2015 RexxLA International Rexx Language Symposium Proceedings

René Vincent Jansen (ed.)

THE REXX LANGUAGE ASSOCIATION
RexxLA Symposium Proceedings Series
ISSN 1534-8954

Publication Data

A publication of **RexxLA Press**

The responsible publisher of this edition is identified as *IBizz IT Services and Consultancy*, Amsteldijk 14, 1074 HR Amsterdam, a registered company governed by the laws of the Kingdom of The Netherlands.

The RexxLA Symposium Series is registered under ISSN 1534-8954
The 2015 edition is registered under ISBN 978-94-032-7936-7

2023-05-31 First printing

Introduction

History of the International Rexx Language Symposium

In 1990, Cathie Dager of SLAC[1] convened the organizing committee for the first independent Rexx[2] Symposium for Developers and Users. SLAC continued to organize this annual event until the middle of the 1990's when the RexxLA took over that responsibility. Symposia have been held annually since 1990.

About RexxLA

During the 1993 Symposium in La Jolla, California, plans for a Rexx User Group materialized. The Rexx Language Association (RexxLA), as it was called, is an independent, non-profit organization dedicated to promoting the use and understanding of the Rexx programming language. RexxLA manages several open source implementations of Rexx.

The selection procedure

Presentation proposals are solicited yearly using a CFP[3] procedure, after which the RexxLA symposium comittee reviews them and votes which presentations are selected for the symposium. The presentations are peer reviewed before being presented. Presenters are not compensated for their presentations.

Location

The 2015 symposium was held in Vienna, Austria from 29 Mar 2015 to 1 Apr 2015.

[1] Stanford Linear Accelerator Center, since 2008 SLAC National Accelerator Laboratory
[2] Cowlishaw, M. F., **The REXX Language** (second edition), ISBN 0-13-780651-5, Prentice-Hall, 1990.
[3] Call For Papers.

Contents

1

Smart Homes with openHAB and ooRexx – Manuel Raffel

Date and Time

30 Mar 2015, 07:30:00 CET

Presenter

Manuel Raffel

Presenter Details

Manuel Raffel is currently about to finish his undergraduate studies in Business, Economics and Social Sciences with a major in Business Administration at the Vienna University of Economics and Business. After having been introduced to the world of ooRexx by one of his professors, Dr. Rony Flatscher, he is currently working on two projects and his thesis, all of them focussing on BSF4ooRexx related topics.

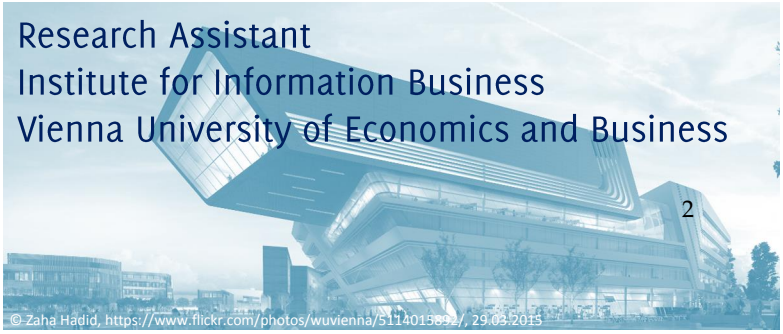
Session Abstract

The presentation is intended to give an introduction to both the openHAB project for home automation and the developed extension which adds support for ooRexx. OpenHAB is a highly extensible, vendor and technology agnostic open source home automation software. The developed binding effectively enables ooRexx to take control of lights, heating, shutters and everything else there is in today's smart homes.

REXXHAB

Bringing ooRexx *into the* Smart Home

Manuel RAFFEL, manuel.raffel@outlook.com



Manuel RAFFEL

TODAY'S *Agenda*

#1
WHAT IS A
Smart Home?

#2
OPENHAB
Empowering the Smart Home

#3
OPENHAB
Demo

#4
REXXHAB
Implementation

#5
REXXHAB
Demo

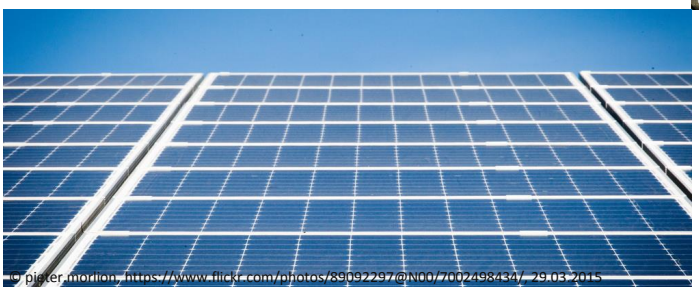
#6
SUMMARY AND
Outlook

#1 WHAT IS A *Smart Home?*

What is a SMART HOME?

SPACE EFFICIENCY?

WASTE WATER RECYCLING?

