"The 2009 Edition of BSF4Rexx"
Part 1, Introduction to BSF4Rexx

2009 International Rexx Symposium
Chilworth, England (May 2009)

Rony G. Flatscher (Rony.Flatscher@wu.ac.at)
Wirtschaftsuniversität Wien, Austria (http://www.wu.ac.at)
Agenda

• Brief History
• Architecture
• Changes
  – Examples
• New features
  – Examples
• Roundup and Outlook 4.0
• 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](http://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

BSF (Java)
- BSF Registry
- BSF4Rexx (C++)
- JNI
- BSF
  - BsfDropFuncs()
  - BsfInvokedBy()
  - BsfLoadFuncs()
  - BsfLoadJava()
  - BsfQueryAllFunctions()
  - BsfQueryRegisteredFunctions()
  - BsfUnloadJava()
  - BsfVersion()

Java Program
- RexxEngine
- RexxAndJava

Rexx scripts
Java Invoking a Rexx Script
An Example

```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
**BSF.CLS**

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()**
  - BsfDropFuncs()
  - BsfInvokedBy()
  - BsfLoadFuncs()
  - BsfLoadJava()
  - BsfQueryAllFunctions()
  - BsfQueryRegisteredFunctions()
  - BsfUnloadJava()
  - BsfVersion()

**Java Program**

- BSF (Java)
- JNI
- BSF4Rexx

(RexxEngine)

- BSF Registry
- RexxAndJava

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

- BSF.cls
- ooRexx scripts
/* "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: 1.6.0_11
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:

dependent on the installed Java version

java.version: 1.6.0_11

/* "getJavaVersion.rex": classic Rexx version, querying the installed Java version */

s=bsf.import('java.lang.System')  /* import the Java class 'java.lang.System' */

say "java.version:" s~getProperty('java.version')

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

/* "getJavaVersion.rex": classic Rexx version, querying the installed Java version */

say "java.version:" .bsf4rexx~system.class ~getProperty('java.version')

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

• 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 – Vienna Features

• Sometimes one needs to supply primitive datatypes embedded in Java classes

• Public routines `box()`, `unbox()`

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

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

Possible Output:

```
java.lang.Long@be991a08
123456789012
```

123456789012
BSF4Rexx

BSF.CLS – Vienna Features

• 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 – Vienna Features

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

Possible Output:

```
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 */
```

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

BSF.CLS – Vienna Features

- 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
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, i.e. `arr[1][1]`
2. `.nil` or data as supplied by ooRexx when event adapter was set up
3. String denoting the event name that has occurred
4. `.nil` or string of event names to react upon
5. A reference to the BSFManager instance
BSF4Rexx - BSF.Dialog

BSF.CLS – Vienna Features

- Public class `bsf.dialog`
- Multiplatform, uses Java's swing GUI
- Dialog (class or instance) methods

```plaintext
res=.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.
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

- `setupBSF.rex [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
BSF4Rexx – Vienna Version Goodies, 2

• 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 ooRexx
  – OpenOffice.org/StarOffice automation examples
Roundup and Outlook

• Vienna Version of BSF4Rexx
  – Introduces typeless interaction with Java
  – Adds utility routines for easying 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
Input for BSF4Rexx 4.0

• 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