



# Using the Binder to assemble your application together

[Barry.Lichtenstein@us.ibm.com](mailto:Barry.Lichtenstein@us.ibm.com)

IBM

August 2013  
Session #14104



# Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

- IBM\*
- z/OS\*
- OS/390\*
- Language Environment\*
- S/360
- MVS
- z/Architecture

\* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

\* All other products may be trademarks or registered trademarks of their respective companies.

## Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

# System z Social Media Channels

- **Top Facebook pages related to System z:**
  - [IBM System z](#)
  - [IBM Academic Initiative System z](#)
  - [IBM Master the Mainframe Contest](#)
  - [IBM Destination z](#)
  - [Millennial Mainframer](#)
  - [IBM Smarter Computing](#)
- **Top LinkedIn groups related to System z:**
  - [System z Advocates](#)
  - [SAP on System z](#)
  - [IBM Mainframe- Unofficial Group](#)
  - [IBM System z Events](#)
  - [Mainframe Experts Network](#)
  - [System z Linux](#)
  - [Enterprise Systems](#)
  - [Mainframe Security Gurus](#)
- **Twitter profiles related to System z:**
  - [IBM System z](#)
  - [IBM System z Events](#)
  - [IBM DB2 on System z](#)
  - [Millennial Mainframer](#)
  - [Destination z](#)
  - [IBM Smarter Computing](#)
- **YouTube accounts related to System z:**
  - [IBM System z](#)
  - [Destination z](#)
  - [IBM Smarter Computing](#)
- **Top System z blogs to check out:**
  - [Mainframe Insights](#)
  - [Smarter Computing](#)
  - [Millennial Mainframer](#)
  - [Mainframe & Hybrid Computing](#)
  - [The Mainframe Blog](#)
  - [Mainframe Watch Belgium](#)
  - [Mainframe Update](#)
  - [Enterprise Systems Media Blog](#)
  - [Dancing Dinosaur](#)
  - [DB2 for z/OS](#)
  - [IBM Destination z](#)
  - [DB2utor](#)



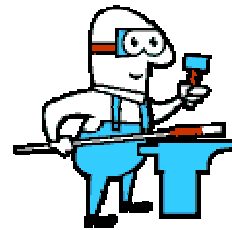
# Agenda

- What is the binder?
- What does the binder do?
- How do I tell the binder to do what it does?
- What else comes with the binder?
- What else can I tell the binder to do?

## What is the binder?

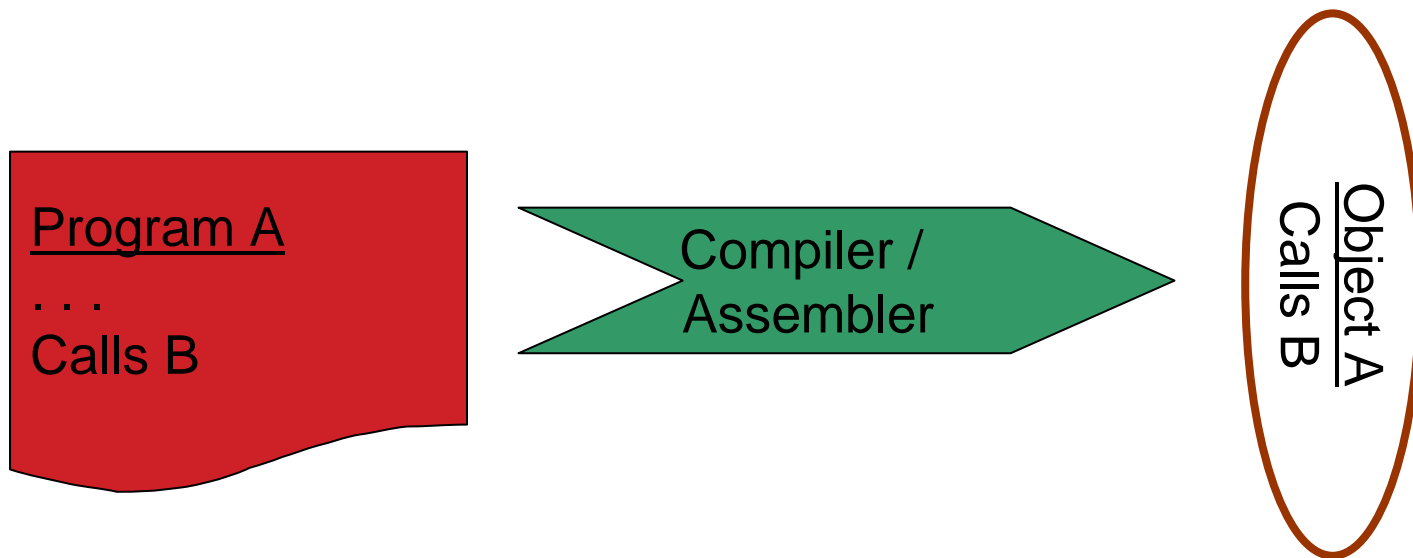
- Wikipedia® under [linker \(computing\)](#):
  - ... a computer program that takes one or more object files generated by a compiler and combines them into a single executable program.
- In IBM mainframe environments such as OS/360 this program is known as a linkage editor.”
- In z/OS the program management binder does this and more!