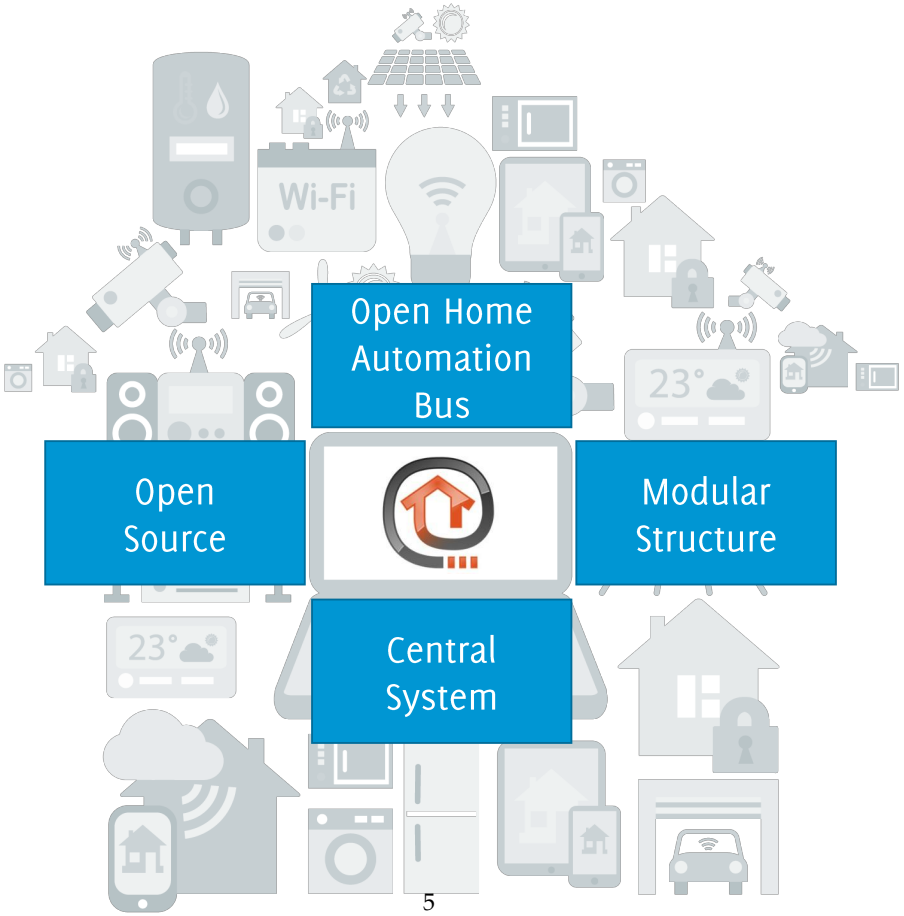


SOLAR POWER?

INTERACTIVE TECHNOLOGY!

#2 openHAB

Empowering the Smart Home



openHAB - Supported Protocols

EnOcean

C-Bus

KNX

DMX

Insteon

X10

One-Wire

Bluetooth

Zigbee

UDP

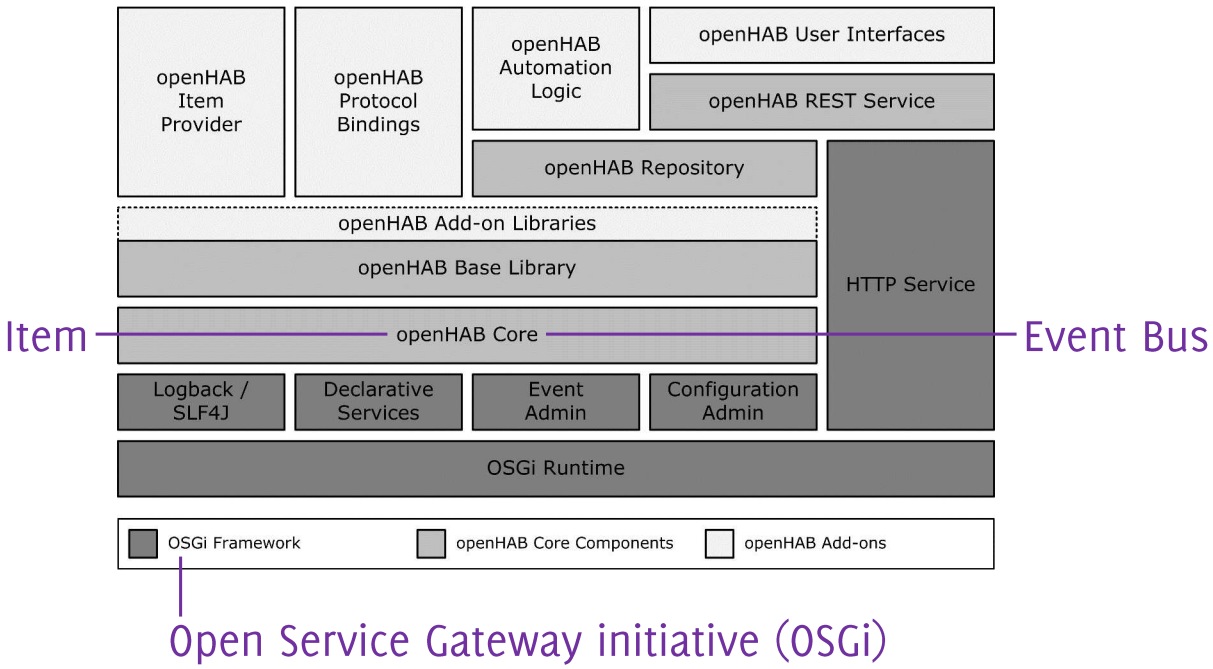
TCP

Universal Powerline Bus (UPB)

