IBM Education Assistance for z/OS V2R2

Item: J-con, RAS, Long section name and LP64 DLL
Element/Component: Binder

Material current as of May 2015

FP0207 J-con

Agenda

- Trademarks
- Presentation Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Presentation Summary
- Appendix

Trademarks

Presentation Objectives

- Binder will fully support J-con in V2R2. This line item is originated from a PMR 66736,49R,000. The corresponding APAR OA42509 is closed as UR1.
- We will discuss
  - Background
  - Usage
  - Effect

Overview

HLASM says “Length constant—J: Use this constant to reserve storage for the length of a DXD, class or DSECT. The assembler fills the field with binary zeros, and the length is entered into this space by the linker(binder). This constant is only available if the GOFF option is specified.”. Prior to V2R2, the binder doesn't support the target of J-con being a DXD (or equivalently DSECT) or PART, and will produce an ABEND (logic error) if given such a GOFF object. (Note that DSECT w/ a J-con for it, just like DSECT w/ a Q-con for it, HLASM makes it be external, so it's just like a DXD.)

- Problem Statement / Need Addressed
  - PMR 66736,49R,000 (APAR OA42509), binder doesn't fully support J-con and its length modifier.
- Solution
  - Binder is improved to support the J-con and J-con length modifier.
  - J-con is only supported in GOFF / PO
- Benefit / Value
  - Binder keeps the consistency with HLASM. And the application (e.g. compilers, translators and debugger) could exploit the benefits and possibilities.
  - Other languages can produce the GOFF to do so, as well as the binder APIs.
  -

Usage & Invocation

- Support J(MYDXD), J(MYDSECT) and J(MYPART). J(MYCLASS) is already supported. Binder will reserve and fill a storage with length value of the J-con target.

```
MYCSECT  CSECT
MYCSECT  AMODE 31
MYCSECT  RMODE ANY
MYDXD    DXD CL33
         DS CL8
MYJCON   DC J(MYDXD)
OTRDXD   DXD CL16
MYCSECT  CSECT
         DS CL76
         DC Q(OTRDXD)
         END
```

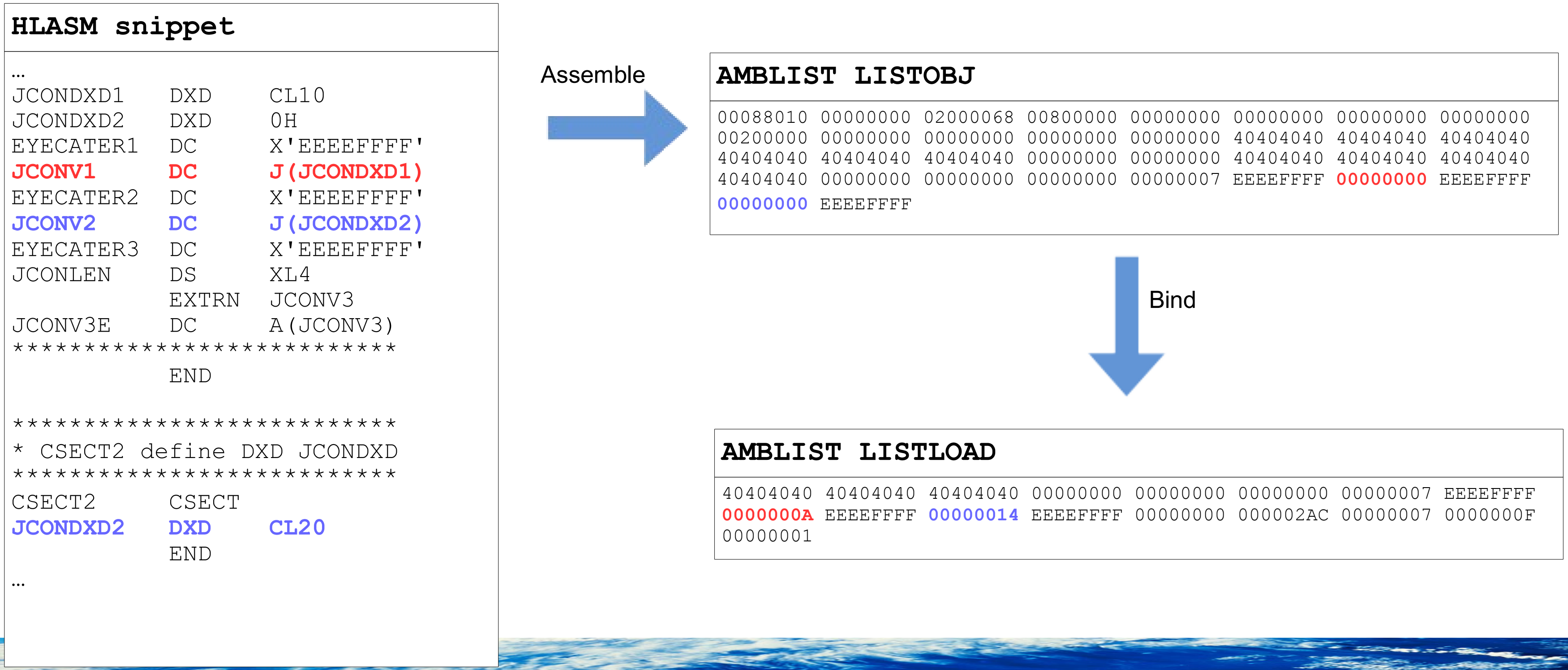
```
MYCSECT  CSECT
MYCSECT  AMODE 31
MYCSECT  RMODE ANY
MYDSECT  DSECT CL33
         DS CL8
MYDXD    DXD CL16
MYCSECT1 CSECT
         DS CL76
         DC Q(MYDXD)
JV       DC J(MYDSECT)
         END
```

```
MYCSECT  CSECT
MYCSECT  AMODE 31
MYCSECT  RMODE ANY
MYCLS    CATTR
PART(MYPART),RMODE(ANY)
         DS CL8
MYJCON   DC J(MYPART)
MYDXD    DXD CL16
MYCSECT2 CSECT
         DS CL76
         DC Q(MYDXD)
         END
```