# linkage editor



- In the old days!

# compiler / assembler



# compiler / assembler object modules

```

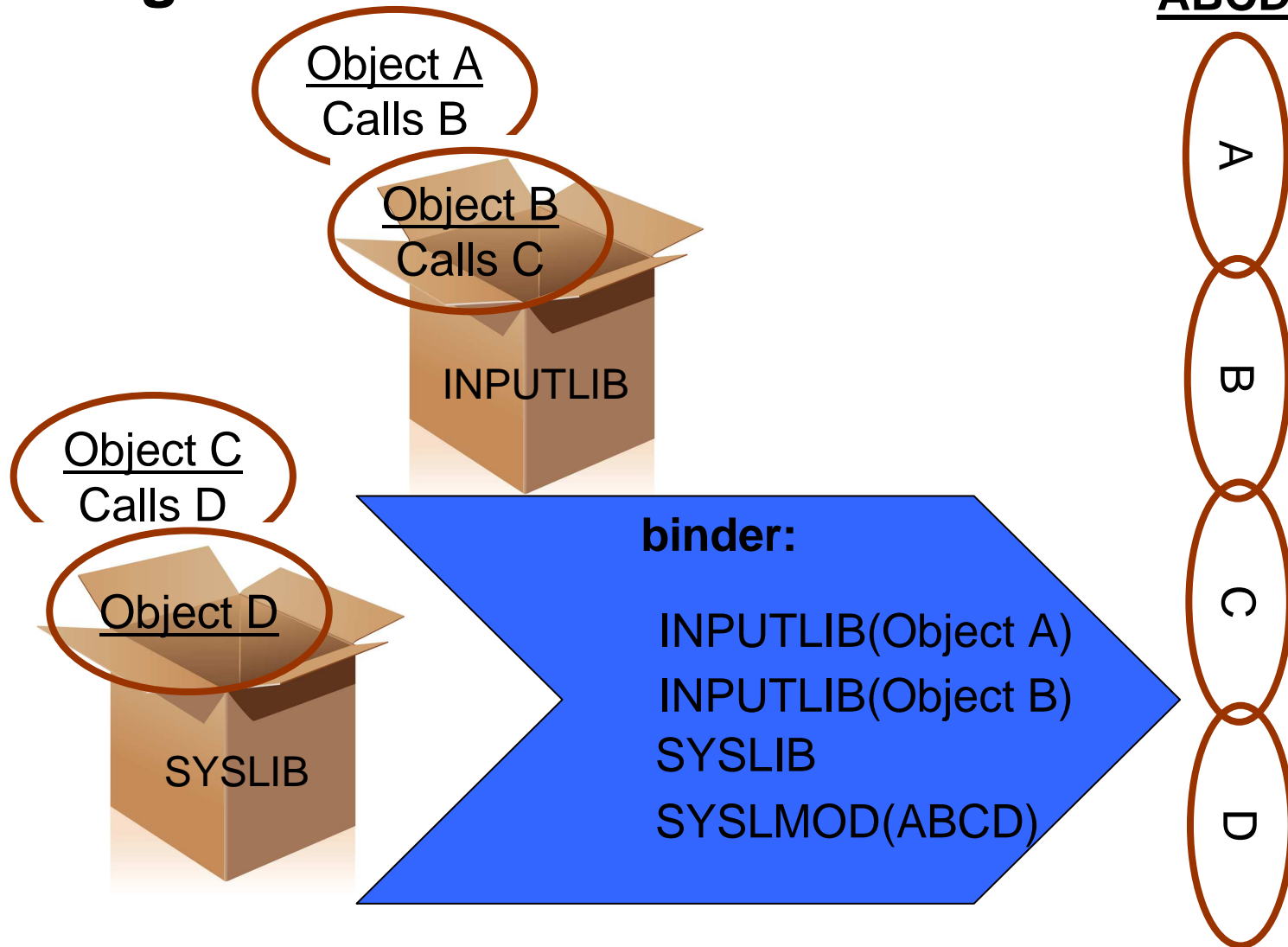
.ESD      ..  ..A      .....B      ....      00000001
.TXT     ...  ..  ..... [X]OBJ Object A - Calls B      00000002
.RLD     ..  .....      00000003
.END     ..  .....      1569623400 010613030      00000004
  