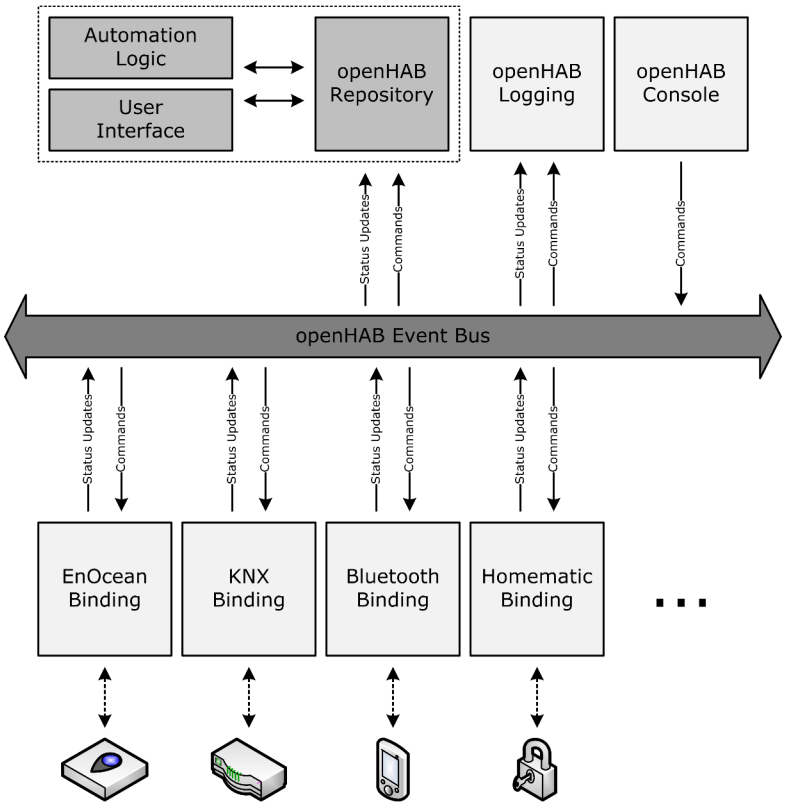
Philips Hue

Z-Wave

openHAB - Architecture

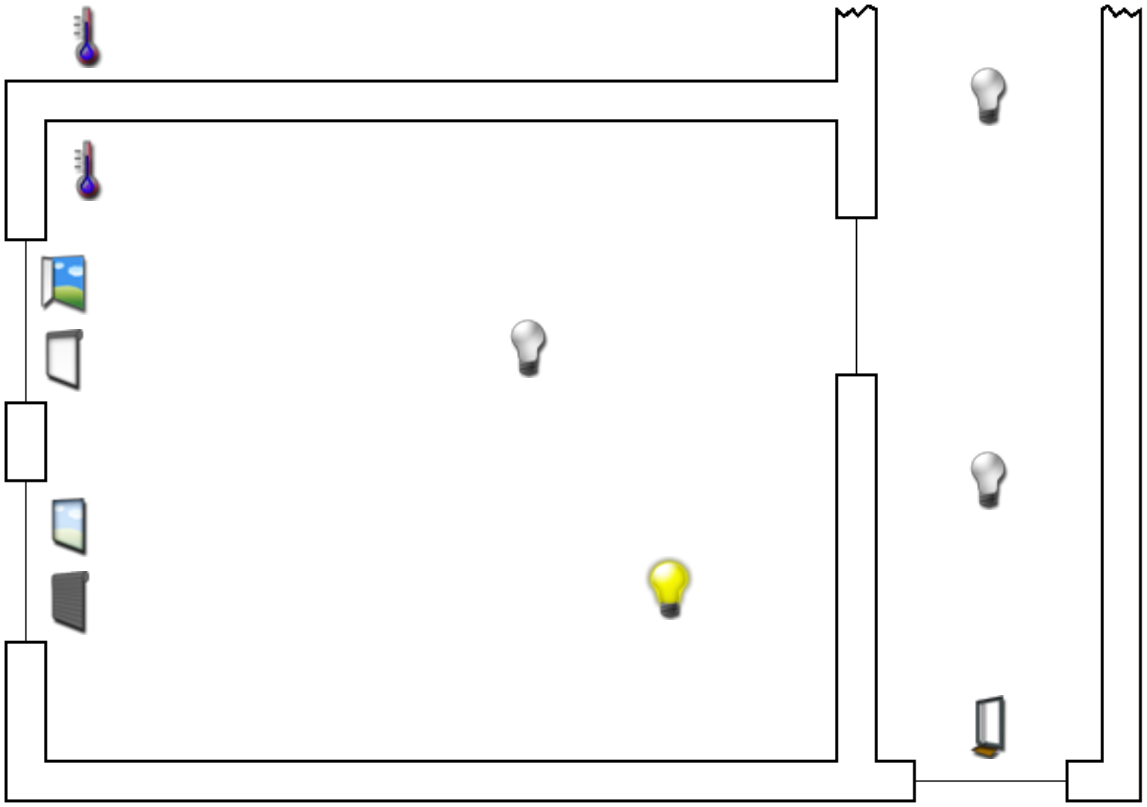


openHAB – Event Bus



#3
openHAB
Demo

openHAB – Demo House



Demo House - Corridor

```
1  Group g_corridor "Corridor" <corridor>
2
3  Switch light_corridor_ceiling_front "Ceiling Front" (g_corridor)
4  Switch light_corridor_ceiling_back "Ceiling Back" (g_corridor)
5
6  Contact door_corridor "Door" (g_corridor)
```