- Length and Length modifiers
  - Length of J (without length) is 4 bytes
  - Length of JD (without length) is 8 bytes
  - Only 2, 3, 4 or 8 is valid length range for J-cons. e.g. 2 bytes for 'JL2(MYDXD)', 4 bytes for 'JL4(MYDXD)'.

Interactions & Dependencies

- Software Dependencies
  - Requires APAR PI07812 to exploit from High Level Assembler (HLASM was producing an incorrect RLD type).
  - Binder also still did not work with the correct RLD type without OA42509.



HLASM snippet

```
…
JCONDXD1    DXD     CL10
JCONDXD2    DXD     0H
EYECATER1   DC      X'EEEEFFFF'
JCONV1      DC      J(JCONDXD1)
EYECATER2   DC      X'EEEEFFFF'
JCONV2      DC      J(JCONDXD2)
EYECATER3   DC      X'EEEEFFFF'
JCONLEN     DS      XL4
            EXTRN   JCONV3
JCONV3E     DC      A(JCONV3)
***************************
            END

***************************
* CSECT2 define DXD JCONDXD
***************************
CSECT2      CSECT
JCONDXD2    DXD     CL20
            END
…
```

AMBLIST LISTOBJ

```
00088010 00000000 02000068 00800000 00000000 00000000 00000000 00000000
00200000 00000000 00000000 00000000 00000000 40404040 40404040 40404040
40404040 40404040 40404040 00000000 00000000 40404040 40404040 40404040
40404040 00000000 00000000 00000000 00000007 EEEEFFFF 00000000 EEEEFFFF
00000000 EEEEFFFF
```

AMBLIST LISTLOAD

```
40404040 40404040 40404040 00000000 00000000 00000000 00000007 EEEEFFFF
0000000A EEEEFFFF 00000014 EEEEFFFF 00000000 000002AC 00000007 0000000F
00000001
```

Presentation Summary

- Binder fully support J-con feature.
  - Background
  - Usage
  - Effect

Appendix

- HLASM Language Reference - SC26-4940-06
  - Subfield 6: Nominal Value
    - Offset and length constants
      - Length constant—J
- z/OS MVS Program Management: User's Guide and Reference - SA23-1393-00
  - Creating programs from source modules
    - Object and program module structure
      - Relocation dictionary
- z/OS MVS Program Management: Advanced Facilities - SA23-1392-00
  - Generalized object file format (GOFF)
    - GOFF record formats
      - Relocation directory record
        - Relocation directory data element flags field

FP0207 RAS

Agenda

- Trademarks
- Presentation Objectives
- Overview
- Usage & Invocation
- Presentation Summary
- Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.