```

```

.0.....
.....B.....
.....Ø.....B_IDRL..
.....Ø.....B_PRV...
.....GOFF Object B - Calls C.....B_TEXT..
.....B.....
.....C.....
.....
.....569623400 01062013030130829500.....
  
```



# linkage editor ...



# executable programs

- Load Module (LM), always in a PDS
  - Records, so this is truly a top-to-bottom view

## Load Module ABCD

```
.Ø.....0@@PPA2 .....C .....çA .....B .....CFUNC
Øx.....
Ø..5695PMB01 ..... "
ØIdØ..5694A01 .....Ø..569623400 ..... Øâ.569623400 ..... ".....
.....M..... ..K.....ç.....Q.....Q.....Q.....Q.....Q.....
.....&.....20130130134316011300.>.....H.
```

\* Chopped off on right and bottom and deleted other lines to make it fit

# executable programs

- Program Object (PO), always in a PDS/E or UNIX file
  - Linear (binary), so bottom-left is maybe the middle

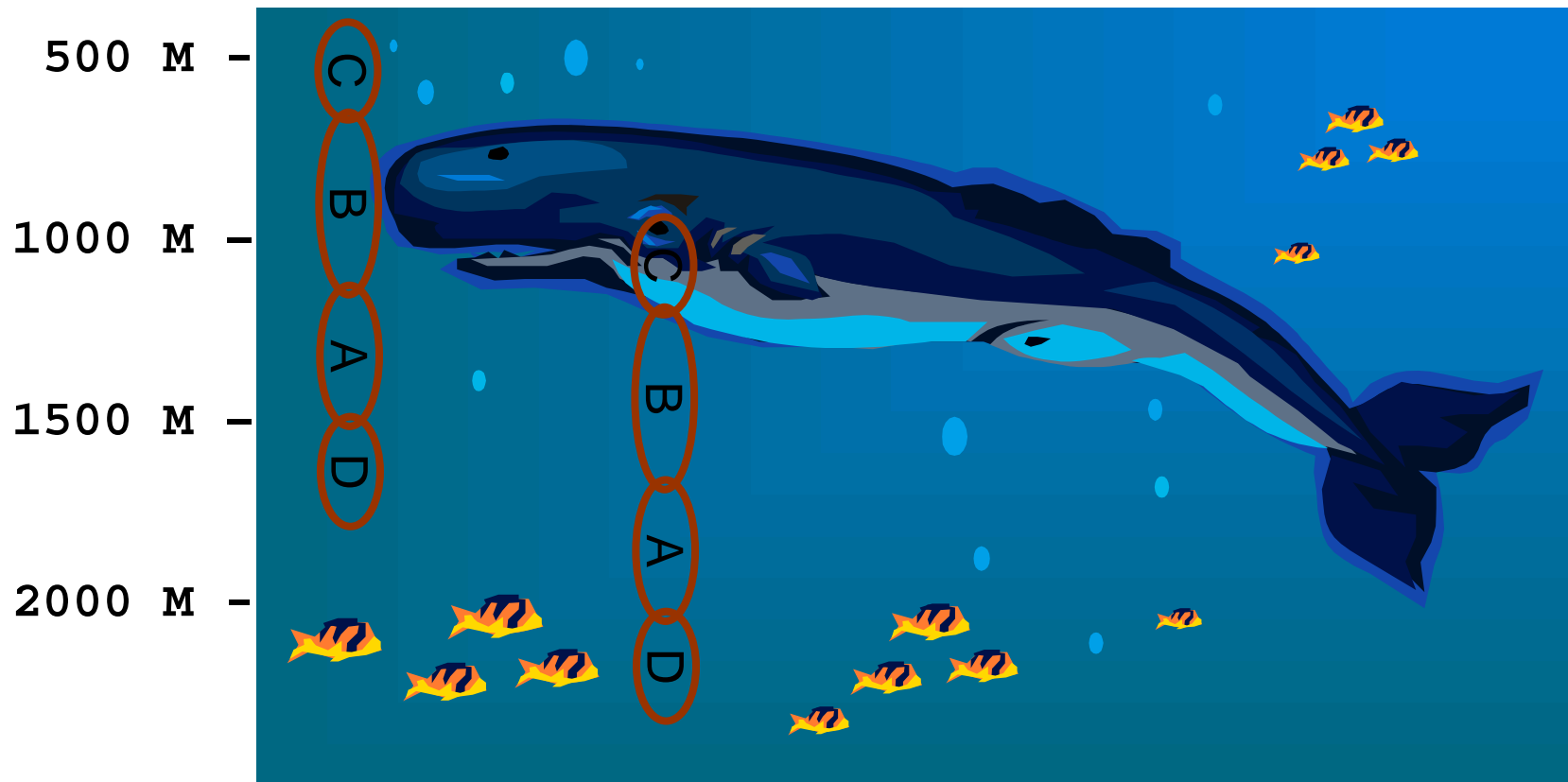
## Program Object ABCD

```

IEWPLMH ...î.....î.....ð.....m...h.....
.....å...å...°O}...â0....².....ÈCEESTART..î0.ª..âu.çâ0.....Ú
.....ø...ø.....ø.....ED .....
L.....ø{.....ø.....ø.....&...&.....PRPRL.....
...SDSD.....ø...ø.....ø.....Q.....8.....ED
.....SDSD.....ø...ø.å...ø.....
.....SDSD.....ø...ø.-...ø.....&...
.....ERWXM...../ø.î...ø.....
  
```

# linkage editor ...

... *and* loader



**MEMORY**

## linkage editor ...

- Symbol resolution
  - all *external* symbol references which need to be satisfied
    - between all input parts
- Relocation
  - all modules combined, relocated relative to origin address
    - zero (or start of segment)
  - final relocation is done by the loader
    - based on information created by the binder

# Program Management Binder

- BCP exclusive base element
  - Wave 0, along with SMP/E and High Level Assembler
- z/OS system linker
  - more than the linkage editor!
- Related utilities
- Programming interfaces

# object file format summary

- Documented
  - Can be produced by non-IBM products
    - Dignus Systems/ASM, Systems/C, Systems/C++ cross-assembler/compiler
- Produced by IBM language translators
  - High Level Assembler (HLASM)
  - Language Environment translators
    - XL C/C++
    - Enterprise COBOL
    - Enterprise PL/I
  - ... and their predecessors
- binder supports 3 flavors
  - OBJ
    - Traditional circa 360 object format
  - XOBJ
    - Initially produced by C/370 for use with the Prelinker
  - GOFF
    - Initially produced by XL C/C++ for XPLink
    - Also produced by High Level Assembler