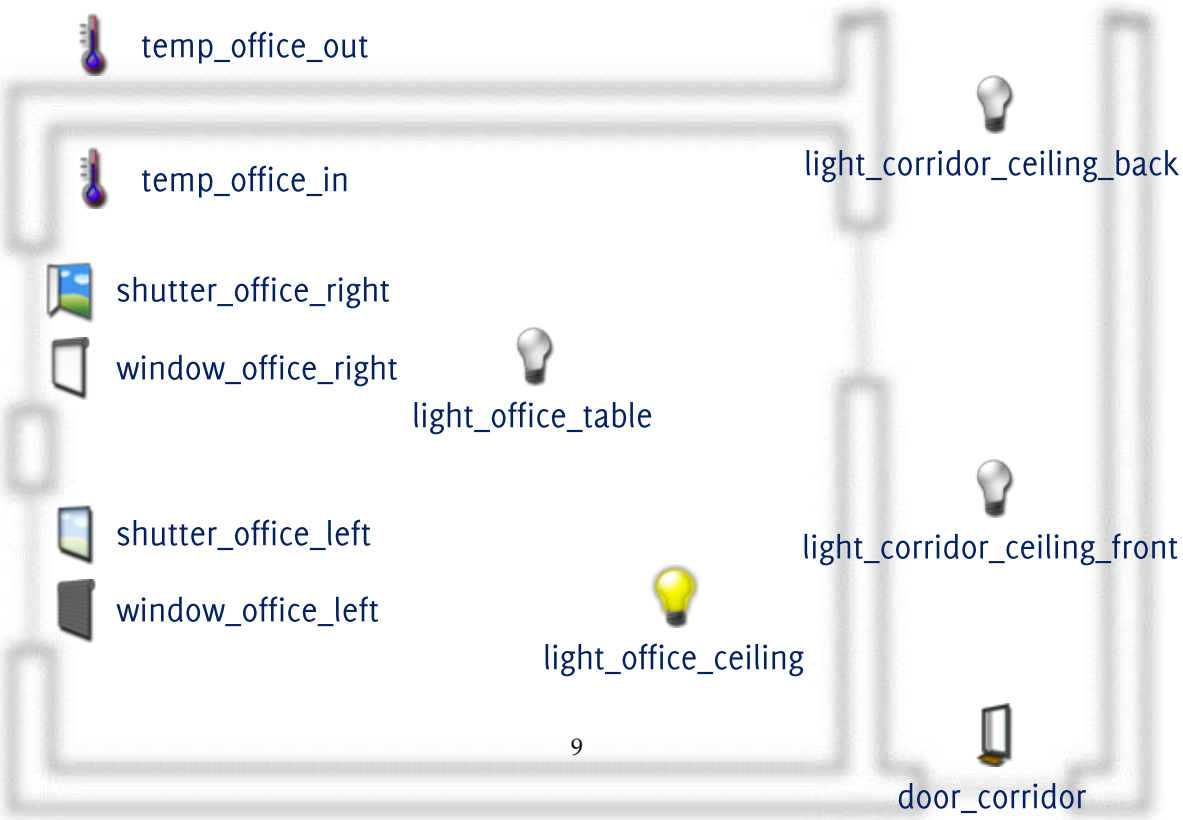
file: openHAB/configurations/items/DemoHouse.items

Demo House - Office

```
1  Group g_office "Office" <office>
2
3  Switch light_office_ceiling "Ceiling" (g_office)
4  Switch light_office_table "Table" (g_office)
5
6  Rollershutter shutter_office_left "Office Left" (g_office)
7  Rollershutter shutter_office_right "Office Right" (g_office)
8
9  Contact window_office_left "Office Left" (g_office)
10 Contact window_office_right "Office Right" (g_office)
11
12 Number temp_office_in "Indoor [%.1f °C]" <temperature> (g_office)
13 Number temp_office_out "Outdoor [%.1f °C]" <temperature> (g_office)
```

file: openHAB/configurations/items/DemoHouse.items

openHAB – Demo House

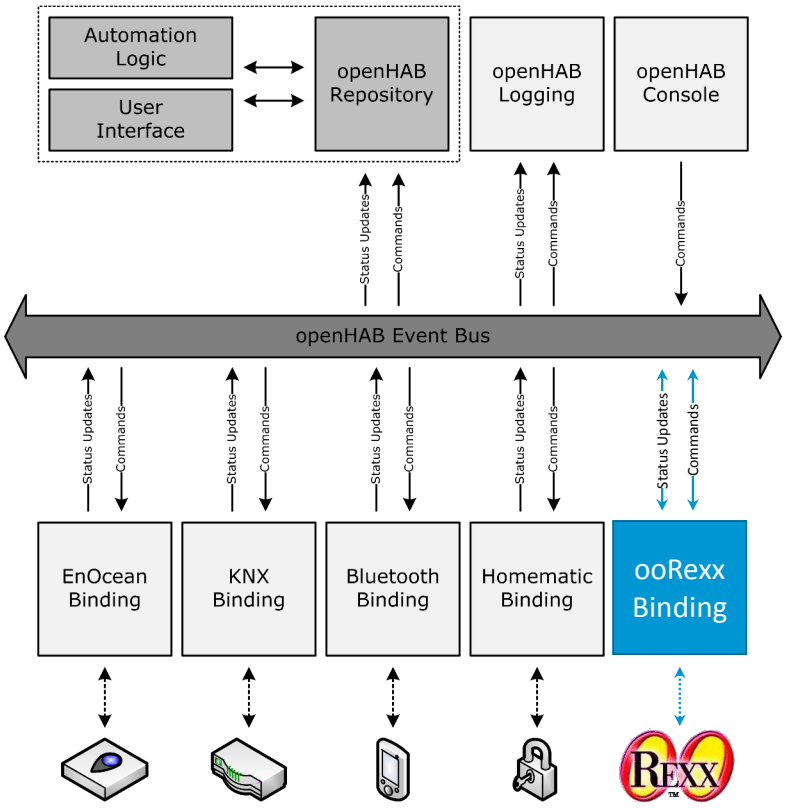


Demo House - Visualization

```
1  sitemap DemoHouse label="DemoHouse"
2  {
3      Frame
4      {
5          Group item=g_corridor label="Corridor" icon="corridor"
6          Group item=g_office label="Office" icon="office"
7      }
8      […]
9  }
```

