Specifies the boot class path.

-Xbootclasspath/p :

Prepends the specified path in front of the default boot class path.

-Xbootclasspath/a :

Appends the specified path to the default boot class path.

-Xms :

Sets the startup amount of the heap size.

-Xmx :

Sets the maximum amount of the heap size.

-Xnoclassgc :

Turn off garbage collection of Java class.

-Xrunhprof :

Enables to profile CPU, heap, and so on.(cannot use -Xrunhprof:help)

-Xrs :

Reduces the use of OS signals.

-Xfuture :

Verifies the format of the class file strictly.

Note

- By default, "-Dsun.java2d.noddraw=true" (suppress DirectDraw) is specified for the Java VM startup option in jbkplugin.properties. It is recommended that the setting of the option not be changed from the default.
- If you want to specify "-classpath" with the path name that includes space characters, you must enclose the path name with "".
- The startup amount of the heap size (specified by "-ms" of "-Xms") must be equal to or less than the maximum amount of the heap size (specified by "-mx" or "-Xmx"). Otherwise, JBK Plugin fails to execute a Java VM.
- The length of an option is limited to 1024 bytes, including "jbk.plugin.vmoption=". Strings that exceed 1024 bytes will be ignored.

Example

Sample specifications of Java VM startup options

```
jbk.plugin.vmoption=-Xms32m -Xmx96m -Dsun.java2d.noddraw=true
```

2.3.3 Security Setting

This section describes how to specify a level of security checks to be performed when an applet is executed.

The Java VM of the J2SE uses a policy file for configuring security level. A policy file describes permissions which allows applets to do the operations subjected to security checks. A policy file enables more flexible security control than that of the Java platform 1.1.

When JBK Plugin works with the J2SE, it uses the policy file for the security control of the applet. For the policy files used by JBK Plugin, look at the notes below. Also check the JDK document for the detail of the security control with the policy file.

Policy Files Used by JBK Plugin

JBK Plugin uses the following two policy files:

- JBK Plugin policy file

  The following line in jbkplugin.properties describes the directory where the JBK Plugin policy file is located:

  ```
  jbk.plugin.policy.url=<policy-file-url>
  ```

  In <policy-file-url>, the user describes the absolute path of the policy file as the URL form. The default setting is:

  - file:${jbk.home}/classes/jbkplugin.policy

    ${jbk.home} means the directory where JBK is installed.

  When the installation folder includes spaces, do not use ${jbk.home} in the jbk.plugin.policy.url. Reading of the policy file will fail.

- The default policy file of the JavaSE

  The JavaSE has its policy file in the following directory:

  - JDK : <JDK-directory>\jre\lib\security\java.policy
  - JRE : <JRE-directory>\lib\security\java.policy
  - common : (${user.home})\.policy

Permissions described in both policy files are valid. If permissions for the same applet are described in both files, both descriptions are valid.

You can describe permissions in either of them. If you use the JBK Plugin policy file, you do not need to re-describe the permissions when you change the Java VM used in JBK Plugin, or when you re-install the Java VM.

Example

Sample Description of the Policy File

The following is a sample description of the policy file. This allows an applet to do all operation subjected to security checks.

```
grant codeBase "(the codebase of the applet)" {
     //AllPermission allows all permissions to the applet.
     permission java.security.AllPermission;
};
```

Note that AllPermission in this sample description enables all permissions for the applet.

In actual operation, specify only the necessary permissions, not AllPermission.

Point

If you do not want to use the JBK Plugin policy file, delete the line of "jbk.plugin.policy.url" in jbkplugin.properties, or insert '#' at the beginning of the line.

Required Permission for Each Operation

The table below lists the required permissions for the operations subjected to security checks. See the JDK documentation for the detail of these permissions and how to specify them in the policy file.

Operation to be checked	Required permission
Reading a local file	(using the file descriptor) java.lang.RuntimePermission "readFileDescriptor";
	(using the file name) java.io.FilePermission "<file>","read";
Writing a local file	(using the file descriptor) java.lang.RuntimePermission "writeFileDescriptor";
	(using the file name) java.io.FilePermission "<file>","write";
Deleting a local file	java.io.FilePermission "<file>","delete";
Asking for a network connection (connect)	(using -1 as the port number) java.net.SocketPermission "<host>","resolve";
	(otherwise) java.net.SocketPermission "<host>:<port>","connect";
Waiting in a network port (listen)	(using 0 as the port number) java.net.SocketPermission "localhost:1024-","listen";
	(otherwise) java.net.SocketPermission "localhost:<port>","listen";
Accepting a request for a network connection (accept)	java.net.SocketPermission "<host>:<port>","accept";
Using the multicast mode	java.net.SocketPermission(maddr.getHostAddress(),"accept,connect");
Setting the factory of a network-related class	java.lang.RuntimePermission "setFactory";
Operating a thread of a different thread group	(operating a thread) java.lang.RuntimePermission "modifyThread";
	(operating a thread group) java.lang.RuntimePermission "modifyThreadGroup";
Starting a local application	(specifying the absolute path for the command) java.io.FilePermission "<cmd>","execute";
	(specifying the relative path for the command) java.io.FilePermission "-","execute";
Loading a local library (DLL)	java.lang.RuntimePermission "loadLibrary.<lib>";
Accessing the system property	(using System.getProperties()) java.util.PropertyPermission "*","read,write";
	(using System.getProperty(String key)) java.util.PropertyPermission "<key>","read,write";
Creating a window	java.awt.AWTPermission "showWindowWithoutWarningBanner";
Loading a class from an applet	java.lang.RuntimePermission "getClassLoader";
Accessing a print job	java.lang.RuntimePermission "queuePrintJob";
Accessing the clipboard	java.awt.AWTPermission "accessClipboard";
Accessing the AWT event queue	java.awt.AWTPermission "accessEventQueue";
Operating the class loader	java.lang.RuntimePermission "getClassLoader";

Operation to be checked	Required permission
Terminating the browser	(With JBK Plugin, an applet cannot terminate the browser.)

Note

Compared with JBK Plugin V3, JBK Plugin V4 strengthens the security check of the applet. For this reason, the applet running JBK Plugin V3 may encounter the following case:

- When the applet connects with the Web server via a proxy, and the host name of the Web server is not registered in DNS, it may takes several minutes to establish the connection.

If it makes trouble for you, you can specify 'jbk.plugin.security.strict=false' in jbkplugin.properties. Then, JBK Plugin uses the same security check as V3.

In the case the applet encounters any security error, however, you should not use 'jbk.plugin.security.strict=false'. You should add proper permissions in the policy file, instead.

2.3.4 Proxy Setting

This section describes how to set an HTTP proxy server to be used by JBK Plugin.

When a local area network (LAN) is built, a firewall may be provided to prevent intrusions from other networks. To download an applet from a Web server outside a LAN, the downloading must be done through an HTTP proxy server. If this operation is needed, set the HTTP proxy server as described below.

In the following cases, the user does not have to make the proxy setting.

- When the same setting as the proxy setting of the browser is used

  By default, JBK Plugin uses the proxy setting of the browser as it is.

- When the user specifies 'hybrid' or 'native' for the downloading method of the applet

  In this case, JBK Plugin uses the proxy setting of the browser instead of the setting in the jbkplugin.properties file described below. See '2.3.6 How JBK Plugin Downloads Class Files' for the detail.

Setting a Proxy Server

To change the default proxy setting, code the following lines in the jbkplugin.properties file:

```
jbk.plugin.proxy.enable=<HTTP-proxy-server-used-or-not>
jbk.plugin.proxy.http.host=<HTTP-proxy-server-host-name>
jbk.plugin.proxy.http.port=<HTTP-proxy-server-port-number>
jbk.plugin.proxy.override=<HTTP-proxy-override-address-list>
```

Specify the following information in the properties:

- jbk.plugin.proxy.enable

  This property specifies whether an HTTP proxy server is used. As <HTTP-proxy-server-used-or-not>, code either of the following:

  true

    An HTTP proxy server is used when an applet is downloaded.

  false

    An HTTP proxy server is not used when an applet is downloaded.

    When this property is set to false, the following three property setting are invalid.

- jbk.plugin.proxy.http.host

  This property specifies the host name of an HTTP proxy server. As <HTTP-proxy-server-host-name>, specify the host name or IP address of an HTTP proxy server.

- jbk.plugin.proxy.http.port

  This property specifies the port number of the HTTP proxy server. As <HTTP-proxy-server-port-number>, specify the port number the HTTP proxy server uses.

- jbk.plugin.proxy.override

  This property specifies a list of addresses to be connected without the HTTP proxy server. If this list includes a Web server address, JBK Plugin downloads an applet directly from the Web server, not via the HTTP proxy server. As <HTTP-proxy-override-address-list>, code an address that does not use the HTTP proxy server, as described below:

  - The address coding can be a host name, domain name, or IP address.

  - The coded host name, domain name, or IP address can contain a wild card character (*).

  - When two or more addresses are coded, the addresses should be separated by a vertical line ('|').

Example

Sample proxy setting

```
jbk.plugin.proxy.enable=true
jbk.plugin.proxy.http.host=proxy.fujitsu.com
jbk.plugin.proxy.http.port=8080
jbk.plugin.proxy.override=*.fujitsu.com|*.foo.co.jp
```

- By default, the proxy setting line in the jbkplugin.properties file is commented out. When setting a proxy server, eliminate the # at the beginning of the line.

- The contents coded as <HTTP-proxy-server-host-name>, <HTTP-proxy-server-port-number>, and <HTTP-proxy-override-address-list> depend on the LAN environment setting. For details of the proxy setting, consult the network administrator.

Browser Proxy Bypass List

In the Java proxy bypass list, asterisks (".*.*") cannot be recognized as wild cards and need to be connected using a proxy. Under the following circumstances, if the bypass list under the browser's proxy setting is using ".*.*", record it as ".*".

- Use the same setting as the browser's proxy setting.

- In the jbkplugin.properties, specify jbk.plugin.proxy.enable=true and set jbk.plugin.proxy.override.

2.3.5 Using the Java Console

This section describes the Java console of JBK Plugin.

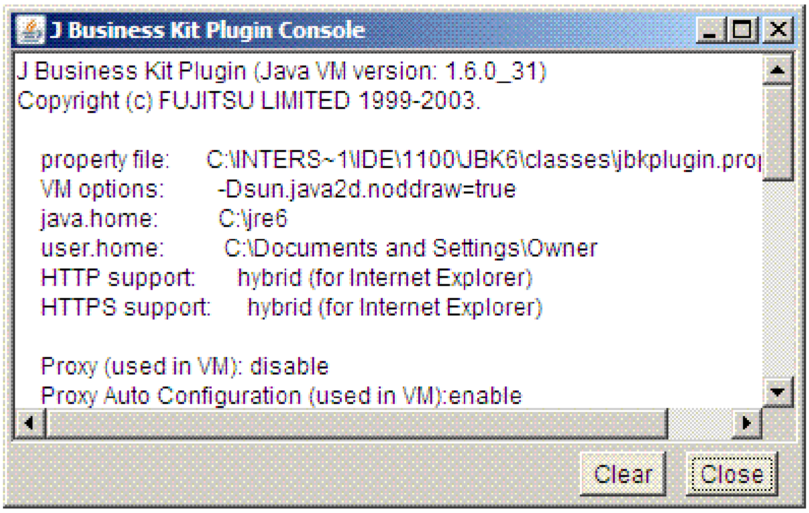
JBK Plugin provides the Java console that displays useful information for debugging an applet. The Java console displays the following information:

- Messages output to the standard output (System.out) or standard error output (System.err) when the applet is executed

- Messages displayed in the status window of the browser when the showStatus() method of the AppletContext class is executed

- Messages indicating exceptions occurring when an applet is executed

Figure 2.2 The Java console of JBK Plugin



```
J Business Kit Plugin (Java VM version: 1.6.0_31)
Copyright (c) FUJITSU LIMITED 1999-2003.

  property file:    C:\INTERS~1\IDE\1100\JBK6\classes\jbkplugin.pro|
  VM options:       -Dsun.java2d.noddraw=true
  java.home:        C:\jre6
  user.home:        C:\Documents and Settings\Owner
  HTTP support:     hybrid (for Internet Explorer)
  HTTPS support:    hybrid (for Internet Explorer)

  Proxy (used in VM): disable
  Proxy Auto Configuration (used in VM):enable
```

The Java console has two buttons:

- "Clear" - Erases all the displayed information.
- "Close" - Hides the Java console.

In addition, the Java console allows the following operations to be performed with the associated keys:

C

Clears the Java console.

G

Executes garbage collection.

H

Shows keyboard help.

M

Shows the amount of memory being used.

S

Outputs the system properties.

T

Outputs the thread list.

0

Switches between JBK Plugin trace outputs (Traces of class loadings or HTTP connection states.).

Ctrl+Shift+E

Switches between AWTEvent trace outputs (Traces of FocusEvent, KeyEvent, MouseEvent, and WindowEvent.).

Displaying or Hiding the Java Console

The Java console is not displayed by the default (initial) setting specified when JBK Plugin is installed.To display or hide the Java console when JBK Plugin starts to run, code the following line into the jbkplugin.properties file:

```
jbk.plugin.console.visible=<Java-console-displayed-or-hidden>
```

As <Java-console-displayed-or-hidden>, specify either of the following:

true

Displays the Java console.

false

Hides the Java console. (default)

To redisplay the Java Console once it has been hidden, press [Ctrl]+[Alt]+[Insert] when the applet has the focus.

Displaying or Hiding the Java Console from an Applet

JBK Plugin provides the PluginAppletContext interface for displaying or hiding the Java console from an applet.

The API of the PluginAppletContext interface is as shown below:

- Interface
    - com.fujitsu.jbk.plugin.browser.PluginAppletContext
      (extends java.applet.AppletContext)
- Methods
    - setConsoleVisible

      Displays or hides the Java console.

      ```
      public abstract void setConsoleVisible(boolean visible)
      ```

      visible

      true:The Java console is displayed.

      fals:The Java console is hidden.

    - isConsoleVisible

      Returns whether the Java console is displayed or not.

      ```
      public abstract boolean isConsoleVisible()
      ```

      Return value

      Returns true when the Java console is displayed. Returns false when the Java console is hidden.

The class file for the PluginAppletContext interface is stored in the jar file for the development of the JBK plugin ("classes\jbkstd.jar" in the JBK installation folder). When compiling the class of an applet that uses the PluginAppletContext interface, check whether the jar file for the development of the JBK plugin is included in the classpath.

The PluginAppletContext instance can be obtained by invoking the getAppletContext() method of java.applet.Applet class. The following is an example of the applet using this functionality, which acts as a button to show the Java console.

Table 2.2 Applet having the Java console display button

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

import com.fujitsu.jbk.plugin.browser.PluginAppletContext;

/**
 * This applet acts as a button to show the Java console.
 */
public class ConsoleButton extends Applet implements ActionListener {
    /**
     * Initializes the applet.
     */
    public void init() {
        // Creates a button to show the Java console.
```

```
        setLayout(new BorderLayout());
        Button b = new Button("Show the Java console");
        b.addActionListener(this);
        add(b);
    }

    /**
     * Invoked when the button is pressed.
     */
    public void actionPerformed(ActionEvent e) {
        // Gets the AppletContext.
        AppletContext context = getAppletContext();

        try {
            // Checks whether the AppletContext is an instance of the PluginAppletContext
            if (context instanceof PluginAppletContext) {
                PluginAppletContext plgContext = (PluginAppletContext)context;
                // Show the Java console if it is not shown.
                if (!plgContext.isConsoleVisible()) {
                    plgContext.setConsoleVisible(true);
                }
            }
        } catch (Throwable ignore) {
            /* do nothing */
        }
    }
}
```

2.3.6 How JBK Plugin Downloads Class Files

This section describes the download methods of the applet used in JBK Plugin.

JBK Plugin provides the following three methods for downloading applets.

- Using Java networking class
- Using the networking functionality of the browser
- Mixing the above two methods

When using the networking functionality of the browser, JBK Plugin downloads applets according to the network setting of the browser, such as the proxy and caching. To use these setting, select the method of using the networking functionality of the browser.

What Kind of Files Are These Download Methods Applied to?

These download methods are applied to the following kind of files:

- Class files used in the applet (*.class)
- JAR files for the applet (*.jar)

  JAR files which are described in the ARCHIVE attribute of the tag for JBK Plugin in the HTML file. For the detail of the ARCHIVE attribute, see '2.2.3 How to Create an HTML File for JBK Plugin'.
- Property files used in the applet (*.properties)
- Image files used in the applet (*.gif, *.jpg, and so on.)
- Other files downloaded by the applet using the API of the java.net.URL class.

The content specified in the section "How JBK Plugin Downloads Class Files" applies to these files.

Note

JBK Plugin does not download the files described below. The browser downloads them. Therefore, the download methods of JBK Plugin are not applied to these files.

- HTML files which includes the tag for JBK Plugin
- Image files used in the HTML files

Specifying the Download Method of the Applet

To specify the download method, code the following line in the jbkplugin.properties file.