# better than the linkage editor!



## Load Modules

- Defined almost 50 years ago for S/360™
- Designed specifically for PDS members
  - Depends on hardware keys, format U data
  - Has critical data in directory entries
  - Can only be stored in PDSs
- Significant limitations
  - Symbol names limited to 8 characters
  - 32K maximum external symbols
  - Max size 16M, no split above/below 16M
- Pack maximum data in minimum bytes
  - Great goal, but limits extensibility
- Documented format is exploited by customers
  - Difficult to change

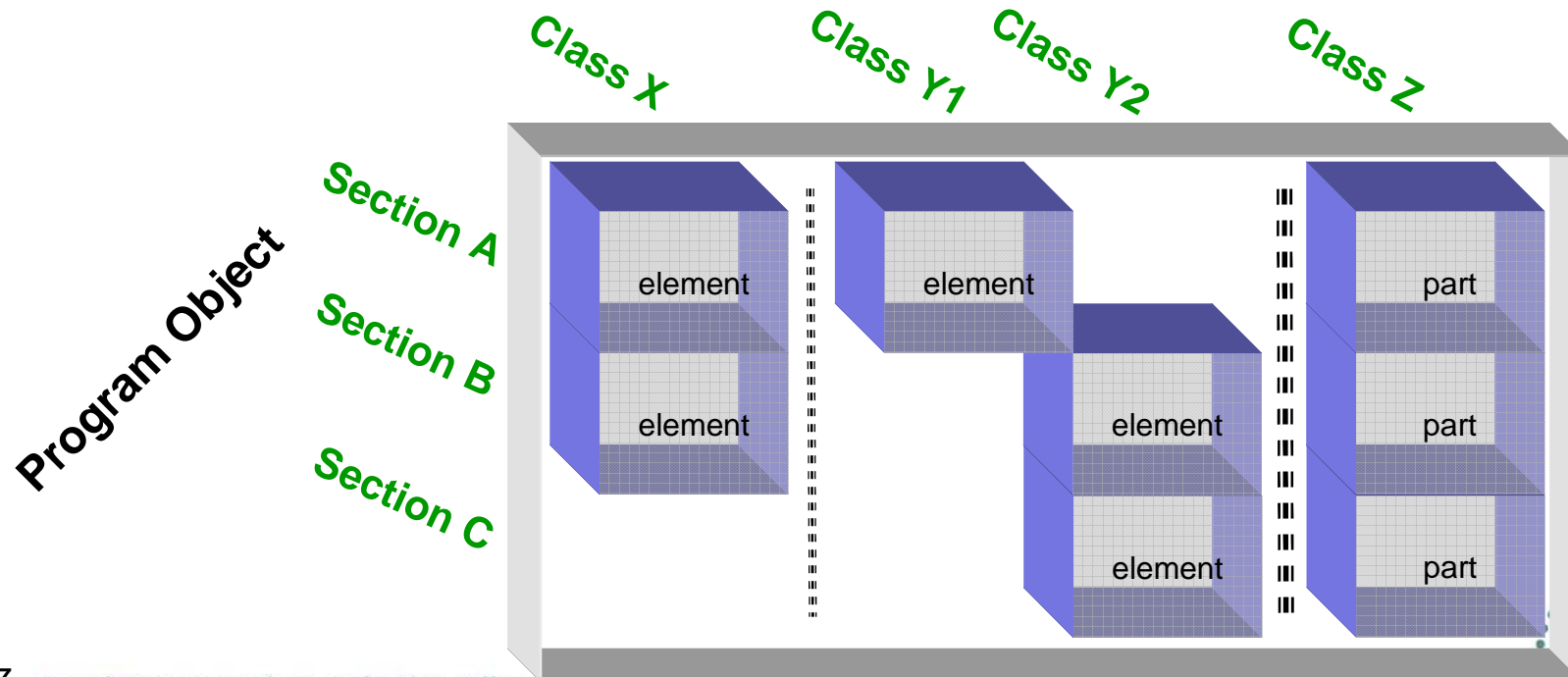
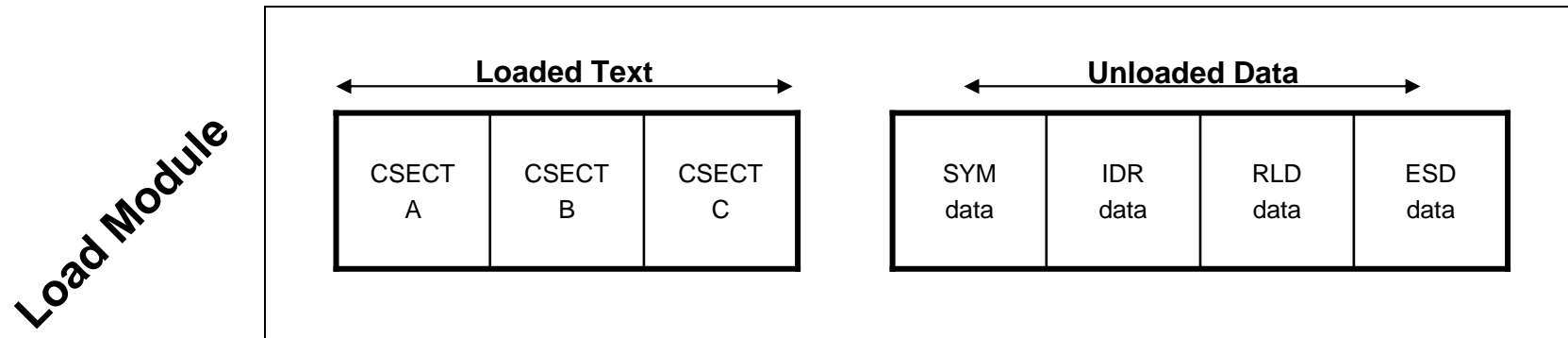
## Program Objects