file: openHAB/configurations/sitemaps/default.sitemap

REXXHAB



ooRexx logo © Julian Choy, http://www.oorexx.org/, 29.03.2015

REXXHAB – EVENTPUBLISHER

```
1  // Synchronous sending of a command.
2  // itemName - name of the item to send the command for
3  // command - the command to send
4  public abstract void sendCommand(String itemName, Command command)
5
6  // Asynchronous sending of a command.
7  // itemName - name of the item to send the command for
8  // command - the command to send
9  public abstract void postCommand(String itemName, Command command)
10
11 // Asynchronous sending of a status update.
12 // itemName - name of the item to send the command for
13 // newState - the new state to send
14 public abstract void postUpdate(String itemName, State newState)
```

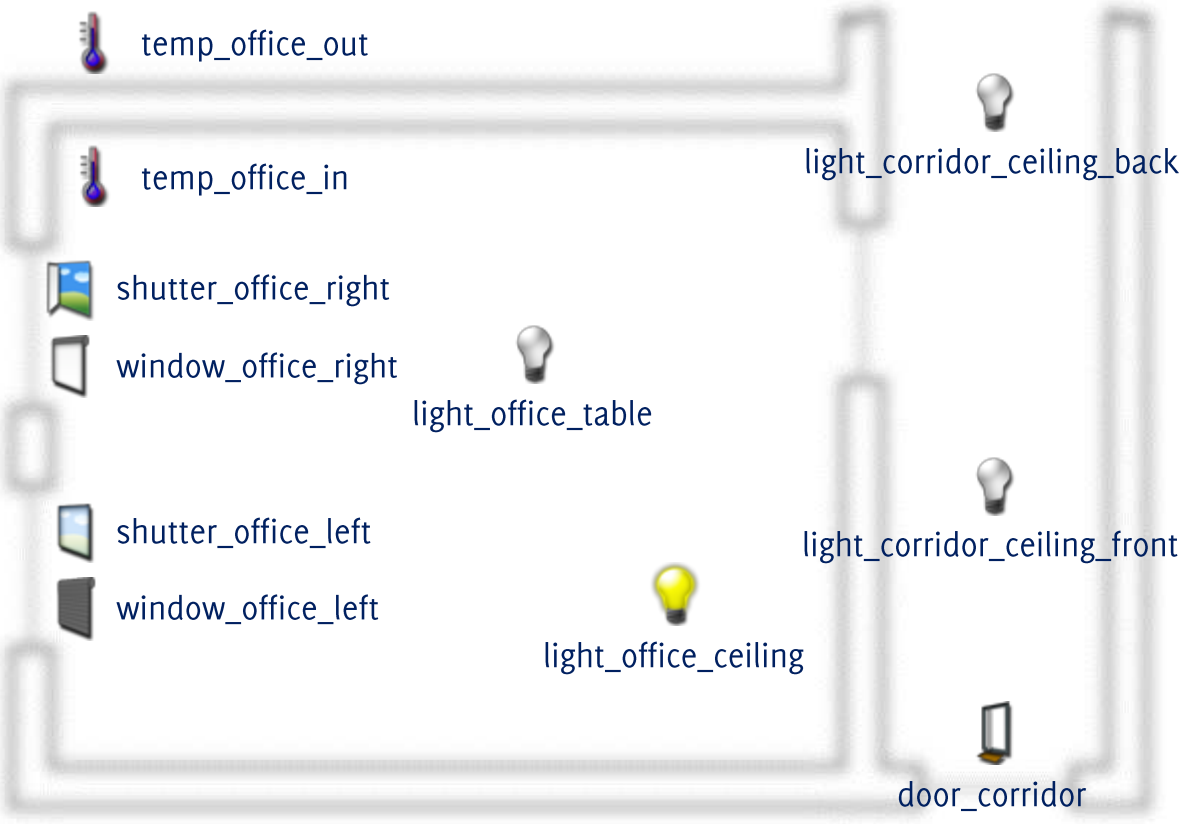
class: org.openhab.core.events.EventPublisher