```
jbk.plugin.protocol.http=<applet-download-method>
```

As <applet-download-method>, specify one of the following keywords:

java

Downloads the files by using Java networking class.

The downloaded files are not cached in the local disk. As the HTTP proxy, JBK Plugin uses the proxy setting in the jbkplugin.properties file (see '2.3.4 Proxy Setting' for the detail of the proxy setting.)

native

Downloads the files by using the networking functionality of the browser.

- The downloaded file is cached in the browser cache directory. From the next time on, the file is loaded from the cache. For the detail of the browser cache and its setting, see the manual of the browser.
- As the HTTP proxy, JBK Plugin uses the proxy setting of the browser. For the detail of the proxy setting of the browser, see the manual of the browser.

hybrid

Downloads the files by using both of the above two methods. In this method, class files and JAR files are downloaded by using the networking functionality of the browser, while the other files are downloaded by using Java networking class. By default, the download method is set to hybrid.

Note

- Earlier versions of JBK Plugin do not support downloading of applets via a browser. To specify that applets are not to be downloaded via a browser, specify "jbk.plugin.protocol.http" for java.
- In Internet Explorer with "hybrid" or "native" specified in the parameter, the followings occur when a file connected by using java.net.URLConnection is cached in the browser:
  - The getHeaderFieldDate or getDate method will return 0.
  - The getHeaderField or getHeaderFieldKey method will not return the Server or Date field.

Example

Sample setting of download method

```
jbk.plugin.protocol.http=native
```

2.3.7 USING THE HTTPS PROTOCOL

This section describes the HTTPS protocol support in JBK Plugin.

The HTTPS protocol is the HTTP protocol secured with SSL (Secure Sockets Layer). The HTTPS protocol enables data encryption, client authentication and server authentication on the HTTP connection.

Currently, most Web servers and Web browsers supports the HTTPS protocol. JBK Plugin also supports the HTTPS protocol and enables to download applets from the Web server that uses the HTTPS protocol.

What Kind of Files Are the HTTPS Download Methods Applied to?

The HTTPS download methods are applied to the following files.These are the same as those of the usual HTTP download methods. See '2.3.6 How JBK Plugin Downloads Class Files', also.

- Class files used in the applet (*.class)
- JAR files for the applet (*.jar)
- Property files used in the applet (*.properties)
- Image files used in the applet (*.gif, *.jpg, and so on.)
- Other files downloaded by the applet using the API of the java.net.URL class.

Specifying the HTTPS Download Method of the Applet

To specify the HTTPS download method, code the following line in the jbkplugin.properties file.

```
jbk.plugin.protocol.https=<https-download-method>
```

As <https-download-method>, specify one of the following keywords:

java

Downloads the files by using Java networking class .Downloads the files by using JSSE (Java Secure Socket Extension).

native

Downloads the files by using the networking functionality of the browser.

- If the browser is set to cache the file downloaded by the HTTPS protocol, it is cached in the browser cache directory. From the next time on, the file is loaded from the cache. For the detail of the browser cache and its setting, see the manual of the browser.
- As the HTTPS proxy, JBK Plugin uses the proxy setting of the browser. For the detail of the proxy setting of the browser, see the manual of the browser.

hybrid

Downloads the files by using both of the above two methods. In this method, class files and JAR files are downloaded by using the networking functionality of the browser, while the other files cannot be downloaded by the HTTPS protocol. By default, the HTTPS download method is set to hybrid.

The communication of the HTTPS protocol is supported by JSSE(Java Secure Socket Extension). Therefore, when "hybrid" is specified, it is possible to download it to files other than the class file and the JAR file by the HTTPS protocol.

Note

- If SSL is disabled in the browser you are using, specifying native or hybrid for "jbk.plugin.protocol.https" does not result in downloading of applets in the HTTPS protocol. To download applets using the HTTPS protocol, enable SSL in the browser beforehand. For information about enabling and disabling SSL in your browser, see your browser's manual.
- In Internet Explorer with "hybrid" or "native" specified in the parameter, the followings occur when a file connected by using java.net.URLConnection is cached in the browser:
  - The getHeaderFieldDate or getDate method will return 0.
  - The getHeaderField or getHeaderFieldKey method will not return the Server or Date field.

Example

Sample setting of HTTPS download method

```
jbk.plugin.protocol.https=native
```

2.3.8 Displaying Download Status Messages

This section describes the download status messages displayed by JBK Plugin.

Download Status Messages

While downloading an applet, JBK Plugin displays the following download status messages in the upper-left corner of the area where the applet is to be shown:

- "Loading an applet..."

  JBK Plugin is loading an applet.

- "Initializing the applet..."

  JBK Plugin is initializing the applet.

If an error occurs during the download, JBK Plugin displays the following message:

- "ERROR: fail to load the applet."

The detail of the error is displayed in Figure 2.2 The Java console of JBK Plugin of JBK Plugin.

Displaying or Hiding the Download Status Messages

To specify whether the download status messages are shown or not, code the following line in the jbkplugin.properties file.

```
jbk.plugin.applet.showmessage=<show-download-status-message>
```

As <show-download-status-message>, code either of the following:

true

Displays the download status messages

false

Hides the download status messages

Note

Earlier versions of JBK Plugin do not show an applet's download status. To make the current version the same as previous versions in terms of showing the download status, specify "jbk.plugin.applet.showmessage" for false.

2.3.9 Starting a Java VM in Advance

This section describes the method of starting a Java VM in advance before starting an applet, by using JBK Plugin.

Generally, when an applet is first executed on a browser, a Java VM is started. It takes several seconds to start a Java VM. During that time, the browser may not accept any operation from the user, and the user may have to wait until the activation of the Java VM is completed. To prevent this situation, JBK Plugin provides the functionality of starting a Java VM in advance, without executing an applet, when the HTML file is coded accordingly. With this functionality, the Java VM can be started before an applet is started, and the user does not have to wait.

How to Create an HTML File for Starting a Java VM in Advance

When using JBK Plugin to start a Java VM in advance without executing an applet, code the name of the applet (NAME attribute) as "@JAVAVMONLY" in the HTML file for JBK Plugin. For the coding of the HTML file for JBK Plugin, see '2.2.3 How to Create an HTML File for JBK Plugin'.

If the NAME attribute is specified as "@JAVAVMONLY," JBK Plugin disables the corresponding CODE attribute and starts a Java VM in advance in the background.

Example

Sample coding for starting a Java VM in advance

When starting a Java VM in advance by using JBK Plugin, include the following in the HTML file :

```
<OBJECT CLASSID="CLSID:BEA62964-C40B-11D1-AACA-00A0C9216A67" WIDTH=1 HEIGHT=1>
    <PARAM NAME="TYPE" VALUE="application/x-JBK-Plugin">
    <PARAM NAME="NAME" VALUE="@JAVAVMONLY">
</OBJECT>
```

Point

Required Attributes of the Tag

- If the NAME attribute is specified as "@JAVAVMONLY," the corresponding CODE attribute is unnecessary.
- If the NAME attribute is specified as "@JAVAVMONLY," the corresponding section is displayed as a rectangle having the width and height specified by WIDTH and HEIGHT on the browser screen. The user is recommended to specify 1 as both the WIDTH attribute and the HEIGHT attribute so that the rectangle does not stand out on the screen.

How to Start a Java VM in Advance

To start Java VM in advance, follow these steps:

1. Creating an HTML file for starting a Java VM in advance

   a. In addition to an HTML file for executing an applet, create an HTML file for starting a Java VM in advance, according to the above description. This HTML file can be used to process any operations needed before the execution of an applet.

      - displaying the title page of the application
      - displaying a notice
      - user's input request, etc.

   b. The specification of "How to Create an HTML File for Starting a Java VM in Advance" is described in this HTML file. Then this HTML file will be the HTML file for starting a Java VM in advance.

2. Executing the HTML file

   a. When the user starts the browser, he should display the HTML file for starting a Java VM in advance. While the user is doing some operations in the HTML file, JBK Plugin starts the Java VM in the background. The activation of the Java VM will not interfere with the processing on the browser screen.

   b. The user finishes the operation and then displays the HTML file for executing the applet. Since the Java VM has already been started, JBK Plugin downloads and starts the applet immediately. The user does not have to wait until the activation of the Java VM is completed.

Note

The HTML file for starting a Java VM in advance should not include the coding for executing an applet for JBK Plugin. If the HTML file contains a coding for executing an applet, the coding for starting a Java VM in advance in the HTML file becomes invalid.

Example

Sample HTML file for starting a Java VM in advance

The sample coding of an HTML file for starting a Java VM in advance is shown below:

```
<HTML>
<HEAD>
<TITLE>An Example HTML file for starting a Java VM in advance</TITLE>
</HEAD>

<BODY>
<!--
  This tag is shown as an 1x1 rectangle with the same color as the
  background color of the browser window.
-->
<OBJECT CLASSID="CLSID:BEA62964-C40B-11D1-AACA-00A0C9216A67" WIDTH=1 HEIGHT=1>
  <PARAM NAME="TYPE" VALUE="application/x-JBK-Plugin">
  <PARAM NAME="NAME" VALUE="@JAVAVMONLY">
<OBJECT>

</BODY>
</HTML>
```

2.3.10 Notifying Applet Activation and Inactivation

This section describes the notification of applet activation and inactivation used in JBK Plugin.

An HTML document can have several applets in it, so the browser may display several applets in its HTML document window at a time. Therefore, the active and inactive status of each applet does not always match those of the browser window.

JBK Plugin defines the active and inactive status of the applet on its own terms and notifies the change of the status to the applet. When constructing system with several applets, this functionality gives the ability of changing some environment for executing applets according to which applet is activated.

When an Applet Becomes Active or Inactive?

In the following cases, JBK Plugin considers that an applet becomes active:

- Just after the applet has been started (i.e. just after the start() method of the applet has been invoked.)

- When a keyboard focus is set on any component of the applet.

- When a mouse is clicked on the applet.

On the other hand, JBK Plugin considers that an applet becomes inactive in the following cases:

- After the applet stops (i.e. after the stop() method of the applet has been invoked.)

- When a keyboard focus is set on any place other than the applet.

- When the browser window which the applet is on becomes inactive.

- When another applet becomes active.

Notification Event for the Applet Activation and Inactivation

When an active applet becomes inactive or an inactive one becomes active, JBK Plugin notifies the PluginAppletEvent to the applet. The PluginAppletEvent class is provided by JBK Plugin.

The API of the PluginAppletEvent class is as shown below:

- Class
  - com.fujitsu.jbk.plugin.browser.PluginAppletEvent
    (extends java.awt.AWTEvent)
- Fields
  - PLUGINAPPLET_ACTIVATE

    This id means that the applet becomes active.

    ```
    public final static int PLUGINAPPLET_ACTIVATE
    ```

  - PLUGINAPPLET_DEACTIVATE

    This id means that the applet becomes inactive.

    ```
    public final static int PLUGINAPPLET_DEACTIVATE
    ```

- Constructor
  - PluginAppletEvent

    ```
    public PluginAppletEvent(Object source, int id)
    ```

    source

    specifies the applet which this event occurred on.

    id

    the event id

Point

Notification Conditions

- PLUGINAPPLET_ACTIVATE notifies the applet at the earliest possible time.
- When the applet in an inactive window starts to run, JBK Plugin does not notify PluginAppletEvent to the applet because the window is inactive.

  Code the following lines in the jbkplugin.properties file to notify PluginAppletEvent of PLUGINAPPLET_ACTIVATE, when applet in an inactive window has started to run.
- PLUGINAPPLET_ACTIVATE notifies the applet first.

```
jbk.plugin.sw.event.initial_active=true
```

Methods to Receive the Notification Event

When receiving a PluginAppletEvent, follow this procedure:

1. First, create an event listener to receive the PluginAppletEvent.

   The listener has to implement the PluginAppletListener interface, which is provided by JBK Plugin.

   The API of the PluginAppletEvent interface is as shown below:

- Interface

    - com.fujitsu.jbk.plugin.browser.PluginAppletListener
      (extends java.util.EventListener)

- Methods

    - eventOccurred

      receives the event notified from JBK Plugin.

      ```
      public abstract void eventOccurred(PluginAppletEvent event)
      ```

      event

      the event notified to the applet

2. Next, register the listener in the applet context (PluginAppletContext).

   The listener created has to be registered in the applet context. The applet context of JBK Plugin (PluginAppletContext class) provides the methods for adding and removing the listener in it.

   The API is shown below:

   - Interface

       - com.fujitsu.jbk.plugin.browser.PluginAppletContext
         (extends java.applet.AppletContext)

   - Methods

       - addPluginAppletListener

         Adds an event listener to receive the notification event from JBK Plugin.

         ```
         public abstract void addPluginAppletListener(PluginAppletListener listener)
         ```

         listener

         an event listener to be added

       - removePluginAppletListener

         Removes an event listener to receive the notification event from JBK Plugin.

         ```
         public abstract void removePluginAppletListener(PluginAppletListener listener)
         ```

         listener

         an event listener to be removed

       - activateApplet

         If the given applet belongs to this applet context, this method activates it. If it does not, this method does nothing.

         ```
         public abstract void activateApplet(Applet applet)
         ```

         applet

         an applet to be activated

The class file for the PluginAppletEvent class, the PluginAppletListener interface and the PluginAppletContext interface is stored in the jar file for the development of the JBK plugin ("classes\jbkstd.jar" in the JBK installation folder). When compiling the class of an applet that uses the PluginAppletContext interface, check whether the jar file for the development of the JBK plugin is included in the classpath.

Example

An example of handling the notification event of the active and inactive status

The applet context of JBK Plugin can be retrieved using the getAppletContext() method of java.applet.Applet class. An example of handling the notification events is shown below.

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

import com.fujitsu.jbk.plugin.browser.PluginAppletEvent;
import com.fujitsu.jbk.plugin.browser.PluginAppletListener;
import com.fujitsu.jbk.plugin.browser.PluginAppletContext;

/**
 * Sample applet for handling its active and inactive status.
 * This applet implements the PluginAppletListener interface and
 * receives the PluginAppletEvent event.
 */
public class ActivateSample extends Applet implements PluginAppletListener {

    /**
     * Initializes the applet.
     */
    public void init() {
        setBackground(Color.lightGray);
        // Gets the applet context.
        // When this applet runs in JBK Plugin,
        // getAppletContext() returns an instanceof the PluginAppletContext.
        AppletContext context = getAppletContext();
        try {
            // Checks whether the applet context is an instance of PluginAppletContext class.
            if (context instanceof PluginAppletContext) {
                PluginAppletContext plgContext = (PluginAppletContext)context;
                // Adds an event listener to receive PluginAppletEvent
                    plgContext.addPluginAppletListener(this);
            }
        } catch (Throwable ignore) {
            // do nothing
        }
    }

    /**
     * Receives the notification event (PluginAppletEvent) from JBK Plugin.
     * This method changes the background color of the applet
     * according to its active or inactive status.
     */
    public void eventOccurred(PluginAppletEvent e) {
        switch (e.getID()) {
            case PluginAppletEvent.PLUGINAPPLET_ACTIVATE:
                // The applet becomes active.
                setBackground(Color.red);
                repaint();
                break;
            case PluginAppletEvent.PLUGINAPPLET_DEACTIVATE:
                // The applet becomes inactive.
                setBackground(Color.lightGray);
                repaint();
                break;
            default;
                break;
        }
    }
}
```

Point

Forcibly activating an applet when the browser becomes active

You can configure the browser such that an applet is forcibly activated when the browser becomes active. To do so, include the following line in jbkplugin.properties:

```
jbk.plugin.sw.fbc.force_activate=<Specifies whether or not to activate an applet when the browser becomes active>
```

Assign one of the following values to the <Specifies whether or not to activate an applet when the browser becomes active> option:

true

Activates an applet when the browser becomes active.

false

Does not activate an applet when the browser becomes active.

