External BSF4ooRexx Functions - Overview

The external BSF4ooRexx functions allow interfacing with Java. If the Rexx script was invoked by Java, then the external Rexx function `BSF()` is registered already.

BSF(args) Main function to interface with Java, see *'Subfunctions of BSF()'*.

BsfContextVariables("get" [, name] | "set", {name, val | dir} | "drop", {name | dir}**)** Gets, creates, drops context variables.

BsfCreateRexxProxy(rexxObj[,[userData] [,jClz [,arg]…]]**)** Creates and returns a Java RexxProxy object ("proxy") allowing Java to send messages to rexxObj. jClz is either a Java interface class (arg may be one or more additional Java interface classes) or an abstract Java class (arg may be an argument for a constructor).

BsfDropFuncs() Drops all registered BSF4ooRexx functions.

BsfGetTID() ooRexx multithreading support: returns the current ooRexx thread id

BsfInvokedBy() Returns 0, if no Java is present, returns 1 if BSF4ooRexx was invoked by Java, returns 2 if BSF4ooRexx was invoked by Rexx.

BsfJavaException("throw",throwable**)** Throws a Java exception.

BsfLoadFuncs() Loader function for registering the external BSF4ooRexx functions.

BsfLoadJava([args]) Loads Java. Optionally pass each startup Java argument as its own argument to this function. If the -Djava.class.path=... argument is given, the environment CLASSPATH value will not be used for starting up Java.

BsfQueryAllFunctions() Returns a stem array denoting all external BSF4ooRexx functions defined in BSF4ooRexx.dll (Windows) resp. libBSF4ooRexx.so (Linux/Unix).

BsfQueryRegisteredFunctions() Returns a stem array denoting all external BSF4ooRexx functions that are registered and can be therefore used.

BsfRawBytes(string|javaByteArray[,length]**)** If string given, returns a Java byte array, if javaByteArray (must be an instance of .BSF) given, returns a Rexx string; no codepage translations will take place.

BsfRexxProxy(proxy[,"ooRexxObject"|"userData"|"refCount"]**)** Returns proxied Rexx object (default), the associated userData Rexx object or the total number of RexxProxy references for the Rexx object.

BsfShowErrorMessage([0|1]) Returns 1, if Java exception messages are displayed, 0 else. Supplying an argument of "1" turns on displaying Java exception messages (default), a value of "0" turns off showing Java error messages. In case of a Java exception, a Rexx variable named BSF_ERROR_MESSAGE is set to the Java exception message.

BsfUnLoadJava() Unloads Java.

BSFVersion() Returns the version string of the BSF4ooRexx dynamic link library, in the form xnn.yyyymmdd_rexx-engine-package-name bitness, where x denotes the major version number, nn the minor version number, and yyyymmdd the date; after the space the Java package name of the Java BSF Rexx engine is given.