REXXHAB – OPENHAB.CLS

```
1  OnOffType = bsf.loadClass("org.openhab.core.library.types.OnOffType")
2  OpenClosedType = bsf.loadClass("org.openhab.core.library.types.OpenClosedType")
3  UpDownType = bsf.loadClass("org.openhab.core.library.types.UpDownType")
4
5  openHAB.command~ON = OnOffType~ON
6  openHAB.command~OFF = OnOffType~OFF
7  openHAB.command~OPEN = OnOffType~OPEN
8  openHAB.command~CLOSED = OnOffType~CLOSED
9  openHAB.command~UP = OnOffType~UP
10 openHAB.command~DOWN = OnOffType~DOWN
11
12 openHAB.state~ON = OnOffType~ON
13 openHAB.state~OFF = OnOffType~ON
14 openHAB.state~OPEN = OnOffType~ON
15 openHAB.state~CLOSED = OnOffType~ON
16 openHAB.state~UP = OnOffType~ON
17 openHAB.state~DOWN = OnOffType~ON
```

file: OpenHAB.CLS [excerpt]

rexxHAB – Demo House



Demo House - Corridor

```
1  Group g_corridor "Corridor" <corridor>
2
3  Switch light_corridor_ceiling_front "Ceiling Front" (g_corridor)
↳         { oorexx="command:commandReceived,update:updateReceived" }
4  Switch light_corridor_ceiling_back "Ceiling Back" (g_corridor)
↳         { oorexx="command:commandReceived,update:updateReceived" }
5
6  Contact door_corridor "Door" (g_corridor)
↳          { oorexx="update:updateReceived" }
```

file: openHAB/configurations/items/DemoHouse.items

Demo House - Office

```
Group g_office "Office" <office>

Switch light_office_ceiling "Ceiling" (g_office)
      { oorexx="command:commandReceived,update:updateReceived" }
Switch light_office_table "Table" (g_office)
      { oorexx="command:commandReceived,update:updateReceived" }

Rollershutter shutter_office_left "Office Left" (g_office)
             { oorexx="command:commandReceived,update:updateReceived" }
Rollershutter shutter_office_right "Office Right" (g_office)
             { oorexx="command:commandReceived,update:updateReceived" }

Contact window_office_left "Office Left" (g_office)
       { oorexx="update:updateReceived" }
Contact window_office_right "Office Right" (g_office)
       { oorexx="update:updateReceived" }
```

file: openHAB/configurations/items/DemoHouse.items

Demo House – RexxDemo.rex

```
use arg eventPublisher
.local~openHAB = eventPublisher
return .OpenHABProxy~new
::requires OpenHAB.CLS

::class OpenHABProxy

::method commandReceived
  use arg itemName, command
  say "REXX noticed that " itemName " received command " command

::method updateReceived
  use arg itemName, state
  say "REXX noticed that " itemName " received update " state
```

file: openHAB/configurations/scripts/RexxDemo.rex

Demo House – Coming Home

```
Switch bluetooth_device_in_range
        { bluetooth="45E43B6CA214", oorexx="update:bluetoothDevice" }
```

file: openHAB/configurations/items/DemoHouse.items

```
::method bluetoothDevice
  use arg itemName, state

  if state~equals(.openHAB.state~ON) then
  do
    .openHAB~sendCommand("light_office_ceiling", .openHAB.command~ON)
    .openHAB~sendCommand("light_office_table", .openHAB.command~ON)
    .openHAB~sendCommand("light_corridor_ceiling_front", .openHAB.command~ON)
    .openHAB~sendCommand("light_corridor_ceiling_back", .openHAB.command~ON)
  end
```

file: openHAB/configurations/scripts/RexxDemo.rex

openHAB

- Smart Homes in general on the rise
- 1000+ installations
- large functionality, but mainly for enthusiasts
- openHAB 2.0 to focus on user comfort

rexxHAB

- already wide range of possibilities
- numerous further improvements possible
- tests within physical environment needed
- portation to openHAB 2.0 necessary

THE END

Questions?

Manuel RAFFEL, manuel.raffel@outlook.com

Thank you!

References

Flatscher, R. G. (2013)
Introduction to Rexx and ooRexx

Harper R. (2003)
Inside the Smart Home: Ideas, Possibilities and Methods

innoQ Podcast [Online] (2013)
URL: https://www.innoq.com/de/podcast/002-openhab/transcript (visited on 03/15/2015)

openHAB Wiki [Online] (2015)
URL: https://github.com/openhab/openhab/wiki/ (visited on 03/15/2015)

OSGi Service Platform – Core Specification (2011)
URL: https://osgi.org/download/r4v43/osgi.core-4.3.0.pdf

Raffel, M. (2014)
openHAB – Empowering the Smart Home – History, Concepts, Examples

2

IBM Rexx Language Update:Classic Rexx and The Rexx Compiler – Virgil Hein

Date and Time

30 Mar 2015, 09:00:00 CET

Presenter

Virgil Hein

Presenter Details

Virgil has been with IBM for 38+ years working in software development. In his current position as an IBM Business Manager he is responsible for all facets of a set of mature technology products. This includes responsibility for strategy, business management, development, marketing, sales, service, and support. Main products include Office Vision products, BookManager, Rexx, and OS/2. In this position the main goals are focused on maintaining/increasing customer satisfaction, supporting customer efforts to migrate to follow-on solutions, and finding creative means of increasing mature/growth product revenue. As the product owner for the IBM Rexx Compiler, Virgil is closely involved with a variety of Rexx activities both inside and outside of IBM.

Session Abstract

Virgil keeps us abreast of developments within the IBM teams in charge of IBM's Rexx products.

REXX Language Coding Techniques

Virgil Hein, IBM
vhein@us.ibm.com

Disclaimers

- **The information contained in this presentation is provided for informational purposes only.**
- **While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided "as is", without warranty of any kind, express or implied.**
- **In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice.**
- **IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other documentation.**
- **Nothing contained in this presentation is intended to, or shall have the effect of:**
  - Creating any warranty or representation from IBM (or its affiliates or its or their suppliers and/or licensors); or
  - Altering the terms and conditions of the applicable license agreement governing the use of IBM software.