By default, false is assigned to jbk.plugin.sw.fbc.force_activate. If true is specified and the browser window has more than one applet, the applet that is activated cannot be predicted.

Note that in Internet Explorer 7 and later, focus returns to the part that had focus immediately previously when the browser is active, so the above setting is generally not necessary.

2.3.11 Confirming Applet Termination

This section describes the confirmation process of the applet termination provided by JBK Plugin

JBK Plugin provides the functionality to confirm the termination of applets in a browser window when the user is going to close the window. The applets that use the functionality can allow or reject the termination.

- If all applets in the browse window allow the termination, JBK Plugin destroys the applets and closes the window.
- If some applet rejects its termination, JBK Plugin cancels to close the window. The applets continue to run.

This functionality enables an applet to prohibit its termination temporary while it access to the database, for instance.

Confirming the Termination of the Applet

To use the functionality of confirming the termination of the applet, the applet has to implement the PluginAppletCallback interface.

The PluginAppletCallback interface is provided by JBK Plugin. Its API is as shown below.

- Interface:
  - com.fujitsu.jbk.plugin.browser.PluginAppletCallback
    (extends java.lang.Object)
- Methods:
  - destroyRequested

    This method is invoked from the browser in order to confirm the termination.

    ```
    public abstract boolean destroyRequested()
    ```

    Return Value

    true - the applet allows the termination.

    false - the applet rejects the termination.

  - destroyCancelled

    This method is invoked when some applet rejects the termination.

```
public abstract void destroyCancelled()
```

Example

An applet confirmed the termination by the browser

```
import java.applet.*;

import com.fujitsu.jbk.plugin.browser.PluginAppletCallback;

/**
 * Sample applet confirmed its termination by the browser.
 */
public class DestroySample extends Applet implements PluginAppletCallback {

    /**
     * Invoked when JBK Plugin confirms the termination.
     */
    public boolean destroyRequested() {
        // Returns false if the applet does not allow its termination.
        if (/* the applet does not allow its termination */) {
            return false;
        }
        // Returns true if the applet allow its termination.
        return true;
    }

    /**
     * Invoked when some applet rejects its termination.
     */
    public void destroyCancelled() {
        // do something if you need.
    }
}
```

Note

Do not include display of a dialog box, a keyboard/mouse operation, or other processing involving a GUI operation in the destroyRequested() and destroyCancelled() methods. Otherwise, the browser may be unable to accept processing requests.

The confirmation process when several applets run in a browser window

When several applets run in one browser window, JBK Plugin invokes the destroyRequested() method of each applet in turn, and confirms the termination.

- If all applets allow the termination, the browser window is closed.
- If one or more applets rejects the termination, JBK Plugin stops the confirmation process, cancels the request for closing the browser window, and invokes the destroyCancelled() method of each applet in turn.

  Therefore, all applets can know that some applet rejects the termination.

How to invalidate the confirmation

If the destroyRequested() method of some applet always returns false by mistake, the termination of the applet is always rejected and the browser window is no longer closed. In this case, do the following:

- Create a file named "jbkplugin.shutdown" in the "class" subdirectory of the directory where JBK is installed. The file size can be zero.

If this file exists while the browser window is closing, JBK Plugin destroys all applets in the window without confirming the termination.

When the user quits the browser, JBK Plugin deletes the "jbkplugin.shutdown" file. However, it is not deleted unless the user is permitted so or it is created as a directory. Therefore, after the browser is closed, the user should check whether the file remains or not. Is it remains, please delete it manually.

2.3.12 Switching the Property Files

This section explains how to switch JBK Plugin property files used in JBK Plugin.

JBK Plugin usually runs based on the settings in the default property file jbkplugin.properties. To run JBK Plugin with different settings on the same client, prepare two or more property files and reference them selectively from the HTML file using JBK Plugin.

For example, the user can create two property files, each of which uses different JDK or JRE. Those property files can be used for each browser to execute applets on different versions of VMs.

To switch JBK Plugin property files, create the two kinds of files explained below.

- Property files for switching
- HTML File for Switching

Creating Selective Property Files

To run JBK Plugin with different settings, prepare individual property files for the particular settings. Use the default JBK Plugin property file (jbkplugin.properties) as template to prepare property for switching. Property files for switching must be stored in the same directory that the default property file (jbkplugin.properties) is stored in.

Name the selective JBK Plugin property files as jbkplugin.properties.xxx (the user can specify any string as 'xxx').

For the detail of the JBK Plugin property file, see "2.3 How to Use".

Example

Creating property files for switching

To switch between property files in order to use the Java VM with JavaSE 6 and JavaSE 7, you can name the property files by adding the JavaSE versions to them, as for example:

- name the property file for Java VM with JavaSE 6 as "jbkplugin.properties.16," and
- name the property file for Java VM with JavaSE 7 as "jbkplugin.properties.17,"

Naming in this way makes it easy to determine the content of a property file.

- Only one type of property file can be used per process. Once a property file is read, the JBK Plugin runs based on that property file until the browser terminates.
- The same value is used for the CLASSPATH environment variable in all property files. To use individual class paths for each property file, specify "-classpath" as a Java VM startup option in each property file instead of using the CLASSPATH environment variable. For information on specifying the Java VM startup option, see "2.3.2 Specifying Java VM Startup Options".
- The same value is also used for the PATH environment variable in all property files. For example, if the applet is calling JNI, and the DLL to be loaded must be changed for each JDK/JRE version, store the respective DLL in the following directories for each JavaSE :
  - For JDK : <JDK-installed-directory>\jre\bin
  - For JRE : <JRE-installed-directory>\bin

Point

- The user can specify any string as the suffix added to the selective property file name as long as the characters and the length of the string is acceptable for the file name.

- The properties that can be defined in each property file for switching are similar to those that can be defined in the jbkplugin.properties default settings file.

Creating an HTML File for Switching

The method for defining an HTML file to select a property file is similar to that commonly used for running an applet, except that a new parameter must be added.

To use a particular property file for the applet, specify the property file in the new parameter of the applet.

Add a new parameter "@JBKPLGPROP" in the parameters passed to the applet (specified by the <PARAM> tag in <OBJECT>...</OBJECT>). As the parameter value, specify the suffix added to the name of the selective property file.

For the detail of creating the HTML file for running the applet, see "2.2.3 How to Create an HTML File for JBK Plugin".

Example

Example of creating an HTML file for Internet Explorer

If the name of the property file is jbkplugin.properties.14, specify the <OBJECT> tag for the applet as follows:

```
<OBJECT CLASSID="CLSID:BEA62964-C40B-11D1-AACA-00A0C9216A67"
  WIDTH=100 HEIGHT=100>
  <PARAM NAME="TYPE" VALUE="application/x-JBK-Plugin">
  <PARAM NAME="@JBKPLGPROP" VALUE="14">
  <PARAM NAME="NAME" VALUE="sample">
  <PARAM NAME="CODE" VALUE="Sample.class">
</OBJECT>
```

- To include two or more applets in the same HTML file, specify the same property file with the @JBKPLGPROP parameter for all applets.

- On the same browser, the user cannot use HTML files that use different property files. If the user want to use those HTML files at a time, launch another browser and use the HTML files separately on each browser.

Point

- Both uppercase and lowercase characters can be specified in the parameter name @JBKPLGPROP.

- If @JBKPLGPROP is omitted in the parameters, the default property file (jbkplugin.properties) is used.

2.3.13 Applet Caching

This section describes the applet caching of JBK Plugin.

By default, JBK Plugin uses the browser cache. As described in '2.3.6 How JBK Plugin Downloads Class Files', when JBK Plugin uses browser functionality for downloading files, they are stored in the browser cache. The browser cache is, however, sometimes cleared because of the limitation of the cache capacity. This causes extra download, though the downloaded file is not actually updated.

To prevent this, JBK Plugin provides its own applet cache. When the applet cache is enabled, the applet is not stored in the browser cache, but stored in the applet cache. JBK Plugin does not automatically clear the applet cache. Therefore, the extra download does not occur. If the user feels inconvenience for the browser cache, we recommend the JBK Plugin applet cache.

Requirements for the Applet Caching

JBK Plugin applet cache is available under the following requirements:

- HTTP protocol and HTTPS protocol are available.

- By default, the following kinds of files can be stored.
  - Class files downloaded from the Web server.
  - JAR files specified in ARCHIVE attribute of the JBK Plugin tag in the HTML file.

    For the detail of the JBK Plugin tag, see '2.2.3 How to Create an HTML File for JBK Plugin'.
- Image files (GIF, JPEG, and so on) downloaded by invoking java.applet.Applet.getImage().

If the user wants to store the other kinds of files into the applet cache, see 'JBK Plugin Applet Caching API'.

Setting for the Applet Caching

To enable the applet cache, code the following line in the jbkplugin.properties file.

```
jbk.plugin.www.plugin_cache.enable=<applet-cache-used-or-not>
jbk.plugin.www.plugin_cache.dir=<applet-cache-folder>
jbk.plugin.www.plugin_cache.dir.freespace=<applet-cache-folder-free-space>
jbk.plugin.www.plugin_cache.update=<applet-cache-update>
jbk.plugin.www.plugin_cache.accelerate=<use-accelerate-mode-in-"newer">
```

Specify the following information in the properties:

- jbk.plugin.www.plugin_cache.enable

  This property specifies whether the applet cache is enabled. As <applet-cache-used-or-not>, code either of the following:

  true

  The applet cache is enabled.

  false

  The applet cache is disabled.

  By default, the value of this property is set to false. When this property is set to false, the following properties are invalid.

- jbk.plugin.www.plugin_cache.dir

  This property specifies the applet cache folder. As <applet-cache-folder>, specify the full path name of the applet cache folder. By default, the applet cache folder is set as follows:

  ```
  ${user.home}\applet_cache
  ```

  ${user.home} means user profile folder.

- jbk.plugin.www.plugin_cache.dir.freespace

  This property specifies the free space of the applet cache folder. If the drive of the applet cache folder does not have enough free space, that is, if the free space is equal to or less than the value of this property, JBK Plugin does not store the downloaded file into the applet cache.

  As <applet-cache-folder-free-space>, the user can specify either of the following:

  - Absolute value

    Numerical value means the free space by bytes. If the user wants to specify the free space by kilobytes, s/he can append 'K' or 'k' to the value. 'M' and 'm' is also available for megabytes.

  - Relative value

    If the user appends '%' to the value, the value is used as the percentage for the total disk capacity.

  By default, the value of this property is set to '1%'.

  Example

  Examples of specifying the size of the allowable space

```
# by bytes
jbk.plugin.www.plugin_cache.dir.freespace=1024
# by kilobytes
jbk.plugin.www.plugin_cache.dir.freespace=256K
# by megabytes
jbk.plugin.www.plugin_cache.dir.freespace=10M
# percentage for the total disk capacity
jbk.plugin.www.plugin_cache.dir.freespace=1%
```

- jbk.plugin.www.plugin_cache.update

  This property specifies the update policy of the applet cache. The following two values can be set.

  newer

    Updates the cache if the modification time of the file on the Web server is newer than that of the file in the applet cache.

  diff

    Updates the cache if the modification time of the file on the Web server is different from that of the file in the applet cache.

  By default, the value of this property is set to 'newer'.

- jbk.plugin.www.plugin_cache.accelerate

  This property specifies to use accelerate mode or not when the update policy of the applet cache is set to "newer". The following two values can be set.

  true

    When connect to the Web server, JBK Plugin issues the HTTP requests with "If-Modified-Since:" header. If the applet is not necessary to update, the amount of the response from the server can be reduced, and the performance is better.

  false

    Issues the HTTP request normally. If cache is available, JBK Plugin stops reading from the server, and reads data from cache file.

  By default, the value of this property is set to 'false'.

How to the Applet Caching Works

The applet cache of JBK Plugin works as follows:

- If the file is not in the applet cache, JBK Plugin downloads it from the Web server and stores in the applet cache.
- If the file is in the applet cache, JBK Plugin checks its modification time and its file size.
  - JBK Plugin checks the modification time according to the update policy specified by 'jbk.plugin.www.plugin_cache.update'. If the file needs to be updated, JBK Plugin downloads it and updates the cache.
  - Also, JBK Plugin checks the file size. If the size of the file in the applet cache is different from that of the file on the Web server, JBK Plugin downloads it and updates the cache.

  In other cases, JBK Plugin does not download the file and load it from the cache.
- When jbk.plugin.www.plugin_cache.accelerate is set to "true", if the cache is available, JBK Plugin asks the Web server with "If-Modified-Since:" header. If the response code is "304 Not Modified", JBK Plugin use the cache.
- JBK Plugin does not delete the file in the applet cache. If the user wants to delete the file in the cache, s/he has to do so manually.
- If the drive of the applet cache folder does not have enough free space (see the setting of 'jbk.plugin.www.plugin_cache.dir.freespace'), JBK Plugin does not store the downloaded file into the applet cache. Even in this case, the file can be downloaded with no problem and the applet runs normally.

JBK Plugin Applet Caching API

To keep the consistency of the applet cache, it must be guaranteed that several threads do not download the same file at the same time. It is guaranteed that Java VM downloads class files, JAR files and image files only from one thread at a time. Therefore, JBK Plugin makes the applet cache enable for only these kinds of files. For the other kinds of files, the user can use JBK Plugin applet caching API to store them into the applet cache. However, the user has to guarantee that those files are downloaded only from one thread at a time.

JBK Plugin applet caching API is as follows:

- Class
    - com.fujitsu.jbk.plugin.browser.www.PluginCacheController
      (extends java.lang.Object)
- Methods
    - getInstance

      returns an instance of PluginCacheController.

      ```
      public static PluginCacheController getInstance()
      ```

      Return Value

      An instance of PluginCacheController

    - getCachedURL

      makes the file of the given URL enable to be stored into the applet cache.

      ```
      public URL getCachedURL(URL origURL)
      ```

      origURL

      the URL of the file to be stored into the applet cache.

      Return Value

      A new URL object that represents the same URL as origURL. The new URL object has the functionality of storing the file into the applet cache. The user can use the new URL object for downloading the file. Then JBK Plugin automatically stores the file into the applet cache.

The class file of the PluginCacheController class is stored in the JBK runtime class directory (the JBK runtime class directory is the "classes" directory below the JBK install directory).

When compiling an applet class using it, check that the class path includes the JBK runtime class directory.

Example

An example of downloading a file

- not using JBK Plugin applet caching API

```
// create a new URL object
URL url = new URL(/* the URL of the file to be downloaded */);
// get a URLConnection
URLConnection conn = url.openConnection();
// get an input stream
InputStream in = conn.getInputStream();

/* downloading the file by the input stream (omitted) */
```

- using JBK Plugin applet caching API

```
// import
import com.fujitsu.jbk.plugin.browser.www.PluginCacheController;
```

```
// creates a new URL object
URL url = new URL(/* the URL of the file to be downloaded */);
```

```
// get a new URL object for applet caching
try {
    url = PluginCacheController.getInstance().getCachedURL(url);
} catch (Throwable ignore) {
    // The exception thrown by getCachedURL() can be ignored.
}

// get a URLConnection
URLConnection conn = url.openConnection();
// get an input stream
InputStream in = conn.getInputStream();

/* downloading the file by the input stream (omitted) */
```

Other Topics for the Applet Caching

- The files stored in the JBK Plugin applet cache are not stored in the browser cache.
- JBK Plugin does not automatically delete the files in the applet cache. If the user wants to delete the files in the applet cache, exit all the browsers that JBK Plugin is running in, and then, delete the files manually.
- The applet caching files will be downloaded in the following timing.
  - For class files, JAR files and image files, JBK Plugin checks the modification time and the file size only at the first time that the file is used in the VM.
  - For the files that uses the applet caching API, JBK Plugin checks the modification time and the file size when getInputStream() (java.net.URLConnection) is invoked for the file.
- HTML files downloaded by showDocument() (java.applet.Applet) are never stored in the applet cache. These HTML files are always stored in the browser cache.
- When the following setting exists in the HTTP response header returned from the Web server, JBK Plugin does not store the file into the applet cache:
  - 'no-cache' in the 'Pragma' field
  - 'no-cache' or 'no-store' in the 'Cache-Control' field

  'max-age' in the 'Cache-Control' field has no effect for the JBK Plugin applet cache.