- Supported by first release of binder
- Designed to be device independent
  - Developed in conjunction with PDSEs
  - Essential for z/OS UNIX support
  - Can only be stored in PDSEs or UNIX files
- Supports symbol names up to 32767 in length and a module length of up to 1 gigabyte
- Designed to support system paging
  - All loadable data is in 4K blocks
  - Loader can treat as extension of page files
- More non-executable data saved
  - Reprocessing is faster and more automatic
  - Supports extra data for debuggers
- Undocumented so allows rapid enhancements
  - 5+ formats to date...





# load module vs. program object



# binder invocation

- **PGM=IEWL** (in JCL)
  - True name
    - **IEWBLINK** (default Link-Edit Utility for **SMP/E**)
  - aliases ala linkage editor names
    - HEWL, HEWLH096
    - HEWLDRGO, HEWLOAD, HEWLOADR
  - aliases of the modern day for binder loader
    - IEWBLDGO, IEWBLODI, IEWBLOAD
    - LOADER
    - IEWLDRGO, IEWLOADI, IEWLOAD, IEWLOADR
  - binder aliases of the modern day
    - IEWL, LINKEDIT
  - alias for customized options
    - IEWBODEF
    - Caution! for sysprogs, rarely used

## but not this, the linkage editor !

- *Invocations of actual linkage editor and batch loader*
  - *HEWLD\* (HEWLD)*
    - *Any remaining invocations of these are batch loader*
  - *IEWL\* (IEWLF880) or HEWL\* (HEWLKED)*
    - *Any remaining invocations of these are linkage editor*
  - *If you use any of these, I'd like to know!!!*
- **NOTE:** *Program Management loader used for PGM=yourpgm*
  - *That is not the Binder!*
  - *It's what is mostly used for program invocation*