Agenda

- **REXX products**
- **REXX Enhancements (z/OS)**
- **External environments and interfaces**
- **Key functions and instructions**
- **REXX data stack vs. compound variables**
- **I/O**
- **Troubleshooting**
- **Programming style and techniques**

REXX Session at SHARE

- **I am site editor of Destinationz.org. Destination z is an online mainframe community of IBMers, those in mainframe related jobs, academics and business partners. Looking over your REXX Language Coding presentation you gave at SHARE, I was wondering if you might be interested in contributing an article to Destination z based off your presentation?**

REXX Interpreter and Libraries

- **The Interpreter executes (interprets) REXX code "line by line"**
  - Included in all z/OS and z/VM releases
- **A REXX library is required to execute compiled programs**
  - Compiled REXX is not an LE language
- **Two REXX library choices:**
  - (Runtime) Library – a priced IBM product
  - Alternate library – a free IBM download
    - Uses the native system's REXX interpreter
- **At execution, compiled REXX will use whichever library is available:**
  - (Runtime) Library
  - Alternate Library

The REXX Products

- **IBM Compiler for REXX on zSeries Release 4**
  - z/VM, z/OS: product number 5695-013
- **IBM Library for REXX on zSeries Release 4**
  - z/VM, z/OS: product number 5695-014
- **VSE**
  - Part of operating system
- **IBM Alternate Library for REXX on zSeries Release 4**
  - Included in z/OS base operating system (V1.9 and later)
  - Free download for z/VM (and z/OS)
    http://www.ibm.com/software/awdtools/rexx/rexxzseries/altlibrary.html
- **REXX Interpreter**
  - Included in all z/OS and z/VM releases

Why Use a REXX Compiler?

- **Program performance**
  - Known value propagation
  - Assign constants at compile time
  - Common sub-expression elimination
  - stem.i processing
- **Source code protection**
  - Source code not in deliverables
- **Improved productivity and quality**
  - Syntax checks all code statements
  - Source and cross reference listings
- **Compiler control directives**
  - %include, %page, %copyright, %stub, %sysdate, %systime, %testhalt

REXX Enhancements in z/OS V2.1

REXX Enhancements in z/OS V2.1 and later

- **EXECIO enhanced to support I/O with RECFM=U, VS, VBS**
- **RECFM=U,VS,VBS support also added to callable I/O interface**
- **New TRAPMSG function allows IRX... messages, if issued from a command invoked by the EXEC, to be captured via OUTTRAP**
- **STORAGE function now supports 64-bit addresses for both reading from and writing to storage.**
- **Empty sequential data set can be part of a concatenation accessed by EXECIO, CLIST I/O, PRINTDS if it is SMS managed**
- **LISTDSI enhanced (REXX and CLIST)**
  - RACF/NORACF operand
  - Multi Volume Support
  - Handles data sets with extended attributes
- **Other smaller requirements**

Overview

- Over the years many customers have asked for the capability to handle I/O to data sets containing records with Variable Spanned (VS, VBS) RECFM, and with data sets having undefined (U) RECFM. This includes the ability to handle spanned files generated by SMF, or to read load library type undefined files.
- Problem Statement / Need Addressed
  - Provide the capability to read or write RECFM=VS, VBS, U type data sets under REXX.

    Note: RECFM=VS/VBS files do not support update mode (DISKRU).
- Solution
  - EXECIO support extended
- Benefit / Value
  - The power of REXX and EXECIO can be used to process data sets with RECFM attributes that were formerly not supported.

Usage & Invocation

- There is no change to the execio syntax. Just enhanced capabilities.
- **Example 1. Use EXECIO to read records from an input RECFM=VS file and write them to a new file having RECFM=VBS. (Assumes input LRECL <= 240).**

```
/* REXX */
"ALLOC FI(INVS) DA('userid.test.vs') SHR REUSE"
ALLOCRC = RC
"ALLOC FI(OUTVBS) DA('userid.test.newvbs') SPACE(1) TRACKS " ,
   " LRECL(240) BLKSIZE(80) RECFM(V B S) DSORG(PS) NEW REUSE"
ALLOCRC = MAX(RC,ALLOCRC)
execio_rc = 0                          /* Initialize              */
error = 0                              /* Initialize              */
IF ALLOCRC = 0 THEN
  do
    /*************************************************************/
    /* When spanned records are read, each logical record is the  */
    /* collection of all spanned segments of that record on DASD. */
    /*************************************************************/
    "execio * DISKR INVS (STEM inrec. FINIS"  /* Read all records */
    if rc /= 0 then
      error = 1                         /* Read Error occurred    */
  end
```


Usage & Invocation

- Example 1 continued

```
ELSE
     do
       say 'File allocation error ...'
       error = 1                          /* Error occurred          */
     end
   IF error = 0 then                   /* If no d is ok             */
     DO
       "execio "inrec.0" DISKW OUTVBS (STEM inrec. FINIS"  /* Write all
                                        records read to the new file */
       if rc=0 then
         do
           say 'Output to new VBS file completed successfully'
           say 'Number of records copied ===> ' inrec.0
         end
       else
         do
           say 'Error writing to new VBS file '
           error = 1                      /* Error occurred          */
         end
     END
```