  For the detail of the HTTP response header, see HTTP/1.1 specification.
- In the applet cache folder, JBK Plugin creates a file for cache management, whose name is 'PluginCacheController.ini'. The user should not edit this file, or the applet cache does not work correctly.
- If the applet caching function is being used in Windows Vista or Windows 7, the user should set the cache folder belongs to the LocalLow folder at user profile folder.
- If GET is used to request the Web server for a file, the file is saved to Applet Caching of JBK Plugin. If POST is used, the file is not saved to this cache.
- If the HTTP header that is returned from the Web server during a file download does not include the Last-Modified field or Content-Length field, the file is not saved to Applet Caching of JBK Plugin.

2.3.14 Download status notification

This section describes the method of download status notification supported by JBK Plugin.

JBK Plugin provides an interface for reporting information about the download of JAR files to the user. You can embed a progress bar or similar item indicating the status of JAR file downloading by creating a class with this interface implemented and specifying that class with jbkplugin.properties.

Download status notification interface

Details of the JBK Plugin download status notification API are as follows.

- Interface:
    - com.fujitsu.jbk.plugin.browser.PluginDownloadNotify
      (extends java.lang.Object)
- Methods:
    - jarLoaded

      This method is called when all class files in the JAR files have been completely loaded for an applet that is to be loaded. If java.net.URLConnection is used in an applet to download files, this method is not invoked. Therefore, explicitly invoke the method from the applet.

      ```
      public abstract void jarLoaded()
      ```

    - setCurrentJar

      This method is invoked when an applet is loaded. It is also invoked when java.net.URLConnection is used in an applet to download files.

      ```
      public abstract void setCurrentJar(java.net.URL jar)
      ```

      jar

      URL of a JAR file to be loaded

    - setJars

      This method delivers an array of the JAR files to be loaded by an applet.

      ```
      public abstract void setJars(java.net.URL[] jars)
      ```

      jars

      Array of URLs of JAR files to be loaded

      ･Applet codebase

      ･JAR files coded in the archive attribute

In addition, the getPluginDownloadNotify() method has been added. This method gets a class by using the com.fujitsu.jbk.plugin.browser.PluginAppletContext interface to implement the PluginDownloadNotify interface.

The class file for the PluginDownloadNotify class is stored in the jar file for the development of the JBK plugin ("classes\jbkstd.jar" in the JBK installation folder). When compiling the class of an applet that uses the PluginAppletContext interface, check whether the jar file for the development of the JBK plugin is included in the classpath.

Specifying a download status notification class

To specify a class with download status notification implemented, code the following line in jbkplugin.properties:

```
jbk.plugin.interface.download_notify=<name-of-a-class-with-download-status-notification-implemented>
```

For <name-of-a-class-with-download-status-notification-implemented>, specify the name of a class that includes a package name.

Using download status notification

Use download status notification as follows:

1. Create a class with the PluginDownloadNotify interface implemented.
2. Place the class created in step 1 on the client. A class path needs to be set for the class .

   To set the class path, add it to the CLASSPATH environment variable or use jbk.plugin.vmoption of jbkplugin.properties.

3. Add the jbk.plugin.interface.download_notify property to jbkplugin.properties to set the name of the class.

Note

When an implemented class internally uses the Class.getResource() method, it may fail to acquire resources. If such a failure occurs, edit the policy file to add the java.io.FilePermission read attribute to the class.

```
Example)
grant {
    java.io.FilePermission "C:\\JBKPLG\\classes\\Progress.jar", "read";
};
```

Note

When an implemented class generates a new thread, it may throw AccessControlException. If such a failure occurs, edit the policy file to add java.lang.RuntimePermission to the class.

```
Example)
grant codebase "file:/C:/JBKPLG/classes/Progress.jar" {
    java.lang.RuntimePermission "modifyThread";
    java.lang.RuntimePermission "modifyThreadGroup";
};
```

Alternatively, you can prevent the failure by adding the implemented class as follows to the boot class path, not to the regular class path:

- Using the -Xbootclasspath option, specify the path to the execution class.
- Create a classes folder under the JRE folder, then store the implemented class in the classes folder.

  (The JRE classes folder has been specified in the boot class path by default.)

2.3.15 JavaScript Communication

This section describes the JavaScript communication of JBK Plugin.

JBK Plugin supports the JavaScript communication with Internet Explorer.

This section describes the following items:

- 2.3.15.1 Calling JavaScript from an applet
- 2.3.15.2 Calling a method of an applet from JavaScript
- 2.3.15.3 Accessing cookie information

2.3.15.1 Calling JavaScript from an applet

To call JavaScript from Java applet, it is necessary that Java communicates with JavaScript. JBK Plugin enables Java to communication with JavaScript using netscape.javascript.JSObject object, which is instance of Java wrapper class, in Internet Explorer.

- Class:
  - netscape.javascript.JSObject
    (extends java.lang.Object)
- Methods:
  - getWindow

    Gets the JSObject of the window (Window object of JavaScript) containing the given applet. This method takes only java.applet.Applet object as the parameter.

```
public static JSObject getWindow(java.applet.Applet a)
```

- call

  Calls the JavaScript method. Equivalent to "this.methodName(args[0], args[1], ...)" in JavaScript.

  ```
  public Object call(String methodName, Object args[])
  ```

- eval

  Evaluates the JavaScript expression.

  ```
  public Object eval(String s)
  ```

- getMember

  Retrieves the named member of the JavaScript object.

  ```
  public Object getMember(String name)
  ```

- getSlot

  Retrieves the indexed member of the JavaScript object.

  ```
  public Object getSlot(int index)
  ```

- setMember

  Sets the named member of a JavaScript object.

  ```
  public void setMember(String name, Object value)
  ```

- setSlot

  Sets the indexed member of the JavaScript object.

  ```
  public void setSlot(int index, Object value)
  ```

- toString

  Converts the JSObject to a String.

  ```
  public String toString()
  ```

- removeMember

  Not supported.

  ```
  public void removeMember(String name)
  ```

  In the specification of Java packages for LiveConnect, removeMember method exists, but is not supported in Internet Explorer.

- Supported objects by JSObject

  The following JavaScript objects can be accessed with JBK Plugin via JSObject.

  JBK Plugin can access the following JavaScript objects via JSObject:

  ```
  Anchor/Applet/Array/Boolean/Button/Checkbox/Date/document/Element/Fileupload/Form/
  Frame/Function/Hidden/History/Image/Link/Location/Math/navigator/Number/Object/
  Option/Password/Radio/RegExp/Reset/screen/Select/String/Submit/Text/TextArea/Window
  ```

  - arguments.length, arguments[], and caller properties of a Function object cannot be used.
  - If the applet calls a JavaScript method whose parameters are instance of JavaScript object, e.g. concat() method of an Array object, use eval() method of JSObject to process exactly.

- How to use JSObject

The following is the example to call JavaScript from the applet using JSObject:

```
import netscape.javascript.*;
import java.applet.*;

public class Applet1 extends Applet {
     public void init() {

            JSObject win = JSObject.getWindow(this);

            try{
                 // call alert method of JavaScript Window object
                 win.call("alert", new Object[]{"Test"});

                 // set the status property of JavaScript Window object
                 win.eval("this.status=\"Hello World\"");

                 /* document.forms[0].elements[0] is <INPUT TYPE=button>. */

                 // refer the button, and set the value property to "Button".
                 JSObject doc = (JSObject) win.getMember("document");
                 JSObject forms = (JSObject) doc.getMember("forms");
                 JSObject aform = (JSObject) forms.getMember("0");
                 JSObject elements = (JSObject) aform.getMember("elements");
                 JSObject script_button = (JSObject) elements.getMember("0");
                 script_button.setMember("value", "Button");

            // exception handling for JSObject
            }catch(JSException e){
                 e.printStackTrace();
            }
     }
}
```

The following shows an example of HTML in which a form is called by the above applet.

[HTML description]

```
<HTML>
<HEAD>
<SCRIPT src="JBKTag.js" type="text/javascript" language="JavaScript">
</SCRIPT>
</HEAD>
<BODY>

<FORM>
  <INPUT TYPE=button value="button">
</FORM>

<SCRIPT type="text/javascript" language="JavaScript">
<!--
    //Use JavaScript and write tags to display applet.
    showapplet();
//-->
</SCRIPT>
</BODY>
</HTML>
```

[JavaScript code(JBKTag.js)]

```
function showapplet() {

    document.writeln('<OBJECT CLASSID="CLSID:BEA62964-C40B-11D1-AACA-00A0C9216A67" WIDTH=100
HEIGHT=100>');
    document.writeln('<PARAM NAME="TYPE" VALUE="application/x-JBK-Plugin">');
    document.writeln('<PARAM NAME="NAME" VALUE="sample">');
    document.writeln('<PARAM NAME="CODE" VALUE="Applet1.class">');
    document.writeln('<PARAM NAME="color" VALUE="blue">');
    document.writeln('<PARAM NAME="useDefault" VALUE="true">');

    document.writeln('</OBJECT>');
}
```

The class file for the JSObject class and JSException class is stored in the jar file for the development of the JBK plugin ("classes\jbkstd.jar" in the JBK installation folder). When compiling the class of an applet that uses the PluginAppletContext interface, check whether the jar file for the development of the JBK plugin is included in the classpath.

Note

Observe the following notes derived from Internet Explorer limitations when using JavaScript Communication in Internet Explorer:

- The user cannot use setMember method to set the properties of objects excluding window and document objects.
- The user cannot use eval emthod excluding window objects.
- Because of the security restrictions of Windows XP Service Pack 2, you must use absolute path instead of relative path to designate local files under the following conditions.
    - When opening a local file by using open method of the window object.
    - When setting a local file by using location property of the window object.
- You cannot use getMember method to get the properties of some objects.
- The type of the return value of the following methods of the Date object is Double in Internet Explorer in Netscape though it is Integer.
    - getYear
    - getMonth
    - getDate
    - getHours
    - getMinutes
    - getSeconds
    - getDay
    - getTimezoneOffset

Information

JSObject also supports the array of JavaScript. To get an array from applet using JavaScript, please refer to the following code.

```
import netscape.javascript.*;
import java.applet.*;

public class Applet1 extends Applet {

     public void init() {
            JSObject win = JSObject.getWindow(this);
            /* Array is set as 'arrayname = new Array("0", "1", "2", ...)' */

            // get JavaScript Array object
            JSObject array = (JSObject)win.getMember("arrayname");
            // get a number of index
            (Integer)array.getMember("length");
            // get the first member of the array
            (String)array.getMember("0");
     }
}
```

2.3.15.2 Calling a method of an applet from JavaScript

JBK Plugin allows calling a public method of an applet from JavaScript.

If the calling public method is overloaded, it cannot be guaranteed which of the methods is called.

- Setting for Calling Method

  To call a method of the applet from JavaScript, put the following line in the jbkplugin.properties file:

```
jbk.plugin.sw.script.enable =<method calling-used-or-not>
```

  Specify true or false to <method calling-used-or-not>.

  true

    Calling applet method is enabled.

  false

    Calling applet method is disabled.

  By default, the value of this property is set to false.

- How to call a method

  To call a method of an applet from JavaScript, Please refer the following code.

  Example: Calling Hello() method of the applet when the button1 is pushed.

  [HTML description]

```
<OBJECT classid="CLSID:BEA62964-C40B-11D1-AACA-00A0C9216A67" width="160" height="120">
            <PARAM NAME="CODE" VALUE="Applet1.class">
            <PARAM NAME="TYPE" VALUE="application/x-JBK-Plugin">
            <PARAM NAME="ARCHIVE" VALUE="TestApplet.jar">
            <PARAM NAME="useDefault" VALUE="true">
    </OBJECT>

    <INPUT TYPE="button" NAME="button1" VALUE="Button" onClick="sample()">

    <Script language="JavaScript" FOR="window">
    <!---
        function sample() { alert(document.applets[0].Hello("XXXX")); }
    //---->
    </Script>
```

  [Applet code]

```
   import java.applet.*;

    public class Applet1 extends Applet{
           public void init() { }
```

```
        // return the formatted String
        public String Hello(String name){
            String retVal = new String("Hello ");
            retVal += name ;retVal += "!!" ;return (retVal);
        }
    }
```

- Type conversion from JavaScript to applet

When JavaScript calls a method of an applet, JavaScript sends the argument to the applet. JavaScript value is converted to Java value. The conversion specifications are as follows:

Table 2.3 JavaScript to Java applet type conversion

argument of applet's method	Type of the value sent from JavaScript	
	Type of JavaScript	Results
Byte/byte	String values	- From "-128" to "127": convert to byte value from -128 to 127 - Out of the range or invalid numerical strings: convert to 0 - Positive floating decimal string: round up to the integer value - Negative floating decimal string: round up to the integer value
	Number values	- From -128 to 127": convert to byte value from -128 to 127 - Out of the range: convert to 0 - Positive floating decimal: round up to the integer value - Negative floating decimal: round up to the integer value
	Boolean values	- true: converted to -1 - false: 0
	null value	- converted to 0
	Object	- converted to 0
Character/char	String values	- From "0" to "65535": convert to char value from 0 to 65535 - Out of the range or invalid numerical strings: convert to char value 0 - Positive floating decimal string: round up to the integer value
	Number values	- From 0 to 65535: convert to char value from 0 to 65535 - Out of the range: convert to char value 0 - Positive floating decimal: round up to the integer value
	Boolean values	- true: converted to char value 65535 - false: converted to char value 0
	null value	- converted to 0
	Object	- converted to 0
Short/short	String values	- From "-32768" to "32767": convert to short value from -32768 to 32767 - Out of the range or invalid numerical strings: convert to short value 0 - Positive floating decimal: round up to the integer value - Negative floating decimal: round up to the integer value
	Number values	- From -32768 to 32767: convert to short value from -32768 to 32767 - Out of the range: convert to short value 0

argument of applet's method	Type of the value sent from JavaScript	
	Type of JavaScript	Results
		- Positive floating decimal: round up to the integer value - Negative floating decimal: round up to the integer value
	Boolean values	- true: converted to short value -1 - false: converted to short value 0
	null value	- converted to 0
	Object	- converted to 0
Integer/int	String values	- From "-2147483648" to "2147483647": convert to int value from -2147483648 to 2147483647 - Out of the range or invalid numerical strings: convert to int value 0 - Positive floating decimal: round up to the integer value - Negative floating decimal: round up to the integer value
	Number values	- From -2147483648 to 2147483647: convert to int value from -2147483648 to 2147483647 - Out of the range: convert to int value 0 - Positive floating decimal: round up to the integer value - Negative floating decimal: round up to the integer value
	Boolean values	- true: converted to int value -1 - false: converted to int value 0
	null value	- converted to 0
	Object	- converted to 0
Long/long	String values	- From "-2147483648" to "2147483647": convert to long value from -2147483648 to 2147483647 - Out of the range or invalid numerical strings: convert to long value 0 - Positive floating decimal: round up to the integer value - Negative floating decimal: round up to the integer value
	Number values	- From -2147483648 to 2147483647: convert to long value from -2147483648 to 2147483647 - Out of the range: convert to long value 0 - Positive floating decimal: round up to the integer value - Negative floating decimal: round up to the integer value *If long value is beyond the limits of int value, the result of conversion becomes 0.
	Boolean values	- true: converted to long value -1 - false: converted to long value 0
	null value	- converted to 0
	Object	- converted to 0
Boolean/boolean	String values	- "0": converted to boolean value false - Digit string except "0": converted to boolean value true

argument of applet's method	Type of the value sent from JavaScript	
	Type of JavaScript	Results
		- Other string: converted to boolean value false
	Number values	- 0: converted to boolean value false - except 0: converted to boolean value true
	Boolean values	- true: boolean value true - false: boolean value false
	null value	- converted to boolean value false
	Object	- converted to boolean value false
Float/float	String values	- From "±1.4e-45" to "±3.4028235e+38": converted to float value from ±1.4e-45 to ±3.4028235e+38 - Out of the range or invalid numerical strings: converted to float value 0.0
	Number values	- From ±1.4e-45 to ±3.4028235e+38: converted to float value from ±1.4e-45 to ±3.4028235e+38 - Out of the range: converted to float value 0.0
	Boolean values	- true: converted to float value -1.0 - false: converted to float value 0.0
	null value	- converted to 0.0
	Object	- converted to 0.0
Double/double	String values	- From "±4.9e-324" to "±1.7976931348623157e+308": converted to double value from ±4.9e-324 to ±1.7976931348623157e+308 - Out of the range or invalid numerical strings: converted to double value 0.0
	Number values	- From ±4.9e-324 to ±1.7976931348623157e+308: converted to double value from ±4.9e-324 to ±1.7976931348623157e+308 - Out of the range: converted to double value 0.0
	Boolean values	- true: converted to double value -1.0 - false: converted to double value 0.0
	null value	- converted to 0.0
	Object	- converted to 0.0
String	String values	- Any string: converted to Java String
	Number values	- Any number: converted to Java String
	Boolean values	- true: converted to String "-1" - false: converted to String "0"
	null value	- converted to null Object
	Object	- converted to Java String
JSObject	String values	- Any string: converted to JSObject null
	Number values	- Any digit string: converted to JSObject null
	Boolean values	- converted to JSObject null
	null value	- converted to JSObject null
	Object	- converted to JSObject

argument of applet's method	Type of the value sent from JavaScript	
	Type of JavaScript	Results
		*Any method of JSObject class cannot be used.