## *more binder invocations...*

- The usual suspects:
  - Batch LINKEDIT, IEWL, etc.
- Invoked as a program call:
  - SMP/E (it's *not really* JCL!)
  - TSO LINK, LOAD, LOADGO
  - ld command (UNIX)
- Using the binder Application Programming Interfaces (APIs)
  - c89 (c++), cob2, pli, xlc (xLC)
  - IEBCOPY (sometimes!)
  - SPZAP
  - AMBLIST

# Control Statements

- Your Wish is My Command!
  - Placement
    - Some depend on where they appear relative to others
    - Some depend on where they appear only relative to the same ones
  - Read into the program
  - Change or replace symbol names
  - Change relative locations
  - Specify entry points and their names
  - Specify where to find missing names and find them
  - Write out the program
  - Override options for a single program
- All control statements have analogous API calls

# Control Statements ... Read into the program

- Binder program (not API) starts by reading **SYSLIN**
  - Could be anything!
- **INCLUDE**
  - Explicitly, so always done
- **AUTOCALL**
  - Autocall, so only if it's found
- **IDENTIFY**
  - Not really reading, but associates user identification information to a section which was read in

# Control Statements ...

## Change or replace symbol names

- **CHANGE**
  - Give a symbol definition and references a new name
- **REPLACE**
  - Delete a symbol, optionally give references to it a new name
    - If it's a section, delete the entire section
- **RENAME**
  - Give a renameable symbol a new name
    - Only if there are unresolved symbols
      - *Prelinker compatibility*

# Control Statements ...

## Change relative locations

- **ORDER**
  - Explicitly move a section before everything else
  - Optionally PAGE align it
- **PAGE**
  - Align a section to a 4K (or 2K) page boundary
- **ALIGN** *new!*
  - Align a section, or element or part of it, on a specified boundary
- **EXPAND**
  - Add extra space (set to zeroes) at the end of a section or element



# Control Statements ...

## Specify entry points and their names

- **ALIAS**

- Give another name to call the program by
  - For partitioned datasets these are aliases
    - *Optionally give an entry point symbol name where that program name begins execution*
    - *Or it will default to an entry point name that matches this name, if there is one*
    - *Or the primary name if there is not a matching name*
  - For UNIX files these are either (hard) links or symbolic links
    - *However there is only ever one entry point, the same as the primary name*

- **ENTRY**

- Give an entry point symbol name where the primary name of the program begins execution

- **NAME**

- Give the primary name to call the program by

## Control Statements ...

### Specify where to find missing names and find them

- Binder program (not API) starts by reading **SYSLIB**
  - After all else is done, before preparing to write out program
- **LIBRARY** (autocall)
  - Augments SYSLIB
  - Changes where symbols may or may not be found
- **IMPORT** (DLLs)
  - Tells what DLL an unresolved symbols should be in at run-time

# Control Statements ...

## Write out the program

- Binder program (not API) writes to **SYSLMOD**
- Allocated to either a partitioned dataset or a UNIX pathname
  - May also include the NAME, in lieu of a NAME control statement
- **NAME**
  - Give a name to the program
    - For partitioned dataset, a member name
    - For UNIX, a filename
    - Optionally tells if the an existing program of that name may be replaced

# Control Statements ...

## Override options for a single program

- PARMs are global, these affect only the NAMEd program being bound
  - **MODE** - see AMODE, RMODE options
  - **SETCODE** - see AC option
  - **SETOPT** - generalization for any PARM
  - **SETSSI** - see SSI option
  - **ENTRY** - see EP option

# Options

## Who needs 'em !?

- Binder program (not API) will by default write a SUMMARY LIST to SYSPRINT (which must be allocated) containing:
  - Control statements
  - Most all messages
  - Processing options
  - Summaries of the saved program (if successful)
    - Name (location), type, time
    - Attributes
    - Entry points and aliases
  - Final return code
  - Summary of messages