Usage & Invocation

- **Example 2. Use EXECIO to read a member of a RECFM=U file and change the first occurrence of the word 'TSOREXX ' within each record to 'TSOEREXX' before rewriting the record. If a record is not changed, it need not be rewritten.**

```
/* REXX */
/* Alloc my Load Lib data set having RECFM=U BLKSIZE=32000 LRECL=0 */
"ALLOC FI(INOUTDD) DA('apar2.my.load(mymem)') SHR REUSE"
readcnt = 0                          /* Initialize rec read cntr    */
updtcnt = 0                          /* Initialize rec update cntr  */
error = 0                            /* Initialize flag             */
EoF = 0                              /* Initialize flag             */
do while (EoF=0 & error=0)           /* Loop while more recs/no err */
  "execio 1 DISKRU INOUTDD (STEM inrec." /* Read a rec for update  */
  if rc = 0 then                     /* If read ok                  */
    do  /* Replace 1st occurrence of 'TSOREXX' in record by 'TSOEREXX'
                                    and write it back               */
      readcnt = readcnt + 1        /* Records read                  */
      z = POS('TSOREXX ',inrec.1,1)    /* Find target within rec   */
      if z /= 0 then               /* If found, replace it         */
         do
           inrec.1 = SUBSTR(inrec.1,1,z-1)||'TSOEREXX'|| ,
                  SUBSTR(inrec.1,z+LENGTH('TSOEREXX'))  /*Replace it*/
           "execio 1 DISKW INOUTDD (STEM inrec." /* Rewrite the update
                                        made to the last record read*/
```


IBM Software

Usage & Invocation

- **Example 2 continued**

```
         if rc > 0 then                /* If error                  */
                 error=1               /* Indicate error            */
               else
                 updtcnt = updtcnt + 1 /* Incr update count         */
           end
         else                          /* Else nothing changed, nothing
                                          to rewrite                */
           NOP                         /* Continue with next record */
       end
     else                              /* Else non-0 RC             */
       if rc=2 then                    /* if end-of-file            */
         EoF=1                         /* Indicate end-of-file      */
       else
         error=1                       /* Else read error           */
  end                                  /* End do while              */
  "execio 0 DISKW INOUTDD (FINIS"      /* Close the file            */
  if error = 1 then
    say '*** Error occurred while updating file '
  else
    say updtcnt' of 'readcnt' records were changed'
  "FREE FI(INOUTDD)"
  exit 0
```


Overview

- TRAPMSG – a new TSO/E REXX function used in conjunction with OUTTRAP to permit REXX to trap REXX messages (i.e. IRX..... msgs) in some instances. Prior to this, no IRX.... msg could be trapped.
- Problem Statement / Need Addressed
  - REXX IRX..... messages should be trappable via OUTTRAP just as other output (e.g. such as say output from nested execs) is trappable.
- Solution
  - Use TRAPMSG('on') to tell REXX to treat REXX msg output in the same was as any other output, for purposes of trapping.
- Benefit / Value
  - REXX msgs issued by nested execs, and by host commands invoked by REXX (e.g. execio) can now be trapped into an OUTTRAP variable, rather than always being written to screen.
  - CLIST error msgs from CLISTs invoked by REXX also now trappable.

Usage & Invocation

- TRAPMSG() - returns current setting. /* OFF perhaps */
- TRAPMSG('ON' | 'OFF') - enables or disables output trapping for IRX.... msgs. Default is 'OFF'

Usage & Invocation

- Example 1: A REXX exec invokes execio without allocating the indd file. EXECIO will return RC=20 and an error message. By trapping the message with OUTTRAP, the exec can decide what to do with the error.

```
( This same technique can be used to trap the IRX0250E message if
execio were to take an abend, like a space B37 abend.)
   ================================================

   msgtrapstat = TRAPMSG('ON')     /* Save current status and set
                     TRAPMSG ON to allow REXX msgs to be trapped  */
   outtrap_stat = OUTTRAP('line.')   /* Enable outtrap             */
   /*************************************************************/
   /* Invoke TSO host cmd, execio, and trap any error msgs issued  */
   /*************************************************************/
   "execio 1 diskr indd (stem rec. finis"

   if RC = 20 then             /* If execio error  occurred     */
     do i=1 to line.0
       say '==> ' line.i       /* Write any error msgs          */
     end
   outtrap_stat = OUTTRAP('OFF')   /* Disable outtrap           */
   msgtrapstat = TRAPMSG('OFF')    /* Turn it off               */
   exit 0
```


Usage & Invocation

- Example 2: A REXX exec turns on OUTTRAP and TRAPMSG and invokes a second REXX exec. The second REXX exec gets an IRX0040I message due to an invalid function call. Exec1 is able to trap the message issued from exec2.

Note that if exec1 had made the bad function call, it could not trap the error message because a function message is considered at the same level as the exec. This is similar to the fact that an exec can use OUTTRAP to trap SAY statements from an exec that it invokes, but it cannot trap its own SAY output.

```
==================================================
   /* REXX - exec1 */
      trapit = OUTTRAP('line.')
      trapmsg_stat = TRAPMSG('ON')
      call exec2
      do i=1 to line.0   /* Display any output trapped from exec2 */
        say '==> ' line.
      end
      trapit = OUTTRAP('OFF')
      trapmsg_stat = TRAPMSG('OFF')
      exit 0

      /* REXX - exec2 */
      say 'In exec2 ...'
      time = TIME('P')    /* Invalid time operand, get msg IRX0040I*/
      return time
```