Presentation Objectives

- Improvements in Binder's RAS capability.
- We will discuss
  - Overview
  - Highlight RAS items
  - MISC – Syntax, Messages

Overview

- Issue for non-executable aliases
- Force true alias with ADDA [to add the ATYPE=T]
- Use input directory information on ADDA
- IEW2646W does not specify CLASS
- TSO LINK and LOADGO don't accept LISTPRIV=INFORM
- _LD_DEBUG_TRACE ld does not support UNIX
- RMODE for load module csects on AMBLIST output
-
- We will highlight the alias related ones and discuss them later

Usage & Invocation

- Issue for non-executable aliases
  - If a user set an alias to an entry point not in the first class of the first segment, the alias would quietly be set to “NOT EXECUTABLE”, and repointed to the offset of the primary entry point (changing it from an alternative entry point to a true alias).
  - Before: binder will keep silence for “NOT EXECUTABLE” alias and re-pointing the alias.
  - After: binder issue a warning message

- Message
  - IEW2619I ALIAS *alias-name* HAS BEEN ASSOCIATED WITH THE PRIMARY ENTRY POINT IN CLASS *class-name1*, BECAUSE EXTERNAL SYMBOL *symbol-name* IS IN CLASS *class-name2*, WHICH IS NOT THE CLASS OF THE PRIMARY ENTRY POINT.
  - IEW2652W ALIAS alias-name HAS BEEN MARKED NON-EXECUTABLE, BECAUSE EXTERNAL SYMBOL symbol-name IS IN CLASS class-name WHICH IS NOT THE CLASS OF THE PRIMARY ENTRY POINT.

Usage & Invocation

- Force true alias with ADDA [to add the ATYPE=T]
  - Currently the binder ADDA API will create an alternate entry point if the alias name matches the name of an LD name in the module. Sometimes this is not what is desired - the user wants this to be a “true alias” which points to the primary entry point. The only way (currently) to cause this to happen is to specify the primary entry point name on ENAME. But if the caller does not know the entry point name it may require several API calls to find it. It would be simpler if the caller could specify ATYPE=T to force a true alias.
- Syntax
  - T: True ALIAS, let the alias name point to the primary entry point in any condition.

<symbol>	IEWBIND	FUNC=ADDA [VERSION=version] [,RETCODE=retcode] [,RSNCODE=rsncode] ,WORKMOD=workmod ,ANAME={alias | pathname} ***[,ATYPE={A | S | P | T}]*** [,ENAME=ename] [,AMODE=amode]

Usage & Invocation

- Use input directory information on ADDA
  - When user copy program with selected aliases, use can INCLUDE module firstly, then use ADDA function call to add specified aliases, but currently binder can not save alias attributes from the directory of an input module for ADDA call.
  - In directory, alias points to entry point offset, if user wants use same offset of alias and other alias attributes from directory when copying the module, this function is needed.
  - To keep alias attributes from directory of an input module, there are following changes:
    - A new ALIASES value “K” (keep) of INCLUDE API is introduced, which will make binder bring alias information in with include module.
    - A new parameter DNAME of ADDA API call is introduced. When specified, it will tell binder the alias source come from directory of included module, binder should find the alias attribute from alias list which brought into binder from directory of input module(by ALIAS=K/Y of INCLUDE). If the name specified for DNAME is not a find in directory, binder will reject ADDA call.
    - Use input directory information on ADDA, when a subset of aliases is desired, or the directory alias names are to be renamed. If it is desired to keep all of them intact then the existing ALIASES=Y would be used.