- Type conversion from the return value of applet method to JavaScript

  When an applet returns the result of calling method to JavaScript, the return value is converted to JavaScript value. These conversion specifications are as follows:

Table 2.4 [Java applet to JavaScript type conversion]

Return value type of applet method	JavaScript value
Void	null value
Byte/byte	Number value
Character/char	Number value
Short/short	Number value
Integer/int	Number value
Long/long	Number value, the range is from -2147483648 to 2147483647
Boolean/boolean	Boolean value true is true, false is false.
Float/float	Number value
Double/double	Number value
String	String value
JSObject	Object value

2.3.15.3 Accessing cookie information

This section describes how to access cookie information.

An applet can access cookie by using cookie property of JavaScript document object which is referred via JSObject.

Example

Gets the cookie information from the browser, and sends it to the server.

```
import java.applet.*;
import java.net.*;
import netscape.javascript.*;

public class Applet1 extends Applet {
    public void init(){
        JSObject win = JSObject.getWindow(this);

        // refer the cookie property of JavaScript document object
        JSObject doc = (JSObject)win.getMember("document");
        String cookie = (String)doc.getMember("cookie");

        try{
            /* set the URL to send the cookie */
            // get codebase of the applet
            URL url = getCodeBase();
            String strurl = url.toString();

            // set URL
            /* in this case, "/servlet/cookie" */
            strurl = strurl + "/servlet/cookie";
            url = new URL(strurl);

             // set the HTTP request property
             HttpURLConnection urlCon = (HttpURLConnection)url.openConnection();
             urlCon.setRequestProperty("cookie", cookie);

             urlCon.disconnect();

             /* Describe the procedure using HttpURLConnection */

        }catch(Exception e){
             e.printStackTrace();
        }
    }
}
```

2.4 Relationship between the Java VM and the Process of the Browser

This section describes the relationship between a browser process and the Java VM started by JBK Plugin.

JBK Plugin starts a single Java VM for each browser process. Once started, the Java VM stays until the corresponding browser process terminates. When an applet is executed by using JBK Plugin, the relationship between the browser process and the Java VM is as described below:

- When the applet is terminated on the browser, the Java VM started by JBK Plugin enters the wait state in the browser process. When the applet is executed again on the same browser, JBK Plugin does not restart another Java VM but uses the waiting Java VM to execute the applet.
- When the browser process terminates, the Java VM started by JBK Plugin in the browser terminates as well.
- When a single browser process is started and multiple applets are executed on that browser by using JBK Plugin, those applets are executed by using the same Java VM.
- When multiple browser processes are started, JBK Plugin starts a single Java VM for each browser. An applet on each browser is executed by the Java VM started for the corresponding browser.

Using Internet Explorer in the Active Desktop

Internet Explorer in the active desktop environment runs as a part of a process of the Window Explorer. When Windows is started, a single Explorer process starts and stays until Windows terminates. Therefore, when JBK Plugin is used on Internet Explorer in the active desktop environment, the relationship between Internet Explorer and the Java VM is as described below:

- When used on Internet Explorer in the active desktop environment, JBK Plugin starts a Java VM in a process of Explorer. When Internet Explorer terminates, the Java VM started by JBK Plugin enters the wait state within the Explorer process.
- When Internet Explorer is restarted, JBK Plugin does not restart another Java VM but uses Java VM held in the wait state within the Explorer process to execute an applet.
- When multiple Internet Explorers are started, JBK Plugin executes applets by using the same Java VM on those Internet Explorers.

It is necessary to start Internet Explorer in the process besides the process of the Explorer start/end Java VM on active desktop according to start/end of Internet Explorer.

2.5 How JBK Plugin Works to Destroy Applets

This section describes the processing at the termination of an applet executed with JBK Plugin.

An applet executed on a browser terminates on the following occasions:

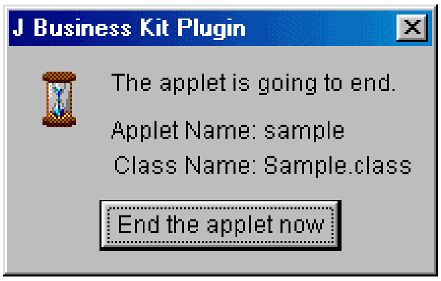
- When another HTML file is loaded
- When the browser window is closed

At the termination of an applet, the stop() method and destroy() method of the applet class are invoked.

When closing the browser window, JBK Plugin terminates an applet on the window, and then closes the browser window. This ensures that the processing for terminating the applet is invoked before the browser terminates.

If the applet termination processing takes time, JBK Plugin displays a message box as shown below when the browser window is closed:

Figure 2.3 Message box indicating that the applet termination processing is in progress



J Business Kit Plugin

The applet is going to end.

Applet Name: sample

Class Name: Sample.class

The message box indicates that the termination processing for the applet shown in the message box is in progress. When the applet termination processing is completed, the message box automatically disappears.

[End the applet now] button

Clicking the [End the applet now] button shown above halts the sequence for terminating the applet and forcibly terminates the applet. Forced termination does not assure successful completion of the stop() method and destroy() method.

Use the [End the applet now] button only to quit an applet when an error makes it impossible to quit it any other way.

How to disable the message box

You can disable the message box shown above so that the browser window can be closed immediately. To do so, modify the following line in jbkplugin.properties:

```
jbk.plugin.sw.applet.delay=<wait-for-termination-of-applet>
```

For <wait-for-termination-of-applet>, specify either of the following:

true

To allow the browser window to wait until the applet termination sequence has been completed.

false

To force the browser window to close immediately without waiting for termination of the applet.

The jbk.plugin.sw.applet.delay default value is true. Setting it to "false" does not assure successful completion of the stop() method and destroy() method for the applet.

2.6 Printing Applets

Printing Applet is a function for printing a Web page together with any applet included on the Web page.

Note the following about using Printing Applet:

- If the width and height attributes of an HTML tag (<OBJECT> or <EMBED>) for JBK Plugin are specified as percentages for display in Web browsers, the aspect ratio of an applet image may not agree with the aspect ratio of the printout. In this event, the applet printout appears to be squeezed vertically or horizontally.

  When using Printing Applet, specify fixed values rather than percentages for the width and height attributes of HTML tags for JBK Plugin.

- The image data displayed on an applet or GUI component may appear to be coarse or enlarged, depending on the operating system and/or printer driver used.

- The following methods of java.awt.Graphics class are not supported in printing. These methods do not output any graphics for printing.
    - copyArea()

2.7 Other Topics

Importing Internet Explorer add-ons

Disable add-ons before running an Applet. If the add-ons are enabled, unexpected termination of Internet Explorer may occur.

Page jump after clicking the taskbar

Internet Explorer 9 or later, if the dialog page inherited from JFDialog is displayed, the focus will be lost when the page jumps to the original dialog page after t clicking the taskbar.

The destruction of an applet

The window is now made to close immediately when the window is closed in Internet Explorer 9. Therefore, in the following cases, the page will disappear even when the destruction of an applet has been canceled. Also, the process will remain even after the page has disappeared:

- The applet implements destroyRequested method, which cancels termination of the browser, and
- When a tab or window with the applet mentioned above is closed

[Action]

Please implement the onbeforeunload handler(to window) on the HTML that embedds the applet.

```
<head>
<script>
function checkClose() {
}
window.onbeforeunload = new Function("checkClose()");
</script>
</head>
```

For Internet Explorer 10, once the destroyRequested() has been used to return false, a window cannot be closed after this method is implemented, even though the window is subsequently allowed to be closed by returning true. When using the destroyRequested in Internet Explorer 10, do not terminate the processing through returning true.

Applet's Parent Window

When an applet's dialog is created and displayed with its parent window set as Parent Window, the icon shown will be different if the dialog can be resized.

Using showDocument() Method of java.applet.Applet Class

The user should not set "" (an empty string) to the second argument of the showDocument() method. Otherwise, JBK Plugin may not be able to define the window to show the document uniquely.

Page Zoom

Page Zoom does not change the size of applet.

The user.home Value in JBK Plugin

JBK Plugin uses the user.home value of the JavaSE as it is. The user.home value of the JavaSE is as shown below.

```
<Windows-drive>\Documents and Settings\<user-name>
```

For instance, if <Windows-drive> is "C:" and <user-name> is "foo", the user.home value is "C:\Documents and Settings\foo".

JAR file caching functionality in the JavaSE

JavaSE has its own JAR file cache functionality. The downloaded JAR files are temporary stored in the Windows temporary directory. The cached JAR file is named "jar_cacheXXXXX.tmp" (XXXXX is a number.)

See below for the relationship between the JAR file caching and the download methods of JBK Plugin (see also '2.3.6 How JBK Plugin Downloads Class Files' and '2.3.7 USING THE HTTPS PROTOCOL'.)

- Using 'java' for the download method

  1. First, JBK Plugin downloads a JAR file from the Web server by using the Java networking class.
  2. By the JAR file caching functionality in the JavaSE, the downloaded JAR file is cached in the Windows temporary directory. JBK Plugin loads the JAR file from the cache next time it is used. The cache is valid until the user quits the browser.
  3. When the user quits the browser, the JavaSE deletes the cached JAR file. Next time the user starts the browser, JBK Plugin downloads the JAR file again.

- Using 'hybrid' or 'native' for the download method

  1. First, JBK Plugin downloads a JAR file from the Web server by using the networking functionality of the browser. If the browser cache is available and the file in the Web server is not updated, JBK Plugin loads the JAR file in the browser cache instead.
  2. By the JAR file caching functionality in the JavaSE, the downloaded JAR file is cached in the Windows temporary directory whether the browser cache is available or not. JBK Plugin loads the JAR file from the VM cache next time it is used. The VM cache is valid until the user quits the browser.
  3. When the user quits the browser, the JavaSE deletes the JAR file that was cached. The browser cache, however, is yet valid. Next time the user starts the browser, JBK Plugin loads the JAR file from the browser cache, if available.

In the case the browser terminates abnormally, the cached JAR file by the JavaSE may remain in the Windows temporary directory. In such a case, please delete those files manually.

Operations in the finalize() method

In finalize(), the user should not write print method such as System.out.println() and System.err.println(); otherwise, during garbage collection, JBK Plugin may occasionally stop its execution. If the user needs such operations, s/he should do the operations only in debugging.

Difference between applet download modes

There are some difference between using Network communication classes and using the browser to download applets.

Download by	Java network communication classes (API)	Browser
Authentication	The java.net.Authenticator class implemented in JBK Plugin handles authentication.	Downloading depends on how the browser handles authentication.
Cookie	A Java communication handling class implemented in JBK Plugin obtains information about the cookie from the browser, then adds it to the HTTP header. (*1)(*5)	Downloading depends on the settings of the browser you are using. (*2)
Proxy	As described below, the value specified for proxy in the JBK Plugin property file determines how proxy is handled. - disable: Connection is direct. - enable: The proxy specified in the property file is used. If no proxy server has been specified, the browser's proxy settings are used. (*3)(*4) - Nothing specified: The browser's proxy settings are used. (*3)	The browser's proxy settings are used.

- *1) Not added when the Cookie header is added by the applet.
- *2) Disabled even when the Cookie header is added by the applet.

- *3) The proxy automatic setting file is supported with the following restrictions:
  - The proxy automatic setting file is not supported for HTTPS communication.
  - The dnsResolve function always returns an empty character string when the address of the host is not an IP address.
  - The isResolvable function always returns "false" when the address of the host is not an IP address.
  - The isInNet function always returns "false" when the address of the host is not an IP address.
- *4) When a SOCKS host has been set up, connection is made through the SOCKS host under the following conditions:
  - Neither HTTP proxy nor HTTPS proxy has been enabled, and a URL that is not included in the list of URLs that are not to be handled by proxy is specified.
  - The proxy automatic setting file has responded to a connection through the SOCKS host.
- *5) Cookies will not be added until the applet is loaded.

[download mode difference in each Protocol]

<table>
<tr><th>Protocol</th><th>Download Mode</th><th>Plugin Caching</th><th>Class File</th><th>JAR File</th><th>Image File(*1)</th><th>Other File(*2)</th></tr>
<tr><td rowspan="6">HTTP/HTTPS</td><td rowspan="2">java</td><td>disable</td><td colspan="4" rowspan="2">Java API</td></tr>
<tr><td>enable</td></tr>
<tr><td rowspan="2">hybrid</td><td>disable</td><td colspan="2">Browser function</td><td colspan="2">Java API</td></tr>
<tr><td>enable</td><td colspan="3">Browser function</td><td>Java API</td></tr>
<tr><td rowspan="2">native</td><td>disable</td><td colspan="4" rowspan="2">Browser function</td></tr>
<tr><td>enable</td></tr>
</table>

- *1) Image files (*.gif, *,jpg), which is downloaded from Web server using getImage() method of java.applet.Applet class.
- *2) Property file (*.properties) or another file which applet downloads from Web server using URL class.

Notes on JavaScript Communication

When an Applet is calling JavaScript, exclusive control is involved in Java calling methods. The JBK Plugin will sometimes call JavaScript internally when Java API are used for HTTP communications, as shown in "Difference between applet download modes". When an Applet is calling JavaScript, deadlock may occur if JavaScript has called the methods for the Java HTTP communications. For these applications, please use the browser function for the download method.

2.8 Error Messages

This section describes error messages that can be output when JBK Plugin is used and the actions to be taken in response to the output messages.

- When JBK Plugin is used, an error message may be output by the browser or an exception occurring during the execution of the Java class may be output, as well as the error messages described below. For those messages, see the manual of your browser or a JDK or JRE document.
- If "ERROR: fail to load the applet." is displayed in the applet area, invoke the Java console and check whether an exception was thrown. If FileNotFoundException is thrown, check the applet location and the applet path.

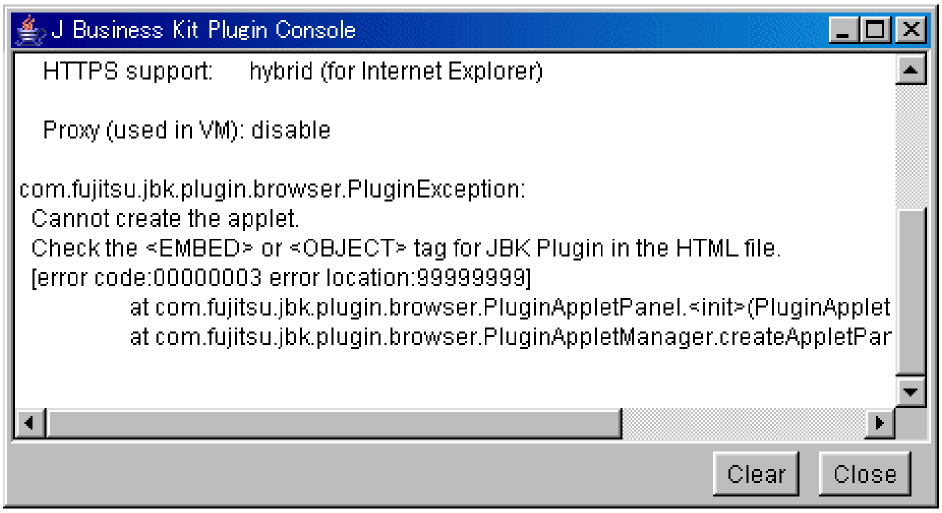
Locations Where the Error Message Is Output

If an error occurs when JBK Plugin is executed, the error message is output as follows:

- If the error occurs in a Java class, the error message is displayed on the Java console in the Java exception format.

