"The Vienna Version of BSF4Rexx"

2006 International Rexx Symposium
Austin, Texas, U.S.A. (April 2006)

Rony G. Flatscher (Rony.Flatscher@wu-wien.ac.at)
Wirtschaftsuniversität Wien, Austria (http://www.wu-wien.ac.at)
Agenda (Vienna Version, 2006)

- Brief History
- Architecture
- Changes
  - Examples
- New features
  - Examples
- Roundup and Outlook
BSF4Rexx History, 1

- Wintersemester 2000/01
  - Seminar assignment at the University of Essen
  - Proof of concept by a student (Peter Kalender)

- Spring 2001
  - Introduction of a re-write and w.r.t. BSF complete version of "BSF4Rexx" to the RexxLA
  - Ongoing work and improvements
• Spring 2003
  – Introduction of the "Augsburg" version of BSF4Rexx to the RexxLA
  – Bug fixing
  – Added a few external Rexx functions to the external function package "BSF4Rexx.dll"
    • E.g. allows to demand load Java on Linux and Windows
Agenda (from 2003)

- Revealing the *real* Title
- Brief Architecture
  - The "Essener" Version of BSF4Rexx (2001)
- An example
  - A Java program
  - A Rexx program
- Additional new features
- Roundup
The Largest External Function Package for Rexx on Earth!

[And already ported to all important operating systems and hardware platforms!]
Bean Scripting Framework

- A Java framework, making it easy for Java to invoke scripts in non-Java scripting languages
  - E.g. JavaScript, NetRexx
- Originally developed by IBM as open source
  - Part of IBM's WebSphere to allow scripts to be deployed within Java Server Pages (JSP)
- Fall 2003 handed over to jakarta.apache.org
  - Used e.g. in ant, xerces
• BSF with a Rexx engine
  – Allows the usage of Rexx from BSF
    • Any Java program can invoke Rexx
    • Rexx scripts are able to communicate with Java objects, if made available by the Java program
  – Allows Java to be used as a huge Rexx function library
    • The public methods and public fields of every Java object and Java class object can be used by Rexx
    • If necessary, Java can be started up by Rexx
BSF4Rexx
Architecture

Java Program

<table>
<thead>
<tr>
<th>BSF (Java)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSF Registry</td>
</tr>
<tr>
<td>RexxEngine</td>
</tr>
</tbody>
</table>

JNI

BSF (Java)

BSF4Rexx

(C++)

<table>
<thead>
<tr>
<th>BSF4Rexx</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSF()</td>
</tr>
<tr>
<td>BsfDropFuncs()</td>
</tr>
<tr>
<td>BsfInvokedBy()</td>
</tr>
<tr>
<td>BsfLoadFuncs()</td>
</tr>
<tr>
<td>BsfLoadJava()</td>
</tr>
<tr>
<td>BsfQueryAllFunctions()</td>
</tr>
<tr>
<td>BsfQueryRegisteredFunctions()</td>
</tr>
<tr>
<td>BsfUnloadJava()</td>
</tr>
<tr>
<td>BsfVersion()</td>
</tr>
</tbody>
</table>

Rexx And Java

Rexx And Java
import com.ibm.bsf.*;  // BSF support
import java.io.*;     // exception handling

public class TestSimpleExec {

    public static void main (String[] args) throws IOException {
        try {
            BSFManager mgr = new BSFManager();
            BSFEngine rexx = mgr.loadScriptingEngine("rexx");
            String rexxCode = "SAY 'Rexx was here!"");

            rexx.exec ("rexx", 0, 0, rexxCode);
        }
    }

    catch (BSFException e) { e.printStackTrace(); }
}

Output:
Rexx was here!
/* "getJavaVersion.rex": classic Rexx version, querying the installed Java version */

/* load the BSF4Rexx functions and start a JVM, if necessary */
if rxFuncQuery("BSF") = 1 then /* BSF() support not loaded yet? */
do
  call rxFuncAdd "BsfLoadFuncs", "BSF4Rexx", "BsfLoadFuncs"
call BsfLoadFuncs /* registers all remaining BSF functions */
call BsfLoadJava /* loads Java */
end

say "java.version:" bsf('invoke', 'System.class', 'getProperty', 'java.version')