Usage & Invocation

- Syntax
  - DNAME=d*name* — RX-type address or register (2-12)
    Specifies the location of a varying character string up to 32767 bytes long that
    alias name come from directory of included module.
    If the name specified for DNAME is not a find in directory, binder will issue a error message

<symbol>	IEWBIND	FUNC=ADDA [VERSION=version] [,RETCODE=retcode] [,RSNCODE=rsncode] ,WORKMOD=workmod ,ANAME={alias | pathname} [,ATYPE={A | S | P | T }] [,ENAME=ename] [,AMODE=amode] ***[,DNAME=dname]***

- Message
  - IEW2620E DIRECTORY ALIAS NAME ***aliasname*** IS NOT FOUND FOR ALIAS ***aliasname***

Presentation Summary

- Binder RAS items
  - Overview
  - Highlight RAS items
  - MISC – Syntax, Messages

Appendix

- z/OS MVS Program Management: Advanced Facilities - SA23-1392-00
  - IEWBIND – Binder regular API functions
    - IEWBIND function reference
      - ADDA: Add alias

-

- z/OS MVS Program Management: Advanced Facilities - SA23-1392-00
  - iewbndd.so, iewbnddx.so - Binder C/C++ API DLL functions
    - Binder API functions
      - __iew_addA() - Add alias

OA44452 Long Section Name

Overview

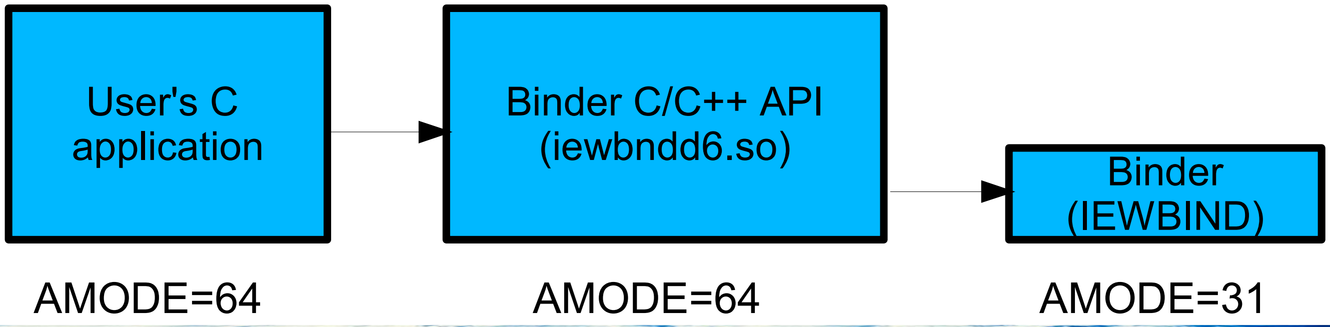
The Binder manual states: “Section names must be 1 to 32767 bytes in length, consisting entirely of EBCDIC characters...... ". At the time when when this was documented, the maximum length of all other external symbol names was increased to 32K-1, however section names still actually could not reliably be longer than 4096 bytes prior to V2R2.

- Problem Statement / Need Addressed
  - OA44452 - binder doesn't support section name up to 8K (actually 32K - 1).
- Solution
  - Previous the APAR OA44452, Binder ABEND if symbol name is larger than 4K
  - After OA44452, binder ABEND if symbol name is larger than 8K
  - Now Binder is improved to support section name up to 32767 in V2R2.
- Benefit / Value
  - The long mangled name from some compilers can be handled. The mangled name could be a function name (only an entry point) or variable name (which is either section names or part names inside of section names with the identical name.)

FP0921 LP64 DLL

Binder C/C++ APIs LP64 DLL

- Requirements
  - Some 31-bit C/C++ applications, which utilize 31-bit Binder C/C++ DLL, are to migrate to 64-bit due to their business requirements. As 31-bit DLLs can not be called by 64-bit C/C++ applications, a new 64-bit Binder C/C++ DLL are required.
- Solutions:
  - A new DLL iewbndd6.so and its sidedeck iewbndd6.x are provided in z/OS UNIX System . In addition, their replications in MVS dataset, SYS1.SIEAMIGE(IEWBNDD6) and SYS1.SIEASID(IEWBNDD6), are also provided.
  - With this new DLL, user's 64-bit C/C++ applications are able to call Binder C/C++ API.


User's C application
Binder C/C++ API (iewbndd6.so)
Binder (IEWBIND)
AMODE=64
AMODE=64
AMODE=31