- Any other error is displayed in a message dialog.

Figure 2.4 Sample error message displayed in the Java exception format

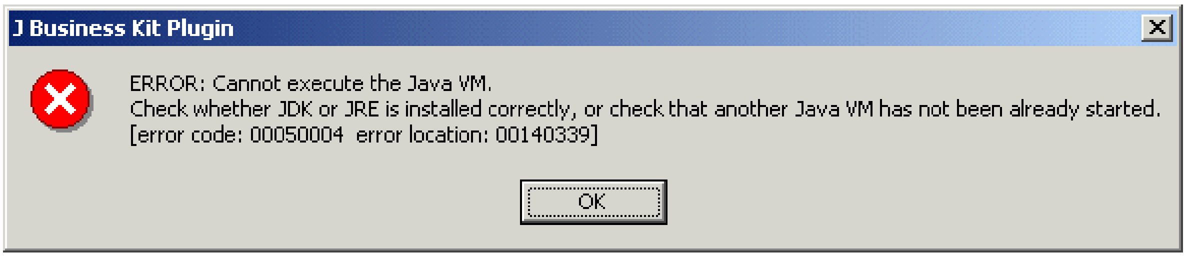


```
  HTTPS support:     hybrid (for Internet Explorer)

  Proxy (used in VM): disable

com.fujitsu.jbk.plugin.browser.PluginException:
 Cannot create the applet.
 Check the <EMBED> or <OBJECT> tag for JBK Plugin in the HTML file.
 [error code:00000003 error location:99999999]
         at com.fujitsu.jbk.plugin.browser.PluginAppletPanel.<init>(PluginApplet
         at com.fujitsu.jbk.plugin.browser.PluginAppletManager.createAppletPar
```

Figure 2.5 Sample error message displayed in a message dialog



ERROR: Cannot execute the Java VM.
Check whether JDK or JRE is installed correctly, or check that another Java VM has not been already started.
[error code: 00050004 error location: 00140339]

Error Message Format

[Java exception output format]

```
com.fujitsu.jbk.plugin.browser.PluginException:
  (Message indicating the problem)
  (Message indicating the cause and the action to be taken)
  (Error codes indicating the error detail)
      Exception trace information
       :
```

[Message dialog output format]

```
Error:(Message indicating the problem)
(Message indicating the cause and the action to be taken)
(Error codes indicating the error detail)
```

- Message indicating the problem

  This message indicates the current problem. For each message, see 'Messages Indicating the Problem' below.

- Message indicating the cause and the action to be taken

  This message indicates the cause of the current error and the action to be taken. For each message, see 'Message Indicating the Cause and the Action to be Taken' below.

- Error codes indicating the error detail

  These error codes indicate the details of the current error and the location of the error in the modules of JBK Plugin. These codes are used when the error must be investigated.

Messages Indicating the Problem

Any of the following messages is displayed as the message indicating the problem:

[Message] Cannot show the HTML document.

[Problem] When the showDocument() method of the java.applet.Applet class was invoked, an error occurred in JBK Plugin.

[Message] Cannot execute the Java VM.

[Problem] An error occurred when the Java VM was started.

[Message] Cannot create the Java console.

[Problem] The Java console of JBK Plugin could not be created. This problem can occur also if the Java console is set to be hidden.

It also occurs when VGA-compatible mode (16 colors) is set in the screen properties.

[Message] Cannot display the Java console.

[Problem] An error occurred when the Java console of JBK Plugin was attempted to be displayed.

[Message] Cannot initialize JBK Plugin.

[Problem] An error occurred in the initialization of JBK Plugin.

[Message] Cannot read the JBK Plugin property file.

[Problem] An error occurred when the JBK Plugin property file (jbkplugin.properties) was being read.

[Message] Cannot connect to the Web server.

[Problem] An error occurred when JBK Plugin was downloading the applet class file or the other files.

[Message] Cannot post to the Web server.

[Problem] An error occurred when the applet was posting data to the Web server by java.net.URL class.

[Message] Cannot create the applet.

[Problem] An error occurred when the applet was being created.

[Message] Cannot destroy the applet.

[Problem] An error occurred when the applet was being discarded.

[Message] Cannot retrieve the parameters of the applet.

[Problem] An error occurred when an applet parameter coded in the HTML file was being obtained. This problem might occur only when Internet Explorer is used.

[Message] Cannot resize the applet.

[Problem] An error occurred when the applet was being re-sized.

[Message] Cannot show the status message.

[Problem] When the showStatus() method of the java.applet.Applet class was invoked, an error occurred in JBK Plugin.

[Message] Cannot retrieve the proxy setting of the browser.

[Problem] An error occurred when the proxy information set for the browser was being obtained.

Message Indicating the Cause and the Action to be Taken

Any of the following messages is displayed indicating the cause of the error and the action to be taken:

[Message] Check the <EMBED> or <OBJECT> tag for JBK Plugin in the HTML file.

[Cause] Not all required attributes are specified in the HTML file for using JBK Plugin.

[Action] The <EMBED> or <OBJECT> tag for JBK Plugin requires the following attributes:

- CODE - Class of the applet
- WIDTH - Width of the applet
- HEIGHT - Height of the applet

Check whether these attributes are correctly specified. For the coding of the HTML file for using JBK Plugin, see '2.2.3 How to Create an HTML File for JBK Plugin'.

[Message] Check whether Internet Explorer is installed correctly.

[Cause] An error occurred when JBK Plugin invoked the plug-in API of Internet Explorer.

[Action] The following are the actions to be taken to correct the error:

- Check whether Internet Explorer has been correctly installed.
- Check that an old version of Internet Explorer is not used.

[Message] Check whether JBK Plugin is installed correctly.

[Cause] This error occurs when the environment for executing JBK Plugin is not correctly set.

[Action] The following are the actions to be taken to correct the error:

- Check whether JBK Plugin is correctly installed.

  For environment setting for using JBK Plugin, see '2.2.1 JBK Plugin Setup'.

[Message] Check the setting in the JBK Plugin property file.

[Cause] This error occurs if the setting of JDK or JRE to be used as a Java VM cannot be obtained.

[Action] The following are the actions to be taken to correct the error:

- Check whether the JBK Plugin property file (jbkplugin.properties) exists.

  The jbkplugin.properties file is placed in the "classes" directory below the JBK runtime install directory.
- If the user specifies the JDK or JRE to be used, check whether the specification is correctly coded in the jbkplugin.properties file.
- If the user does not specify the JDK or JRE to be used, check whether JBK runtime has been correctly installed. JBK Plugin uses the JDK or JRE specified at the installation of JBK as a Java VM.

For the specification of the JDK or JRE to be used as a Java VM, see '2.3.1 Specifying the Java VM Used in JBK Plugin'.

[Message] Internal error.

[Cause] An internal error occurred when JBK Plugin was executed.

[Action] Notify a Fujitsu systems engineer.

[Message] Check whether the correct URL is specified and whether the Web server is available.

[Cause] This error occurred when JBK Plugin downloaded files by using the networking functionality of the browser.

[Action] The following are the actions to be taken to correct the error:

- Check whether the correct URL
- Check whether the Web server is available.
- If you use a proxy server, check whether the proxy server is available.

[Message] Check whether the current property file specified by parameter for applet is different from previous.

[Cause] This error occurred when two applets are going to execute in the same browser but each applet uses different property file.

[Action] The following are the actions to be taken to correct the error:

- Check whether some applet has '@JBKPLGPROP' parameter and some does not have.
- Check whether the '@JBKPLGPROP' parameters of all applets have the same value.

For the detail, see '2.3.12 Switching the Property Files'.

[Message] Check whether JDK or JRE is installed correctly, or check that another Java VM has not been already started.

[Cause] This error occurs if the Java VM cannot be correctly executed.

[Action] The following are the actions to be taken to correct the error:

- If the user specifies the JDK or JRE to be used
    - Check whether the JDK or JRE to be used is correctly specified in the JBK Plugin property file (jbkplugin.properties).
    - Check whether the specified JDK or JRE has been correctly installed.
- If the user does not specify the JDK or JRE to be used
    - Check whether JBK runtime has been correctly installed.
    - Check whether the JDK or JRE specified at the installation of JBK has been correctly installed.

For the specification of the JDK or JRE to be used as a Java VM, see '2.3.1 Specifying the Java VM Used in JBK Plugin'.

Also consider the action below:

- Check whether another Java VM has been already started in this process.
- Check that the CLASSPATH environment variable does not include a path to a class whose version differs from that of the JDK or JRE to be used.

For the class path used by JBK Plugin, see 'Setting Environment Variables'.

[Message] Memory Overflow.

[Cause] A sufficient memory space for executing the applet using JBK Plugin could not be obtained.

[Action] Take either of the following actions:

- Terminate another active application, then start the browser again.
- Restart the system.

[Message] The user cancelled the connection with the Web server.

[Cause] This message indicates that, while JBK Plugin was downloading files, the user stopped the download by pressing the Stop button of the browser.

[Action] Reload the HTML file shown in the browser window by pressing the Reload button of the browser. Then, JBK Plugin reloads the applet again.

[Message] Unexpected error.

[Cause] An unexpected error occurred when JBK Plugin was executed.

[Action] Notify a Fujitsu systems engineer.

2.9 jbkplugin.properties

The table below lists the setting in the JBK Plugin property file (jbkplugin.properties). For each setting, see '2.3 How to Use'.

Property	Key	Value and its description
The Java VM used in JBK Plugin	jbk.plugin.javahome	Specifies the install directory of the JDK or JRE to be used as a Java VM by JBK Plugin. Initial value: None (JDK or JRE specified at the installation of JBK is used.)
Java VM startup options	jbk.plugin.vmoption	Specifies the Java VM startup options Initial value: -Dsun.java2d.noddraw=true
JBK Plugin policy file	jbk.plugin.policy.url	Specifies the policy file used for JBK Plugin. Initial value: the default policy file of JBK Plugin (file:${jbk.home}/classes/jbkplugin.policy) Specify in the URL representation.Replace spaces with '%20'
Proxy Setting	jbk.plugin.proxy.enable	Specifies whether an HTTP proxy server is used when an applet is downloaded. - true : Uses an HTTP proxy server.

Property	Key	Value and its description
		- false : Does not use an HTTP proxy server. Initial value: None (The proxy setting of the browser is directly used.)
	jbk.plugin.proxy.http.host	Specifies the host name of the HTTP proxy server to be used in downloading an applet. Initial value: None (The proxy setting of the browser is directly used.)
	jbk.plugin.proxy.http.port	Specifies the port number of the HTTP proxy server to be used in downloading an applet. Initial value: None (The proxy setting of the browser is directly used.)
	jbk.plugin.proxy.override	Specifies a list of addresses which are connected without the HTTP proxy server. Initial value: None (The proxy setting of the browser is directly used.)
	jbk.plugin.proxy.secure.host	Specify the host name of the proxy used for HTTP communications. Initial value: None (The proxy setting of the browser is directly used.)
	jbk.plugin.proxy.secure.port	Specify the port number of the proxy used for HTTP communications. Initial value: None (The proxy setting of the browser is directly used.)
The Java console	jbk.plugin.console.visible	Specifies whether the Java console is displayed or not when JBK Plugin starts to run. - true : Displays the Java console. - false : Hides the Java console. Initial value: false
The download method of applets	jbk.plugin.protocol.http	Specifies the download method of applets. - java : Uses the Java networking class. - native : Uses the networking functionality of the browser. - hybrid : Performs as 'native' for class files and JAR files. Performs as 'java' for other files. Initial value: hybrid
The HTTPS download method of applets	jbk.plugin.protocol.https	Specifies the HTTPS download method of applets. - native : Uses the networking functionality of the browser. - hybrid : Performs as 'native' for class files and JAR files. Other files cannot be downloaded by HTTPS protocol. - java : No file can be downloaded by HTTPS protocol. This keyword is reserved for the future use. Initial value: hybrid
Shows the download status messages	jbk.plugin.applet.showmessage	Specifies whether the download status messages are shown or not. - true : Shows the download status messages. - false : Hides the download status messages.

Property	Key	Value and its description
		Initial value: true
Applet Caching	jbk.plugin.www.plugin_cache.enable	Specifies whether the applet caching is enabled or disabled. - true : Enables the applet caching. - false : Disables the applet caching. Initial value: false
	jbk.plugin.www.plugin_cache.dir	Specifies the applet cache folder. Initial value: ${user.home}\applet_cache
	jbk.plugin.www.plugin_cache.dir.freespace	Specifies the free space of the applet cache folder. Initial value: 1% of the drive capacity of the applet cache folder.
	jbk.plugin.www.plugin_cache.update	Specifies the update policy of the applet cache. - newer : Updates the cache if the modification time of the file on the Web server is newer than that of the file in cache. - diff : Updates the cache if the modification time of the file on the Web server is different from that of the file in cache. Initial value: newer
	jbk.plugin.www.plugin_cache.accelerate	Specifies use accelerator mode or not if cache mode is newer. - true : Use accelerator mode. - false : Not use accelerator mode. Initial value: false
	jbk.plugin.www.plugin_cache.twopass	Specifies the method of the inquiry processing to a Web server, and download processing of a file. - true : The inquiry to a Web server and download are performed independently. - false : Use of cash is judged based on the result downloaded from the Web server. Initial value: false
JavaScript Communication	jbk.plugin.sw.script.enable	Specifies call applet methods from JavaScript or not. - true : Use applet method calling. - false : Does not use applet method calling. Initial value: false
Download status notification	jbk.plugin.interface.download_notify	Specify a class with the com.fujitsu.jbk.plugin.browser.PluginDownloadNotify interface implemented to use download status notification. Initial value: None
Troubleshooting (Add these properties as needed)	jbk.plugin.debug.showvmmsg	Specifies output messages of Java VM. If [Ctrl]+[Break] is held down when the applet is running, a full-thread dump is output to the VM log file. Also, the user must specify these options to output a GC log with the "-verbose:gc" argument in the VM startup option. - true : Output messages of Java VM. - false : Does not output messages of Java VM. Initial value: false

Property	Key	Value and its description
	jbk.plugin.debug.tracedir	Specifies output folder of messages of Java VM. Output file is named as "jbktrace.?" (?: number 0 - 9). If the name of the output folder includes a blank character, enclose the output folder name by double quotation (") marks.
	jbk.plugin.debug.console_log	Specifies output of Java console is saved to a file or not. - true : Saves to a file. - false : Does not save to a file. Initial value: false
	jbk.plugin.debug.console_log.dir	Specifies folder name of saving output of Java console. Initial value: ${jbk.home}/console_log
	jbk.plugin.debug.console_log.history	Specifies the number of Java console output files to be saved. Initial value: None (no limit)
	jbk.plugin.debug.console_log.size	Specifies the maximum size of a single Java console output file. Initial value: None (no limit)
	jbk.plugin.debug.console_log.header	Specifies whether or not to add a line header indicating the time to Java console output files. - true: Adds a line header. - false: Does not add a line header. Initial value: false
Supplement (Add this property if necessary)	jbk.plugin.sw.applet.delay	Specifies whether to wait for termination of applets on the closing of a browser window: - true: Waits for termination of the applets. - false: Closes the browser window immediately. Initial value : true
	jbk.plugin.sw.classpath.use_env	Specifies whether the Java VM is to use the CLASSPATH environment variable: - true: The Java VM uses the CLASSPATH environment variable. - false: The Java VM does not use the CLASSPATH environment variable. Initial value : true
	jbk.plugin.sw.trustproxy	Specifies whether names returned by the proxy server are to be used for URL name resolution. "-DtrustProxy=<true|false>" is added to the startup option: - true: The trustProxy property is set to "true." - false: The trustProxy property is set to "false." Initial value : false
	jbk.plugin.sw.fbc.force_activate	Specifies whether or not to activate an applet when the browser becomes active. - true: Activates the applet. - false: Does not activate the applet. Initial value: false

Property	Key	Value and its description
	jbk.plugin.ie.connection.max	Set the maximum number of simultaneous connections when downloading with the browser. A value that is smaller than the default value of the browser is specified, it is not valid.Refer to the following content for the setting. http://msdn.microsoft.com/en-us/library/cc304129(v=vs.85).aspx Initial value :32
	jbk.plugin.sw.event.initial_active	When the applet in an inactive window has started to run, specify whether PluginAppletEvent of PLUGINAPPLET_ACTIVATE is to be notified. - true: PluginAppletEvent is to be notified. - false: PluginAppletEvent is not to be notified. Initial value :false

2.10 Determining whether JBK Plugin is installed

Provided below is an explanation on how to use Web pages to determine whether JBK Plugin is installed on a client machine.