Invoking the program either with:
  rexx getJavaVersion.rex
or:
  java org.rexxla.bsf.RexxDispatcher getJavaVersion.rex
or (shorthand of the above):
  {rexxj.cmd|rexxj.sh} getJavaVersion.rex

Possible Output:
java.version: 1.5.0_06
Entering ooRexx

- **BSF.CLS**
  - An ooRexx module containing
    - Supporting BSF via the proxy class **BSF**
    - Supporting BSF routines, e.g. `bsf.import(…)`
    - Services like making the most important and pre-registered Java classes directly available via the environment symbol `.bsf4rexx`
  - Will load Java transparently, if not yet loaded
    - Rexx programs
BSF4Rexx with **BSF.CLS**

Architecture

- BSF() (Java)
- BsfDropFuncs()
- BsfInvokedBy()
- BsfLoadFuncs()
- BsfLoadJava()
- BsfQueryAllFunctions()
- BsfQueryRegisteredFunctions()
- BsfUnloadJava()
- BsfVersion()

Java Program

- BSF Registry
- RexxEngine

JNI

BSF4Rexx

(C++)

ooRexx environment (e.g. ".bsf4rexx")

BSF.cls

ooRexx scripts

RexxAndJava
/* "getJavaVersion.rex": classic Rexx version, querying the installed Java version */
call bsf.cls /* load the Java support */
say "java.version:" bsf('invoke', 'System.class', 'getProperty', 'java.version')

::requires bsf.cls /* load the Java support */

Invoking the program either with:
   rexx getJavaVersion.rex
or:
   java org.rexxla.bsf.RexxDispatcher getJavaVersion.rex
or (shorthand of the above):
   {rexxj.cmd|rexxj.sh} getJavaVersion.rex

Possible Output:
java.version: 1.5.0_06
BSF4Rexx with BSF.CLS
A Rexx Script Interfacing with Java, 2

Invoking the program either with:
  rexx getJavaVersion.rex
or:
  java org.rexxla.bsf.RexxDispatcher getJavaVersion.rex
or (shorthand of the above):
  {rexxj.cmd|rexxj.sh} getJavaVersion.rex

Possible Output:
java.version: 1.5.0_06
Java's Strong Typing

• Every variable needs to be typed
  – Java compiler must have access to type
  – Java compiler checks whether all variables are used according to their type
  – Java compiler checks whether all arguments are of the correct type
• Hence interfacing with Java means to supply the correct types!
"Type indicators" preceed the argument in BSF()-subfunctions.

"Type indicators" are one of the following strings:
- Boolean, Byte, Char, Double, Float, Int, Long, Object, Short, String

Only bold and uppercase letters need to be given.
Java type information is given in the HTML documentation.
"Boolean", "Byte", "Char", "Double", "Float", "Int", "Long", "Short", "String" are the Java "primitive" data types.
"Object" is any Java object.
BSF4Rexx Type Indicators, 2

New Feature

• Starting with the Vienna version of BSF4Rexx no need to indicate Java types anymore
  – Makes it simpler to use Java
    • BSF4Rexx will figure out the correct types and supply Java with them!
  – Still, strongly typed subfunctions are made available and start with the word "Strict"
    • May be needed in very rare circumstances
BSF4Rexx Type Indicators, 3
BSF.CLS – New Feature

- Sometimes one needs to supply primitive datatypes embedded in Java classes
- Public routines `box()`, `unbox()`

```rexx
def obj = box('Long', '123456789012')  /* wrap a long value in a Java object */
say obj  /* name of object in BSF registry */
say obj~toString  /* string representation by Java class */
def val = unbox(obj)  /* Rexx string */
say val
```

```
::requires bsf.cls  /* load the Java support */
```