Loading the External BSF4ooRexx Functions

```
if rxFuncQuery("BSF") = 1 then /* not registered yet, hence load it! */
do
   call rxFuncAdd "BsfLoadFuncs", "BSF4ooRexx", "BsfLoadFuncs"
   call BsfLoadFuncs          /* registers all BSF4ooRexx 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 BSF4ooRexx 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 strict expect a typeIndicator to precede each argument, according to the following table (bold characters in typeIndicator strings must be at least given):

TypeIndicator	The Immediately Following Argument is of Type
"**BO**olean"	boolean, ie. the value 0 (false) or 1 (true)
"**BY**te"	a byte value
"**C**har"	char, ie. a single (UTF8) character
"**D**ouble"	a double value
"**F**loat"	a float value
"**I**nt"	int, ie. an integer value
"**L**ong"	long" a long value
"**O**bject"	a Java object stored in the BSF registry
"**SH**ort"	a short value
"**ST**ring"	a String value (UTF8)

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 *deprecated!*]
2. arrayAt, arrayBeanName, idx0 [, idx1]…
   arrayAt, arrayBeanName, intArrayBean
3. arrayLength beanName
4. arrayPut, arrayBeanName, newValue, idx0 [, idx1]…
   arrayPut, arrayBeanName, newValue, intArrayBean
5. arrayPut*Strict*, arrayBeanName, *typeIndicator*, newValue, idx0 [, idx1]…
   arrayPut*Strict* arrayBeanName, *typeIndicator*, newValue, intArrayBean
6. bsfPrefixReturnValue([0|1]) (if '1': prefix '<O>' and '<S>' added to all)
7. createJavaArray, JavaClassObjectBeanName, dim0 [, dim1]…
   createJavaArray, JavaClassObjectBeanName, intArrayBean
8. createJavaArrayOf, JavaClassObjectBeanName [, element]…
9. exit [, [exitCode] [, time2wait_in_msec]]
10. getBSFManager
11. getFieldValue,   beanName, fieldName
    getFieldValue*Strict*, beanName, fieldName
12. getPropertyValue, beanName, propertyName, index|.NIL
13. getStaticValue,   JavaClassName, fieldName
    getStaticValue*Strict*, JavaClassName, fieldName
14. haltThread, tid
15. invoke,   beanName, methodName [, argument]…
    invoke*Strict*,   beanName, methodName [, *typeIndicator*, argument]…
16. isA,   beanName, javaClassName
17. loadClass, JavaClassName
18. lookupBean, beanName
19. new|registerBean, [beanName], JavaClassName [, argument]…
20. newStrict|registerBean*Strict*, [beanName], JavaClassName - [, *typeIndicator*, argument]…
21. [pollEventText [, timeout_in_msec] *deprecated!*]
22. [postEventText, eventText[, priority] (priority: 0=low,1=normal,2=high) *deprecated!*]
23. rawRegisterBean, beanName, object
24. setFieldValue,   beanName, fieldName, newValue
    setFieldValue*Strict*, beanName, fieldName, [*typeIndicator*,] newValue
25. setPropertyValue,   beanName, propertyName, index|.NIL, newValue
    setPropertyValue*Strict*,   beanName, propertyName, index|.NIL, *typeIndicator*, newValue
26. setRexxNullString, newString
27. sleep, time2sleep_in_seconds
28. unregisterBean, beanName
29. version ["All"|"ARrayWrapper"|"BsfManager"|"EnumerationWrapper"| "Java4Rexx"|"REXXAndJava"|"REXXEngine"|"Supplier"]
30. wrapArray, arrayBeanName

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:*

BeanName (Key for Registry)	References the Java Class Object
"Array.class"	java.lang.reflect.Array
"Class.class"	java.lang.Class
"Method.class"	java.lang.reflect.Method
"Object.class"	java.lang.Object
"String.class"	java.lang.String
"System.class"	java.lang.System
"Thread.class"	java.lang.Thread
"**b**oolean.class"	boolean (primitive datatype)
"Boolean.class"	java.lang.Boolean
"**b**yte.class"	byte (primitive datatype)
"Byte.class"	java.lang.Byte
"**c**har.class"	char (primitive datatype)
"Character.class"	java.lang.Character
"**d**ouble.class"	double (primitive datatype)
"Double.class"	java.lang.Double
"**f**loat.class"	float (primitive datatype)
"Float.class"	java.lang.Float
"**i**nt.class"	int (primitive datatype)
"Integer.class"	java.lang.Integer
"**l**ong.class"	long (primitive datatype)
"Long.class"	java.lang.Long
"**s**hort.class"	short (primitive datatype)
"Short.class"	java.lang.Short
"**v**oid.class"	void (primitive datatype)
"Void.class"	java.lang.Void

ooRexx Interface (Package BSF.CLS)

The object-oriented interface support for ooRexx is realized by requiring the ooRexx package BSF.CLS, which defines public routines, classes and the environment symbol .BSF4REXX (a directory containing BSF objects). You can get at that support by simply requiring it:

```
::requires BSF.CLS  /* make oo-like BSF4ooRexx supoort available */
```

If Java needs to be started by BSF.CLS it honors the Java startup options (like setting the Java heap size) given in the optional shell environment variable named "BSF4Rexx_JavaStartupOptions". It sets bsfPrefixReturnValue.

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. Java errors get stored in the environment (.BSF_ERROR_MESSAGE).

Most of the BSF4ooRexx 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. box(typeIndicator, value) wraps value as a Java wrapper object of type typeIndicator and returns the reference to it
2. bsf.createJavaArray(…) see BSF-subfunction
3. bsf.createJavaArrayOf(…) see BSF-subfunction
4. bsf.getConstant(JavaClassName, fieldName)
5. bsf.getStaticValue(…) see BSF-subfunction
6. bsf.getStaticValueStrict(…) see BSF-subfunction
7. bsf.haltThread(…) see BSF-subfunction
8. bsf.importClass(javaClassName[, .nil | name-for-.local-entry])
9. bsf.loadClass(…) see BSF-subfunction
10. bsf.lookupBean(…) see BSF-subfunction
11. bsf.pollEventText(…) see BSF-subfunction
12. bsf.postEventText(…) see BSF-subfunction
13. bsf.redirectTo(["J[ava]"|["oorexx"]] [,newIn] [,newOut] [,newErr] [,newDebugIn] [,newTraceOut]) redirect ooRexx' .input, .output, .error, .debugInput, .traceOutput to/from the standard Java streams
14. bsf.scriptContext2Rexx(ScriptContext[, "A" | "G" | "E" | num[, num…]])
15. bsf.terminateRexxEngine() if Rexx loaded Java, then the Rexx Java engine gets terminated to inhibit callbacks from Java
16. bsf.unregisterBean(…) see BSF-subfunction
17. bsf.version(…) see BSF-subfunction
18. bsf.wrap(strBSFRegistryKey) returns an ooRexx proxy object, if string is a Java object in the BSF registry, otherwise the supplied argument
19. bsf.wrapStaticFields(javaClassName) returns an ooRexx directory object
20. iif(truthValue, valueIfTrue, valueIfFalse) returns valueIfTrue, if first argument is .true, returns valueIfFalse else
21. pp(argument) returns argument's string value enclosed in square parenthesis
22. unbox(o) returns the value from Java's primitive datatype wrapper object o

Public Class BSF

This is the ooRexx proxy class for representing Java classes. ooRexx messages sent to it or its instances cause the invocation of the appropriate Java methods. All methods starting with bsf. are pass-through methods and their arguments are documented in the BSF subfunctions on the other page.
If the ooRexx proxy object gets garbage collected by ooRexx, the

---

reference counter for the represented Java object in the BSF registry gets decreased (if that counter hits zero, the Java object is removed from the BSF registry, such that it can be garbage collected by Java).

BSF's CLASS METHODS

1. bsf.createJavaArray(…) see BSF-subfunction
1. bsf.createJavaArrayOf(…) see BSF-subfunction
2. bsf.exit(…) see BSF-subfunction
3. bsf.getStaticValue(…) see BSF-subfunction
4. bsf.getStaticValueStrict(…) see BSF-subfunction
5. bsf.haltThread(…) see BSF-subfunction
6. bsf.importClass(JavaClassName [, .nil | name-for-.local-entry])
7. bsf.loadClass(JavaClassName) see BSF-subfunction
8. bsf.lookupBean(…) see BSF-subfunction
9. bsf.pollEventText(…) see BSF-subfunction
10. bsf.postEventText(…) see BSF-subfunction
11. bsf.setRexxNullString(…) see BSF-subfunction
12. bsf.sleep(…) see BSF-subfunction
13. bsf.wrapArray(…) see BSF-subfunction
14. newStrict([typeIndicator, argument]…) only available, if the ooRexx class object proxy was created with bsf.import(); creates an instance of the given Java class using typed arguments (as opposed to the new method) and returns an ooRexx proxy object

BSF's INSTANCE METHODS

1. bsf.class returns Java class object proxy
2. bsf.dispatch(methName[,args]…) forwards to method bsf.invoke(…)
3. bsf.exit(…) see BSF-subfunction
4. bsf.invoke(…) see BSF-subfunction
5. bsf.invokeStrict(…) see BSF-subfunction
6. bsf.isA(javaClass[javaInterface])
7. bsf.getFieldValue(…) see BSF-subfunction
8. bsf.getFieldValueStrict(…) see BSF-subfunction
9. bsf.setFieldValue(…) see BSF-subfunction
10. bsf.setFieldValueStrict(…) see BSF-subfunction
11. bsf.getPropertyValue(…) see BSF-subfunction
12. bsf.setPropertyValue(…) see BSF-subfunction
13. bsf.setPropertyValueStrict(…) see BSF-subfunction

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 arrays (e.g. index values start with 1, and the ooRexx array methods AT, [], DIMENSION, ITEMS, MAKEARRAY, PUT, []=, SUPPLIER are implemented).

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 BSF4ooRexx   */
say .bsf4rexx~system.class ~getProperty('java.version')
```

or create a two dimensional Java array (3 by 4 elements) of the primitive datatype int:

```
a=bsf.createArray(.bsf4rexx~int, 3, 4) /* create Java int array (3x4) */
::requires BSF.CLS   /* load the ooRexx support for BSF4ooRexx   */
```

Index	References the Java Class Object
ARRAY.CLASS	java.lang.reflect.Array
BOOLEAN	boolean (primitive datatype)
BOOLEAN.CLASS	java.lang.Boolean
BYTE	byte (primitive datatype)
BYTE.CLASS	java.lang.Byte
CHAR	char (primitive datatype)
CHARACTER.CLASS	java.lang.Character
CLASS.CLASS	java.lang.Class
DOUBLE	double (primitive datatype)
DOUBLE.CLASS	java.lang.Double
FILE.SEPARATOR	File separator (e.g. "/", "\")
FLOAT	float (primitive datatype)
FLOAT.CLASS	java.lang.Float
INT	int (primitive datatype)
INTEGER.CLASS	java.lang.Integer
LINE.SEPARATOR	Line end character(s)
LONG	long (primitive datatype)
LONG.CLASS	java.lang.Long
METHOD.CLASS	java.lang.reflect.Method
OBJECT.CLASS	java.lang.Object
PATH.SEPARATOR	Path separator (e.g. ":", ";")
SHORT	short (primitive datatype)
SHORT.CLASS	java.lang.Short
STRING.CLASS	java.lang.String
SYSTEM.CLASS	java.lang.System
THREAD.CLASS	java.lang.Thread
void	void (primitive datatype)
VOID.CLASS	java.lang.Void

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]) always returns .nil
2. dialogBox(message, [title], [type], [optionType], [icon], [txtButtons], [defaultTxtButton]) returns button number (0=first button)
3. inputBox(message, [title], [type], [icon], [txtOptions], [defaultTxtOption]) returns entered/chosen text

where:
type one of error, information, plain, question, warning
optionType one of default, OkCancel, YesNo, YesNoCancel

```
.bsf.dialog~messageBox("Think about it!")  /* using the class object */

    /* create and show a frame object for which a modal dialog is used */
f=.bsf~new("java.awt.Frame", "Hello!")~~pack~~show /* create and show */
fdlg=.bsf.dialog~new(f)      /* create an instance, tell it about 'f' */
say fdlg~dialogBox("Continue?", , "warning", "YesNo") /* modal to 'f' */
```