Using VBScript or JavaScript, you can determine the version number of the JBK Plugin instance on the client machine.

As a result, if the latest version of JBK Plugin is not installed on the client machine, the user of the client machine can be redirected to a Web page prompting download of the installer, when the user opens the relevant Web page in a Web browser.

The following sample script determines whether JBK Plugin is installed on the client machine:

Table 2.5 Sample script for determining whether JBK Plugin is installed

```
<HTML>
<HEAD>
<TITLE>JBK Plugin Installation Checking Page</TITLE>

<!-- Script for Internet Explorer -->
<SCRIPT LANGUAGE="VBScript">
<!--
    ' Latest JBK Plugin version number
    Const PLG_REQUIRED_VERSION = 9.01
    ' URL of the installer Web page
    Const PLG_INSTALL_URL = "http://foo.fujitsu.com/index.html"

    Dim CurrentVersion
    CurrentVersion = 0.0

    ' Get the version number of the JBK Plugin instance that is installed.
    ' If JBK Plugin is not installed, GetPluginVersion() returns an error,
    ' but because "On Error Resume Next" is specified, processing
    ' continues, with CurrentVersion remaining 0.0.
    On Error Resume Next
    CurrentVersion = GetPluginVersion()

    ' If the latest JBK Plugin version is not installed:
    If CurrentVersion < PLG_REQUIRED_VERSION Then
        ' The user is asked about being redirected to the Web page that explains
        ' the installation procedure.
        Dim Msg1, Msg2
        Msg1 = "You need to install the latest JBK Plugin version."
        Msg2 = "Do you want to go to the Web page explaining the installation
                procedure?"

        Dim ok
        ok = MsgBox (Msg1 + Chr(13) + Msg2, vbOKCancel)

        ' Go to the Web page explaining the installation procedure.
        If ok = vbOK Then
            document.location = PLG_INSTALL_URL
        End If
    End If

    '
    ' This function acquires the version number of the installed JBK Plugin version.
    '
    Function GetPluginVersion
        Const PLG_NAME = "F5CXWPIE.JBKPluginCtrl.1"
        Dim Plugin
        Set Plugin = CreateObject(PLG_NAME)
        GetPluginVersion = Plugin.version
    End Function
-->
</SCRIPT>
</HEAD>

<BODY>
JBK Plugin installation checking
</BODY>
</HTML>
```

- If JBK Plugin is not installed on the client machine or the version of the installed JBK Plugin is older than 9.01, a message box is displayed to prompt the user to go to the installer Web page.
- It is also possible to use JavaScript although VBScript is used in the example.

  For details on VBScript or JavaScript, refer to the pertinent documentation.
- The following method is used in the script code to determine the JBK Plugin version number:
  1. In Explorer, right-click on (JBK Plugin installation folder)\bin\f5cxwpie.ocx.
  2. A menu is displayed. Select [Properties]
  3. The f5cxwpie.ocx properties are displayed. Select the [Version] tab.
  4. Select [Product Version] in the [Item name] area.
  5. The VL notation is displayed in the [Value] area.

     The version number of the JBK Plugin can be calculated from this VL notation by the following calculating formula:

     (Version number) = (V) + ((L) / 1000)

     *indicates : (V) - Version, (L) - Level.

     - Please replace it with V9.2L00 for V9.2.0. The version number becomes 9.2+(0/1000)=9.2 .
     - Please replace it with V11.0L00 for V11.0. The version number becomes 11.0+(0/1000)=11.0 .
- SCRIPT elements in HTML facilitates use of the same script in multiple HTML files.

  ```
  <SCRIPT LANGUAGE="VBSCript" SRC="xxx.js">
  ```

  For details on SCRIPT elements, refer to the relevant HTML documentation.

Note

If a tag indicating use of JBK Plugin (<OBJECT> or <EMBED>) is written between <BODY> and </BODY> and JBK Plugin is installed on the client machine, JBK Plugin becomes active regardless of the result of the version check by the script. Write this script in a Web page that does not use any applet.

Chapter 3 GUI Library

The GUI Library provides classes for Java application graphical user interface (GUI) such as buttons, labels, frames and dialog boxes.

Also, it includes the UI Screen Control Library, which transfers screens while creating and deleting screens dynamically.

The UI Screen Control Library provides both heavy weight and light weight components.It is recommended to use light weight components on the environment with insufficient memory resources.

For the detail, see 'GUI Library User's Guide'.

3.1 Continuous printing from a Web browser

Components might not be printed when printing successively from a Web browser.

3.2 Supplementary characters

The GUI Library that is older than JBK V5.1L10 does not support supplementary characters, which are characters with code points in the range U+10000 to U+10FFFF.

When supplementary characters are set as parameters of the constructor or method of GUI components that has java.lang.String or char[] as parameters, the GUI components exclude the supplementary characters from the string and proceed with the remaining string.

When a character string that includes supplementary characters is input to the following GUI components on the screen, the supplementary characters are excluded from the character string.

- JFField
    - JFFieldString
    - JFFieldLong
    - JFFieldDate
    - JFFieldDouble
    - JFFieldFilled
    - JFFieldRichString
- JFTextArea

3.3 Problems Caused by JavaSE

The following table lists problems caused by JavaSE.

No.	Notes	Remarks
1	When you press the JFImageButton, move the mouse cursor out of it at high speed with keeping it pressed, release it and move the cursor into it again, the image button might look like as it were pressed.	Do not move the mouse with pushing button at high speed.
2	The shape of the cursor might become "|", even though the cursor is not on the input component.	This is only an abnormality in display.
3	A background might be drawn on the bottom of the JFListView when scrolling.	This is only an abnormality in display.
4	A component which displays a caret such as JFFieldString might cause the language bar to blink in synchronization with the caret.	This is only an abnormality in display. Specify "-Dsun.java2d.noddraw=true" in the Java VM startup option.
5	A component which draws strings such as JFGroupbox might not draw strings.	This is only an abnormality in display. Specify "-Dsun.java2d.noddraw=true" in the Java VM startup option.

No.	Notes	Remarks
6	Drawing of a focus frame may fail, and the resulting display may be hollow.	This is only an abnormality in display. You can prevent it by arranging components so that JFPanel is at the bottom.
7	In a thread other than the AWT EventQueue event dispatch thread, a deadlock may occur when GUI parts that meet the following conditions are deleted from the container: - GUI parts that have displayed a tooltip even once - JFChoice that have displayed a pop-up menu even once	Do not delete GUI parts from containers in threads other than the AWT EventQueue event dispatch thread.
8	When double-click java.awt.Button, sometimes no text can be entered even though the focus is set on the Input field component. (JDK6)	Use the Lightweight Button components of the JBK GUI Library, such as JFImageButton.

Chapter 4 JBK Download Installer

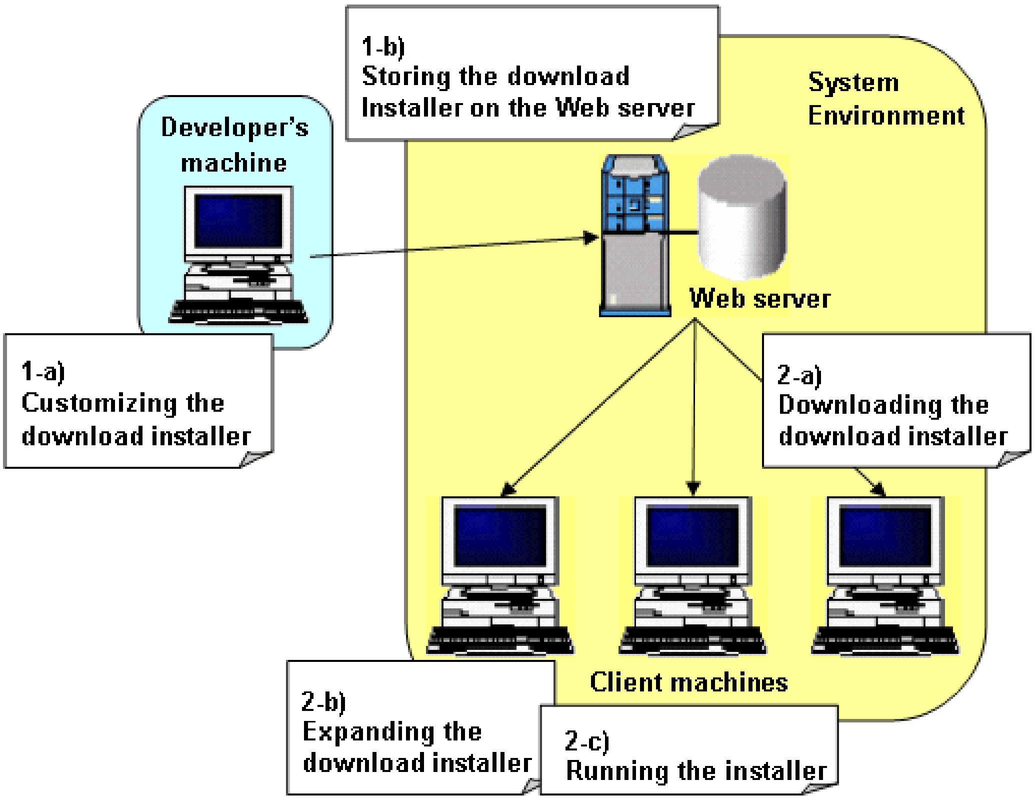
This appendix describes JBK download installer.

4.1 Overview of JBK Download Installer

The JBK download installer creates an installer that can install JBK Plugin, GUI Library, and JRE on a client machine.

By saving the installer created by the JBK download installer to a Web server, you can distribute JBK Plugin, the GUI library, and the JRE on a Windows client.

Figure 4.1 Running the download installer


1-b)
Storing the download Installer on the Web server
System Environment
Developer's machine
Web server
1-a)
Customizing the download installer
2-a)
Downloading the download installer
Client machines
2-b)
Expanding the download installer
2-c)
Running the installer

1. Customizing the download installer(1-a)
2. Storing download installer on the Web server (1-b)
3. Downloading the download installer to a client machine (2-a)
4. Expanding the download installer (2-b)
5. Running the installer(2-c)

JBK download installer has the following functionalities:

- The user can select the version of JREs to be installed.

  You can install the two versions of JRE 6 and 7. It is recommended, however, to install only necessary ones to reduce the amount of downloading size.

- Install the JBK plugin and GUI library.

  Install the same library version of the GUI library as that of the selected JRE.

- The user can customize JBK Plugin property files in advance.

  Before storing the installer on a Web server, the user can customize the JBK Plugin setting files (jbkplugin.properties and jbkplugin.policy) in advance. These customized setting files can be distributed to all client machines.

- The user can use silent installation.

  The silent installation performs automatic installation. No dialogs are displayed and the user does not have to respond during installation. This function can be used if the installation folders and installation options are the same on all client machines.

Note

The number of client machines

For one product that includes JBK, the user has the right to install JBK Plugin and JREs in a single machine. The user must purchase as many products as the number of client machines that s/he wants to install them in with this download installer.

4.2 How to Customize the Download Installer

This section explains how to use the download installer.

The system developer must customize the download installer as shown below before storing it on the Web server:

- Select the required version of JRE, and include it in the download installer.
- Customize the JBK Plugin setting file.
- To use the silent installation function, create a response file for silent installation.

4.2.1 Required Preparations

Before customizing the download installer, make the following preparations:

- Download installer composition kit

  Before installing JBK, select the JBK Plugin download installer composition kit. The composition kit is installed in the \IDE\jbkplgdi subfolder of Product-installation-folder.

  The download installer composition kit includes the following folder and files.

Folder name	Explanation
jbkplgdi	Tool for copying modules belong to the download installer is included.
jbkplgdi\mainins	This is the main component file of the download installer.
jbkplgdi\jbkplug	This is the file for installing the JBK Plugin.
jbkplgdi\jre6	This is the file for installing JRE 6.
jbkplgdi\jre7	This is the file for installing JRE 7.

- Archive tool

  Before storing the customized download installer on a Web server, compress and save it as a single file. For the purpose, an archive tool is required.

  Use any archive tool that preserves folder hierarchy. Among archive tools available from Fujitsu or other companies, select and use the one that best satisfies the requirements of the user's system.

4.2.2 How to Use

Follow the procedure below to customize the download installer:

1. Selecting modules to be included in the download installer

To extract files of modules to be installed, execute <Product-installation-folder>\IDE\jbkplgdi\Setup.exe with Administrator authority. After that execute Setup.exe, then [JBK Download Tool] dialog appears.

1. [Chose Destination Location]

   Select a folder in where store the modules included download installer and do customized.

   Note

   Destination Location

   The default initial folder of this dialog is the value of the TEMP environment variable. The default initial folder might not be displayed because of the settings of Windows Explorer. When the default initial folder is not displayed, change the installation folder to the folder that is displayed in Windows Explorer.

2. [Select Components]

   Select the version of the JRE to attach to the installer.

   - JRE 6
   - JRE 7

3. [Start Copying files]

   Confirm the modules before installing.
   Files are copied when <next> button of this dialog is pressed.

4. [Setup Complete]

   Copying modules to the folder in where do customize is completed when this dialog is shown.

2. Files extracted in working folder

Main part of download installer are extracted under the working folder and the installer which is installing each modules are extracted in sub-folder by executing copying tool.

The sub-folder name of each modules are follows:

Folder name	Explanation
(working folder)	main
(working folder)\jbkplug	J JBK Plugin
(working folder)\jre6	JRE 6
(working folder)\jre7	JRE 7

Note

Subfolder name

The main component of the download installer identifies each attached module by its subfolder name. When installing individual modules, do not change the name of the subfolder that contains the file.

3. Customizing JBK Plugin setting files

The JBK Plugin setting files (jbkplugin.properties and jbkplugin.policy) are stored in the "jbkplug\setupdir\0011" folder under the work folder. Edit and customize these files as necessary. Please store the customized configuration file in the "jbkplug" folder after it edits it.

- The followings are the examples that the user needs to customize the JBK Plugin setting files.
    - If security permission setting is required to use Java2 JRE

      Specify the required security permission in the jbkplugin.policy file.

    - If class path setting is required

      To use the class files and JAR files installed on the client machine while the applet is running, specify a class path for them in the VM startup option of jbkplugin.properties.

    - If two JREs are used

      To select one of the JREs installed on the client machine for the applet, create the property files for switching (see the above description), and specify the name of the JRE to be used in each property file.

      To switch the JBK Plugin property files, create files called jbkplugin.properties.xxx (xxx is an arbitrary character string) in the jbkplug folder under the work folder as necessary. For the detail of switching the property files, see the "2.3.12 Switching the Property Files" section in the JBK Plugin manual.

Note

The download installer installs the property files stored in the jbkplug folder. Be sure to store the JBK Plugin setting files in the jbkplug folder.

4. Creating a response file for using the silent installation function (if needed)

For the detail of the silent installation, see "4.4 Silent Installation".

5. Compressing and saving work folder files in a single file

Use an archive tool to compress all files in the work folder and save them as a single file. Make sure to compress the files in such a way that the subfolder hierarchy does not change when the compressed file is expanded.

After finishing the tasks above, the user has only to store the compressed file on a Web server.

Files extracted in working folder should be deleted when those are not needed.

4.3 Running the Download Installer

Install with the privilege of the computer administrator or a member of the Administrators group.

1. The system user must first download the download installer from a Web server to a client machine.
2. Since the downloaded installer is a compressed file, expand it on the client machine.

   Expand the file as a temporary file on the client machine, or create a designated work folder for the expanded file.

3. When the file is expanded, the file setup.exe is created in the folder.

   To start installation, execute setup.exe.

   Note

   Setup.exe

   To start installation, execute setup.exe in the folder where the file has been expanded. Setup.exe has also been created in the subfolders (jbkplug and jre6) in which the installation file for each module is stored; however, if the setup.exe-files in the subfolders are executed, no files are installed.

4. When the download installer runs, the following dialog boxes open.
    a. [Welcome]

       Press the <Next> button in this dialog box to start the installation.

b. [License Agreement]

Press the <Next> button.

c. [Choose Destination Location]

Choose the folder in which files are to be installed. In the installation folder, the following subfolder is created where the JREs are installed.