Possible Output:

```
java.lang.Long@be991a08
123456789012
123456789012
```
BSF4Rexx
BSF.CLS – New Feature

• Camouflaging Java fields as if they were ooRexx attributes
  – Querying the value of a Java field by merely sending the Java field's name
  – Setting the value of a Java field by merely sending the Java field's name followed by the assignment operator and new value
BSF4Rexx: Accessing Static Fields

BSF.CLS – New Feature

- Sometimes one needs to access static values of Java (interface) classes
- Public routine `bsf.wrapStaticFields()`

```rexx
dir=bsf.wrapStaticFields(nameOfJavaInterfaceClass)
```

```
javaClassName="org.oorexx.datergf.DTC" /* interface class defining constants */
dtc=bsf.wrapStaticFields(javaClassName) /* wrap up interface class */
say "version:" dtc~version "january:" dtc~january

::requires bsf.cls /* load the Java support */
```

Possible Output:

```
version: 92.20060101 january: 1
```
BSF4Rexx – Getting at Event Objects, 1

BSF.CLS – New Feature

• Allows retrieving the Java event object giving further information of the event
  – The event object's bean name (index into the BSF registry) will be encoded in the leading comment inserted by BSF4Rexx
• New subfunction, method of BSF.CLS
  `bsf.addEventListenerReturningEventInfos()`
• New routine in BSF.CLS
  `bsf.getEventInfoObject(eventText)`
  • Returns a proxy (array) object that will remove the event Java object from the BSF registry upon deletion
BSF4Rexx – Getting at Event Objects, 2

BSF.CLS – New Feature

- Information in the received array object arr

  [1] ... an array of the arguments that the event generated, usually the respective event object is at the first index, ie. arr[1][1]

  [2] ... .nil or data as supplied by ooRexxx when event adapter was set up

  [3] ... string denominating the event name that has occurred

  [4] ... .nil or string of event names to react upon

  [5] ... a reference to the BSFManager instance
Public class `bsf.dialog`

Multiplatform, uses Java's swing GUI

Dialog (class or instance) methods

- `.nil= bsf.dialog~messageBox(message, [title], [type])`
- `buttonNumber= bsf.dialog~dialogBox(message, [title], [type], [optionType], [icon], [txtButtons], [defaultTxtButton])`
- `text= bsf.dialog~inputBox(message, [title], [type], [icon], [txtOptions], [defaultTxtOption])`

where "type":
- error, information, plain, question, warning

Where "optionType":
- default, OkCancel, YesNo, YesNoCancel

If using the class object `.BSF.DIALOG`, then the dialog is centered relative to physical screen, if created for a Java window object the dialog is modal for it and centered relative to it.
say "Using class object .BSF.DIALOG, hence centered relative to screen..."
.bsf.dialog~messageBox("Think about it!")

say "dialogBox: returns -1 for escape, 0 for first button, 1 for second button..."
pause
buttonText=.array~of("Save it 0", "Delete 1", "Copy 2", "whoops, that's it!!")
say .bsf.dialog~dialogBox("Please choose one", "Choices", "warning", , , buttonText, "Delete 1")

say "inputBox: returns .nil for escape, text value entered or chosen..."
pause
buttonText=.array~of("Save it 0", "Delete 1", "Copy 2", "whoops, that's it!!")
say .bsf.dialog~inputBox("Please choose one", "Choices", "information", , , buttonText, "Delete 1")

say .bsf.dialog~inputBox("Please enter your name:", "Querying stuff", "question")

::requires bsf.cls /* load the Java support */

Possible Output:

Using class object .BSF.DIALOG, hence centered relative to screen...
dialogBox: returns -1 for escape, 0 for first button, 1 for second button...
Drücken Sie eine beliebige Taste . . .
1
inputBox: returns .nil for escape, text value entered or chosen...
Drücken Sie eine beliebige Taste . . .
Delete 1
Rony G. Flatscher
BSF4Rexx: BSF.Dialog
Examples, 2 (Relative to a Frame)

Possible Output:

Using an instance of .BSF.DIALOG, hence centered relative to a frame object...
dialogBox: returns -1 for escape, 0 for first button, 1 for second button...
Drücken Sie eine beliebige Taste . . .
3
inputBox: returns .nil for escape, text value entered or chosen...
Drücken Sie eine beliebige Taste . . .
whoops, that's it!!
BSF4Rexx – Installation Scripts
Running on Linux, Windows

• setupBSFRexx.rax [path2java.exe [dir4scripts]]
  – installBSF4Rexx.{cmd|sh}
  – uninstallBSF4Rexx.{cmd|sh}

• setupOOo.rex path2OOoSOHomeDir
  – installOOo.{cmd|sh}
  – uninstallOOo.{cmd|sh}

• setupJava.rex
  – Linux only
• Date and time arithmetics/manipulations
• Java version of the datergf package, named
  – org.oorexx.datergf
    • DTC ... defines datergf constants
    • DateRGF
      – e.g. subtractions, additions, determining Easter, Labor Day...
    • TimeRGF
    • DateTimeRGF
    • DateFormatRGF
      – Allows formatting of date and time values with easy to apply formatting patterns
org.oorexx.misc
   - Class RgfFilter
      • Implements the Java interface "java.io.FilenameFilter"
      • Needed e.g. for file dialogs that need to filter the files to be displayed

org.rexxla.bsf
   - Class RexxDispatcher
      • Allows starting BSF4Rexx Rexx programs from the command line via Java, supplying the command line arguments to the Rexx program
• org.oorexx.uno
  – RgfReflectUNO
    • A Java class allowing for full reflection/introspection of UNO objects and/or UNO IDL definitions
    • Results are delivered as strings

• Quite a few new nutshell examples
  – Lee's examples of the 2006 Symposium demonstrating platform independent GUI and printing for ooRexxx
  – OpenOffice.org/StarOffice automation examples
Roundup and Outlook

• Vienna Version of BSF4Rexx
  – Introduces typeless interaction with Java
  – Adds utility routines for easing interfacing with Java considerably, e.g.
    • `box()`, `unbox()`, `bsf.wrapStaticFields()`
  – Public routines `iif()`, `pp()`
  – Public class `BSF.Dialog` to allow for using cross-platform `messageBox()`, `dialogBox()`, `inputBox()` functionality
Open Issues

• Real-time handling of events
  – E.g. no canceling possible

• Creating Java proxy objects for Java interfaces
  – E.g. Java Filter interface class
    • At the moment one needs to create a Java class
      which implements the Java interface and control that
      from ooRexx

• Creating ooRexx proxy objects to which Java
  methods can be forward to
  – implementing Java methods in ooRexx
External BSF4Rexx Functions - Overview

The the external BSF4Rexx functions allow interfacing with Java. If the Rexx script is invoked by Java, then the external Rexx function BSF() is registered already.

BSF
Main function to interface with Java.

BSFDropFuncs
Drops all registered BSF4Rexx functions.

BSFInvokerBy
Returns 0, if no Java is present, returns 1 if BSF4Rexx was invoked by Java, returns 2 if BSF4Rexx was invoked by Rexx.

BSFLoadFuncs
Loader function for registering the external BSF4Rexx functions.

BSFLoadJava
Loads Java. Optionally pass each startup Java argument as its own argument to this function. If the argument is given, the environment CLASSPATH value will not be used for starting up Java.

BSFQueryAllFunctions
Returns a stem array denoting all external BSF4Rexx functions defined in BSF4Rexx.dll (OS/2, Windows) resp. libBSF4Rexx.so (Linux, Unix).

BSFQueryRegisteredFunctions
Returns a stem array denoting all external BSF4Rexx functions that are registered and can be therefore used.

BSFUnloadJava
Unloads Java (no effect at present).

BSFVersion
Returns the version string of the BSF4Rexx dynamic link library, in the form xnn.yyyyMMdd_rexengine-package-name.jar, where xnn denotes the major version number, yyyy the year, month, and day; meanwhile xyyyyMMdd the date, after the space the Java package name of the Java BSF Rexx engine is given.