Name of subfolder in which JREs are installed.	JRE version
jre6	JRE 6
jre7	JRE 7

Note

Installation folder

The maximum length of the installed folder is 200 characters. Please don't use the following characters in the installed folder name. ":", ";", "/", "*", "?", "\", "<", ">", "(", ")", "|", "#", "%", "^", "!".

d. [Select Components]

Select the JRE to be installed.

e. [Select Default JRE]

This dialog box opens if JREs of two or more versions are installed. It displays the following message: "Choose JRE to use by the default." The JRE selected in this dialog box becomes the default JRE. If no JRE is specified in the JBK Plugin property file, JBK Plugin runs the applet using the default JRE.

If only one type of JRE is installed, this JRE becomes the default JRE.

f. [Start Copying Files]

Before starting the file copy operation, check the modules to be installed.

Press the <Next> button in this dialog box to start the file copy operation.

g. [Complete Setup]

- Select Yes to immediately restart the machine.
- If No is selected, the machine is not restarted. If No is selected, restart the machine before using JBK Plugin.

This completes the installation of JBK Plugin, GUI library, and JREs. After the installation ends, the user can delete the compressed file of the downloaded installer and the related expanded files.

Note

If uninstallation must be performed immediately after installation because an incorrect setting was selected during installation, restart the machine before uninstallation. Otherwise, uninstallation may fail.

4.4 Silent Installation

This section explains the silent installation of the download installer.

The silent installation performs an installation automatically, without displaying dialog boxes at runtime for the user response. This can be used in the case that the installation folders and installation options are the same on all client machines. The silent installation enables to reduce the amount of the user's work during installation as well as to prevent installation failure due to incorrect operations.

Tasks to be performed by the system developer

1. Customize the JBK Plugin setting files in the procedure for "4.2.2 How to Use".

2. Create a response file for silent installation.

   To create the response file, execute the file setup.exe of the main installer in the command prompt, by specifying the -r option. An example is shown below:

   ```
   C:\TEMP>Setup.exe -r
   ```

   When the -r option is specified, the download installer is executed in response file creation mode. When the installer is executed in this mode, the same dialog boxes as those explained in "4.3 Running the Download Installer" open, and selection results for each dialog box are recorded in the response file (setup.iss). The file is not actually installed in this mode.

   According to the instruction in the dialog boxes, enter information such as the names of installation folders and installation options for the client machine.

   Note

   - The silent installation of this download installer, when doing silent installation the modules is not installed correctly at request file creation mode disabled some JRE shown in [Select Components].

     - Include only required subinstallers in the main installer.

     - In response file creation mode, select all of the JRE displayed in the [Select Components] dialog box.

   - If the user select Yes in the last dialog box, [Complete Setup], the silent installation automatically reboots the client machine after the installation is completed. To prevent the machine from rebooting automatically, select No in the [Complete Setup] dialog box.

3. Copy the created response file to the work folder.

   The response file (setup.iss) is created in the Windows folder (e.g., C:\Windows). Copy this file to the work folder for customization of the download installer.

   Note

   Copy the response file to the same folder as that of setup.exe of the main installer. If this file is copied to another folder, the silent installation does not work.

4. Compress and save the work folder files, including the response file, to a single file. After that, the system developer has only to store the compressed file to a Web server.

Note

When the download installer runs in response file creation mode, no file installation is performed, but installation information is recorded in the Windows registry. For deleting the information in the registry, select Add/Remove Programs in the control panel window and uninstall the information by selecting "JBK Download Installer".

Tasks to be performed by the system user

1. To perform a silent installation, first download the download installer containing a response file from the appropriate Web server, and then expand it on a client machine.

2. Next, in the command prompt, execute the file setup.exe in the folder where the file has been expanded, by specifying the -s option. An example is below:

   ```
   C:\TEMP>Setup.exe -s
   ```

   If the installer containing the response file is executed with the option -s, it runs in silent mode. In this mode, the installer performs the installation using the information recorded in the response file instead of opening dialog boxes and waiting for responses from the user.

Point

- Some archive tools automatically execute specified executable files after expansion of a compressed file. Use this type of tool to automatically start a silent installation by executing "setup.exe -s".
- When silent installation ends, execution results are recorded in a log file (setup.log). The log file is created in the same folder as that of setup.exe. To determine whether the installation succeeded, check the following line in the log file:

  ```
  [ResponseResult]
  ResultCode=0
  ```

  If ResultCode is 0, the installation succeeded. If an error occurred, a value other than 0 is recorded. If this occurs, the response file may be invalid. In this case, execute setup.exe without the -s option to install according to the usual procedure.

4.5 Notes of JBK Download Installer

This section contains notes on download installation.

- When making or running of the download installer on Windows 7, the program compatibility assistant dialog box is displayed,click [Cancel] to close the dialog box.
- If the download installer is used to install files on a machine where JBK has already been installed, the files cannot be installed in the same folder as the JBK installation folder. Install the program in another folder.
- When using the JBK plugin in an environment where the following products or features are installed, and a JBK plugin is installed with the download installer, that plugin is enabled and the previously installed JBK plugin will not work:
  - Interstage Application Server Client Package
  - Interstage Studio Standard-J Edition
  - Interstage Studio Client Runtime

  Furthermore, the above JBK plugin will not work if the download installer is uninstalled.
- The previous version of JBK Downloader Installer can do the overwrite installation only in the system that has been installed :
  - JBK Download Installer of Interstage Apworks 8.0.0, Interstage Studio V9.0.0 or later
  - Download Installer of Interstage Application Server 8.0.0 or later
- To carry out overwrite installation with the previous version of JBK Downloader Installer, choose all all components that have already been installed and install it.
- It is not possible to specify an installation folder in the overwrite installation. The files are copied in the already installed folder.
- When JBK Plugin and JRE of a new download installer is installed in the environment that installs JBK Plugin and JRE of the previous download installer in the overwrite Installation, the message which confirms overwrite is displayed.
- The download installer cannot remove only a specific component. Please uninstall a download installer, in removing.
- When you uninstall "JBK Download Installer", select Add/Remove Programs in the control panel window and uninstall the information by selecting "JBK Download Installer".
- Reinstallation or uninstallation of JBK Download Installer may fail because of unexpected errors that occur during installation or uninstallation.

  In such cases, execute work referring to the following files.

  ```
  (Product-installation-folder)\IDE\jbkplgdi\mainins\HowToDel.txt
  ```

Chapter 5 Notes

This section provides common notes on use of JBK.

5.1 Notes on Security

Access to local resources

The following classes access to the clipboard or load native libraries:

- JFFieldDate, JFFieldDouble, JFFieldFilled, JFFieldLong, JFFieldString, JFTableView, JFToolbar, JFTree, JFTextArea, and JFChoice in GUI Library

If the applet which uses any of the above classes does not exist on the local machine and is downloaded from a Web server, the access to the local resources is limited. To use such an applet, take one of the followings:

- Use signed applets.
- Add the minimum security authority required to the java.policy file.

In general, native libraries involve security problems because the libraries can freely access the resources on the client machine.

Glossary

Applet

Executable program downloaded from Web Server.

Applet with a signature

Applet with a signature is an applet accompanied by the signature to prove that this applet is safe in security.

Automatic exit function

The automatic exit function is a function to move focus to the next component automatically when the number of characters exceeds the maximum in the input field.

Calendar

The Calendar (JFCalendarView class) is parts which displays the calendar and selects the date on it.

Caret

The Caret is a sign by which the position which the character inputs is shown.

Cell

The Cell (JFCell class) is parts which draw to a rectangular area in the character string and the image.

Checklist

The Checklist (JFCheckList class) is parts which display the list with the check box.

Choice

The Choice (JFChoice class) is parts for the item to be selected, and to display the item which has been selected.

Clipboard

The Clipboard is an area where data is temporarily memorized when data is forwarded between the applications.

Data are transferred from a certain application to the Clipboard, and a data transfer between the applications can be done by putting those data from another application.

The content memorized in the Clipboard is preserved from the end of the system or until data is forwarded from the application to the Clipboard.

Column header

The Column header is an uppermost division header line of each row of the component to which the data of a secondary origin is displayed by the table form.

Condition color specification function

The Condition color specification function is a function to change the color displayed according to the content of the character string.

Domain

The Domain is a name to identify the computer connected with the network.

Double buffering

The Double buffering is processing by which the content of the buffer is copied on the screen at a time after all drawing processes are done by the buffer of an off screen.

Dummy column header

The dummy column header is a column header to display it in the area where the entire length of the column header remains in the small case than the print area.

Embedded Character String Field

The Embedded Character String Field (JFFieldFilled class) is parts which consist of some labels (fixed part) and input fields (variable).

Event

The Event which occurs corresponding to the event when user of transaction processing system clicks pushbutton or selects menu.

Event Listener

The Event Listener is the object on the side where the event is received. Event Listener is only said the listener.

Function key manager

The Function key manager is JFTextEditKey class. The Function key manager is class which registers processing with function keys such as [Ctrl]+[C] and [Delete].

The Function key manager can register the processing which has been defined, the insertion processing of a specified character string, and the user definition processing in each function key.

Group Box

The Group Box (JFGroupbox class) is parts with the label used so that plural controls may be grouped and the frame line.

GUI Library

The GUI Library is a class library to achieve graphical user interface (GUI) of the Java application.

The expression power/productivity can be improved compared with the application form screen made by using the function of the graphical user interface which JDK offers by using the GUI Library.

HTML (HyperText Markup Language)

HTML is a language used to describe the page (document) of WWW (World Wide Web).

HTML might be called the Hypertext descriptive language. HTML is a descriptive language widely used to describe the document in WWW (World Wide Web).

It depends on setting the program made by the Java which is called an applet in the HTML document for the program developed by Java on World Wide Web can be used.

HTTPS protocol

The HTTPS protocol is a protocol specified when the communication of HTTP which uses SSL is done. The HTTPS protocol is specified for the head of URL "https://".

Image Button

The Image Button (JFImageButton class) is parts which can display the image.

Moreover, The Image Button is possible to use it as a switch button which can express the state of ON and OFF.

Indent

The indent is indent of the character.

Input-output field

Input-output fields (JFFieldDate, JFFieldDouble, JFFieldLong, and JFFieldString class) are parts which input and output the date value, the real number value, the integer, and the character string with the single line respectively.

Inset

The Inset is space of the upper and lower right and left which the container leaves for own edge.

The area becomes a boundary and blank space or title.

Interstage Studio Standard-J Edition

Interstage Studio Standard-J Edition is a product that offers the integrated development environment. This product enables the user to develop business application efficiently. In Interstage Studio Standard-J Edition, the development environment of the business application and the execution environment are offered. In Interstage Studio Standard-J Edition, various applications are developed and can be operated.

J Business Kit offers the function to develop the Java application as one component which composes Interstage Studio Standard-J Edition.

IP address (Internet Protocol address)

IP address is an address for the network uniquely allotted to the computer.

Java

Java is a programming language which United States Sun Microsystems, Inc.(Present Oracle) advocated.

The compiler of the Java language is converted into the intermediate code which is called byte code of the format by which the source code does not depend on the platform. The program which executes this byte code is called Java VM (Java virtual machine). Java VM is offered to each platform such as UNIX, Windows, and Macintosh.

Java application

As for the program developed by the Java installed in the personal computer and the server first, the Java application is a program of the format executed in the Java execution environment built in those models.

JavaBeans

The JavaBeans is a mechanism to make the software parts (component) which can be reused.

The program can be easily developed by reusing parts specifying the relation between each part by an easy operation which uses GUI.

Java console

The Java console is a console to display the character string output to the standard output and the standard error output when the JBK Plugin is executed.

Java Development Kit (JDK)

Java Development Kit is a Java development kit which the Oracle Corporation offers.

In Java Development Kit, a basic general-purpose class library requisite to develop the program by Java is offered.

Java Plug-in

Java Plug-in is an application program into which browser's function is expanded to operate the Java program by the Internet browser.

The Oracle Corporation offers Java Plug-in.

Java virtual machine

When the Java program is compiled, the intermediate code which is called byte code is made. The Java virtual machine installed in each platform interprets and executes this intermediate code, and the program operates.

JBK

JBK is abbreviation of J Business Kit.

JBK Plugin

JBK Plugin is a plug-in function which JBK offers to support the applet execution on the Web browser.

The applet can be executed by using JDK/JRE of the Oracle Corporation instead of Java VM which the browser offers when JBK Plugin is used.

jbkplugin.properties file

The jbkplugin.properties file is a file to describe set information on JBK Plugin.

jbkplugin.shutdown file

The jbkplugin.shutdown file is a file used when the function to inquire the end of the applet which the JBK Plugin offers is temporarily invalidated.

JIT

JIT is abbreviation of the Just In Time compiler.

JIT is software which dynamically compiles the program of the Java of byte code to the native code and executes it.

JPEG

JPEG is abbreviation of Joint Photographic Expert Group.

In ISO, CCITT, and EIC, JPEG standardizes it to encode the color static image jointly. The compression expansion technology of the color static image adopted here is used.

List Form

The List Form (JFListView class) is parts which display the data of a secondary origin by the List Form (Table Format).

Locale

The locale is a specific geographic, political or cultural region.

The locale is mounted in the Locale class of JDK, and used for processing according to the locale.

Method

The Method is a function defined in the class, and the meeting of the execution sentence which operates the field etc. of Java is called a method.

Multiple-Line Label

The Multiple-Line Label (JFMultiLineLabel class) is parts which can display the multiple lines.

Pane

Panes are each area divided when one rectangular area is delimited and it divides into some.

Point

The point is one of units of length in the print relation, and one point is about 0.35mm (about 1/72inch).

At present, the point is used as the name of the size of the type.

For example, it is called with "8 point" when the size of one side of the type is 8 point.

Policy file

The policy file is a file to which security information like the access authority etc. of the file is set.

The policy file is a file which Java VM offers.

Progress Bar

The Progress Bar (JFProgressbar class) is parts used to show the progress situation of the operation.

Proxy server

The Proxy server is a server for the relay used when connecting it with the server outside the firewall on the network.

Null value (null)

Null value is a value to show the state that the value is not set in the item.

Offset

The offset is a relative value from a certain standard.

Panel

The Panel (JFPanel class) is parts with the double buffering function etc.

Password input function

The password input function is a function to replace the input character with a specified character ('*' etc.), and to display it on the screen.

Pixel

The pixel is a minimum unit to display graphic on the screen.

RGB

RGB value shows the color displayed on the screen by elements of the red-green-blue three primary colors.

Security level

The Security level is the level of the security limitation which hangs when applet without a signature is carried out with JBK Plugin.

Slider

The Slider (JFSlider class) is parts which are composed of the switch and the scale, and move the switch by the mouse and the key operation and change the value.

Socket

The Socket is a mechanism to receive and to pass data between the applications which operate on different Java virtual machine or a different host.

Spin Button

The Spin Button (JFSpinButton class) is parts by which two buttons in the vertical direction or horizontal direction become one.

SSL (Secure Sockets Layer)

SSL is one of the encoding technologies of the data when communicating the network.

Status Bar

The Status Bar (JFStatusbar class) is parts which display the message line and the status indicator.

Stream

The stream is data and a flow of the content of the communication.

Table Format

The Table Format (JFTableView class) is parts to display/to edit the data of a secondary origin by the table format (table format).

Tab Panel

The Tab Panel (JFTabPanel class) is parts with two or more pages to switch the display of two or more components.

Text Area

The Text Area (JFTextArea class) is parts which edit and display the text of multiple lines.

Time zone

It is said "The time zone is different" that the standard time is different according to the region.

The Time zone is mounted in TimeZone class of JDK and is used for processing according to the time zone.

Toolbar

The Toolbar (JFToolbar class) is parts which arrange the component, and display the tool chip.

ToolTip

The ToolTip (JFToolTip class) is parts to display the character string in the pop up window.

Trace

When the event occurs, the generation event is recorded one by one and historical information of the program execution process is collected.

Tree Format

The Tree Format (JFTree class) is parts which display data by the tree format with the multiple column information (multicolumn).

URL (Uninfom Resource Locater)

URL is an address where the home page of Internet and the destination of the storage of the resource file, etc. are shown.

Web Browser

Web Browser is software to inspect information of Web Server on the client side.

Internet Explorer, etc. are typical software.

Web Server

Web Server is a server to manage the data of the home page and the applet, etc.

Workbench

The Workbench is integrated development environment to develop application by using GUI of Windows.