Loading the External BSF4Rexx Functions

if nxFuncQuery("BSF") = 1 then /* not registered yet, hence load it! */
do
  call nxFuncAdd "BSFLoadFuncs", "BSF4Rexx", "BSFLoadFuncs"
call BSFLoadJava /* registers all BSF4Rexx functions */
call BSFLoadJava /* loads Java */
end

Subfunctions of BSF()

The external Rexx function BSF() is the main interface to Java from Rexx and resides in the BSF4Rexx.DLL/so. As such this function offers a wealth of functionality, organized into subfunctions, e.g.: call BSF 'sleep', 1.05 /* sleep 1050 msec, invoked as a procedure */
or res=BSF('sleep', 1.05) /* sleep 1050 msec, invoked as a function */

Sometimes return values or arguments indicate that no value is supplied. For that purpose one uses the string NIL, which represents the Java value null.

In very rare cases it is necessary to indicate the exact type of an argument. All subfunction names containing the string typeIndicator expect a typeldentifier to precede each argument, according to the following table (bold characters in typeldentifier strings must be at least given):

<table>
<thead>
<tr>
<th>TypeIndicator</th>
<th>The Immediately Following Argument is of Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>boolean, ie. the value {false} or {true}</td>
</tr>
<tr>
<td>Byte</td>
<td>a byte value</td>
</tr>
<tr>
<td>Char</td>
<td>a char, i.e. a single (UTF8) character</td>
</tr>
<tr>
<td>Double</td>
<td>a double value</td>
</tr>
<tr>
<td>Float</td>
<td>a float value</td>
</tr>
<tr>
<td>Int</td>
<td>int, i.e. an integer value</td>
</tr>
<tr>
<td>Long</td>
<td>a long value</td>
</tr>
<tr>
<td>Object</td>
<td>a Java object stored in the BSF registry</td>
</tr>
<tr>
<td>Short</td>
<td>a short value</td>
</tr>
<tr>
<td>String</td>
<td>a String value (UTF8)</td>
</tr>
</tbody>
</table>

In the following subfunctions beanName denotes the key in the BSF registry referencing the desired Java object (the bean):

1. addEventListener, beanName, eventSetName, eventName[, eventText]
2. addEventListenerReturningEventIndex, beanName, eventSetName[, eventText], eventSendBackData
3. arrayAt, arrayBeanName, idx0 [, idx1]...
4. arrayAt, arrayBeanName, intArrayBean
5. arrayPut, arrayBeanName, newValue, idx0 [, idx1]...
6. arrayPut, arrayBeanName, newValue, intArrayBean
7. createArray, JavaClass, beanName, dim0 [, dim1]...
8. createArray, JavaClass, beanName, intArrayBean
9. exit ([, exitCode], [, timeWaitInMsec])
10. getFieldValue, beanName, fieldName
11. getStaticValue, beanName, fieldName
12. invoke, beanName, methodNames [argument]...
13. loadClass, JavaClassName
14. lookupBean, beanName
15. pollEventText [, timeout_in_msec]
16. postEventText, eventText[, priority] {priority: 0=low, 1=normal, 2=high}
17. registerBean, new [beanName], JavaClassName [argument]...
18. registerBeanStr[registerBean, JavaClassName]....
19. setFieldValue, beanName, fieldName, newValue
20. setStaticValue, JavaClassName, fieldName, newValue
21. setRexxNullString, newValue
22. sleep, time2sleep_in_seconds
23. unregisterBean, beanName
24. version
25. wrapArray, arrayBeanName
26. wrapEnumeration, enumerationBeanName

Preregistered Java Objects (BSF Registry)

To ease creating Java array objects, the most important Java class objects are preregistered in the BSF registry on the Java side. Please note: the name of the BSF registry keys for class objects representing the primitive datatypes byte, char, short, int, long, float and double start with a lower case letter:

<table>
<thead>
<tr>
<th>BeanName (Key for Registry)</th>
<th>References the Java Class Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array.class</td>
<td>java.lang.reflect.Array</td>
</tr>
<tr>
<td>Class.class</td>
<td>java.lang.Class</td>
</tr>
<tr>
<td>Method.class</td>
<td>java.lang.reflect.Method</td>
</tr>
<tr>
<td>Object.class</td>
<td>java.lang.Object</td>
</tr>
<tr>
<td>String.class</td>
<td>java.lang.String</td>
</tr>
<tr>
<td>System.class</td>
<td>java.lang.System</td>
</tr>
<tr>
<td>Thread.class</td>
<td>java.lang.Thread</td>
</tr>
<tr>
<td>boolean.class</td>
<td>boolean (primitive datatype)</td>
</tr>
<tr>
<td>java.lang.Boolean</td>
<td>byte (primitive datatype)</td>
</tr>
<tr>
<td>java.lang.Byte</td>
<td>byte (primitive datatype)</td>
</tr>
<tr>
<td>char (primitive datatype)</td>
<td>char (primitive datatype)</td>
</tr>
<tr>
<td>Character.class</td>
<td>java.lang.Character</td>
</tr>
<tr>
<td>double.class</td>
<td>double (primitive datatype)</td>
</tr>
<tr>
<td>java.lang.Double</td>
<td>double (primitive datatype)</td>
</tr>
<tr>
<td>float.class</td>
<td>float (primitive datatype)</td>
</tr>
<tr>
<td>java.lang.Float</td>
<td>float (primitive datatype)</td>
</tr>
<tr>
<td>Integer.class</td>
<td>int (primitive datatype)</td>
</tr>
<tr>
<td>java.lang.Integer</td>
<td>long (primitive datatype)</td>
</tr>
<tr>
<td>java.lang.Long</td>
<td>long (primitive datatype)</td>
</tr>
<tr>
<td>short.class</td>
<td>short (primitive datatype)</td>
</tr>
<tr>
<td>java.lang.Short</td>
<td>short (primitive datatype)</td>
</tr>
<tr>
<td>void.class</td>
<td>void (primitive datatype)</td>
</tr>
<tr>
<td>java.lang.Void</td>
<td></td>
</tr>
</tbody>
</table>

Rony_G_Flatscher_WU(http://www.wu-wien.ac.at/english/)
ooRexx Interface (Module BSF.CLS)

The object-oriented interface support for ooRexx is realized by calling or requiring the ooRexx module BSF.CLS, which defines public routines, classes and the environment symbol BSF4REXX (a directory containing BSF objects). You can get at that support in one of two ways:

```
call BSF.CLS / make ooRexx support available */
or
 requires BSF.CLS /* make ooRexx support available */
```

Sometimes return values or arguments indicate that no value is supplied. For that purpose the ooRexx object .nil (an environment symbol, hence not quoted) is used, which represents the Java value null.

Most of the BSF4REXX subfunctions are made available as class or instance methods of the public class BSF, prepended with the string 'bsf'.

The public class BSF is used for representing Java (class) objects. Its instances are proxy objects which forward received messages to the Java side for invocation. When such ooRexx proxy objects get garbage collected, they will cause the BSF registry reference counter of the represented Java object to be decreased. If that reference counter drops to zero, the Java object gets removed from the BSF registry.

Public Routines

1. bsf.createArray(...)
2. bsf.exit(...)
3. bsf.getFieldValue(...)
4. bsf.getFieldValueStrict(...)
5. bsf.invoke(...)
6. bsf.setFieldValue(...)
7. bsf.setFieldValueStrict(...)
8. bsf.setPropertyValue(...)
9. bsf.setPropertyValueStrict(...)
10. bsf.getClass(...)
11. bsf.getOption(...)
12. bsf.getProperties(...)
13. bsf.getReturnType(...)
14. bsf.getException(...)

BSF’s class methods

1. bsf.createClass(...)
2. bsf.createPackage(...)
3. bsf.createClass(...)
4. bsf.getStaticValue(...)
5. bsf.getStaticValueStrict(...)
6. bsf.import(...)
7. bsf.loadClass(...)
8. bsf.lookupBean(...)
9. bsf.postEventText(...)
10. bsf.setNull(...)
11. bsf.setStaticValue(...)
12. bsf.setStaticValueStrict(...)
13. bsf.setPropertyValue(...)
14. bsf.setPropertyValueStrict(...)

Private Class BSF_ARRAY_REFERENCE

BSF ARRAY REFERENCE is a subclass of BSF that allows interacting with Java array objects (stored in the BSF registry) as if they were ooRexx primitive arrays. Only available, if the ooRexx object is a Java array object proxy (a Java object in the BSF registry) or a Java array class object proxy was created with bsf.import().

The public routine bsf.wrap will use this class to create the ooRexx proxy object, if it detects that the supplied BSF registry key refers to a Java array object.

Public Class BSF_PROXY

BSF_PROXY is a subclass of BSF that allows creating proxy objects by passing the BSF registry key (a string) to its init method. This allows sending ooRexx messages which will cause the appropriate Java methods to be invoked.

Unlike direct instances of the BSF class, these proxy objects, if garbage collected, do not decrease the BSF registry counter.

Directory Object .BSF4REXX

.BSF.CLS will initialize a directory object to contain proxies to all preregistered Java objects in the BSF registry. As ooRexx will translate statements into uppercase before executing them, the directory index values Byte.class and byte.class cannot be distinguished anymore as they both become BYTE_CLASS. Therefore the index value of the primitive class object references do not carry the suffix .CLASS.

Get the Java version by querying the Java class object java.lang.System:

```
call bsf.cls /* load the ooRexx support for BSF4REXX */
msg .bsf4rexx-system-class -getProperties('java.version') /* or create a two dimensional Java array (3 by 4 elements) of the primitive datatype int */
```

References the Java Class Object

<table>
<thead>
<tr>
<th>Index</th>
<th>References the Java Class Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>java.lang.reflect.Class</td>
</tr>
<tr>
<td>Class</td>
<td>java.lang.Object</td>
</tr>
<tr>
<td>Method</td>
<td>java.lang.reflect.Method</td>
</tr>
<tr>
<td>Object</td>
<td>java.lang.Object</td>
</tr>
<tr>
<td>String</td>
<td>java.lang.String</td>
</tr>
<tr>
<td>System</td>
<td>java.lang.System</td>
</tr>
<tr>
<td>Thread</td>
<td>java.lang.Thread</td>
</tr>
<tr>
<td>Boolean</td>
<td>java.lang.Boolean</td>
</tr>
<tr>
<td>Byte</td>
<td>java.lang.Byte</td>
</tr>
<tr>
<td>Char</td>
<td>java.lang.Char</td>
</tr>
<tr>
<td>Character</td>
<td>java.lang.Character</td>
</tr>
<tr>
<td>Double</td>
<td>java.lang.Double</td>
</tr>
<tr>
<td>Float</td>
<td>java.lang.Float</td>
</tr>
<tr>
<td>Int</td>
<td>java.lang.Int</td>
</tr>
<tr>
<td>Integer</td>
<td>java.lang.Integer</td>
</tr>
<tr>
<td>Long</td>
<td>java.lang.Long</td>
</tr>
</tbody>
</table>

Public Class BSF.DIALOG

Supplies the methods messagebox(), dialogbox() or inputbox(), via the class object .BSF.DIALOG or an instance of it.

```
1. messagebox(message, [title], type=question) always returns nil
2. dialogbox(message, [title], [type], [optionType], [icon], [txtOptions], [defaultTxtButton], [defaultTxtButton] returns button number (0-first button)
3. inputbox(message, [title], [type], [icon], [txtOptions], [defaultTxtOption] returns entered/chosen text
```

Bridging_OpenObject_Rexx_(ooRexx_and_java)

Rory_G_Flatshcer_WU (http://www.wwu.edu)