# Options ...

## Who needs 'em !?

- UNIX command invocations (c89, ld) by default will write to **stderr**:
  - All messages severity 4 (WARNING) and higher
    - That is, no informational messages
  - Use the **-V** option to get most everything written to **stdout**

## Options precedence rules (low to high)

1. Installation options from IEWBODEF
2. Primary invocation options, from one of the following:
  1. The PARM field of the JCL EXEC statement
  2. The first parameter passed to IEWBLINK, IEWBLOAD, etc.
  3. The PARMS parameter of IEWBIND FUNC=STARTD
3. ***The IEWPARMS DD statement – introduced in z/OS V1R11 !***
4. The OPTIONS parameter of IEWBIND FUNC=STARTD
5. IEWBIND\_OPTIONS environment variables via the ENVARS parameter of IEWBIND FUNC=STARTD
6. Dynamic option changes from either:
  1. Options set from attributes by an INCLUDE -ATTR control statement or
  2. The SETOPT control statement, or
  3. The PARMS parameter, followed by the OPTION/OPTVAL parameter, of IEWBIND FUNC=SETO

## OPTIONS option

- **OPTIONS=***ddname*
  - primarily invented to overcome JCL limitations...
    - typically in-stream data set
  - but can be convenient for example to have files of options common to a set of JCL
    - *making it easy to update options without changing JCL etc.*



# Other option sources from UNIX

- makefiles
  - Environment variables which become make macros
    - LDFLAGS
- c89 – YAEV (“yet another environment variable”)
  - `_C89_OPTIONS`
  - `_C89_OPERANDS`
- ld – yikes, just like (you can guess why!)...
  - `_LD_OPTIONS`
  - `_LD_OPERANDS`

# Types of options

- Options for **SYSPRINT**
  - Most common
- Behavior changing options
  - Next most common
- Program changing options
  - Depends on functional requirements

# Options for SYSPRINT

- **LIST, MAP, XREF**
  - SMP/E Link-Editor Utility defaults:
    - LET, LIST, NCAL, XREF
    - NCAL once upon a time was unconditionally set
      - *now based on CALLIBS*
    - If you specify overrides, you must list the others too!
    - SMP/E is picky (it's *not really* JCL)
    - Avoid using control statements to specify options (SMP/E won't know)

## Options for SYSPRINT ...

- SYSPRINT
  - Messages (IEW2nnnns)                      also *SYSTEM*
  - DDname cross-reference
  - Message Summary
  
  - **LIST**ing of processing information
  - Module **MAP**
    - Includes Data Set Summary
  - Cross(**X**) **REF**erence between symbol definitions and references
    - includes DLL IMPORT/EXPORT table

## Options for SYSPRINT ...

- SYSPRINT extras; requires **MAP** or **XREF**
  - **Renamed symbol cross-reference**
    - Usually only for special predefined list of C symbol names
    - Also RENAME control statement
  - **Long symbol abbreviation table**
  - **Short Mangled Name report**
  - **Symbol References Not Associated with any AdCon**
    - “Dangling” External References
    - Also produced with **LIST**
    - Heading may be there even if no symbols
    - Due to external reference ESD entry from object module

# Options for SYSPRINT ... MAP



\*\*\* MODULE MAP \*\*\*

Class name  
and attributes

CLASS C\_CODE

LENGTH = 160 ATTRIBUTES = CAT, LOAD, RMODE=ANY  
OFFSET = 0 IN SEGMENT\_001 ALIGN = DBLWORD

CLASS binding  
attribute

CLASS loading  
behavior

SEGMENT  
containing  
CLASS

SECTION OFFSET	CLASS OFFSET	NAME	TYPE	LENGTH	DDNAME	SOURCE SEQ MEMBER
0	0	CEESTART	CSECT	7C	/0000001	01
0	0	CEESTART	LABEL			
0	80	this_is_a-g_name	CSECT	E0	/0000001	01
0	80	this_is_a-g_name	LABEL			
28	A8	main	LABEL			

Offset of LABEL main  
within section (CSECT)  
this\_is\_a-g\_name

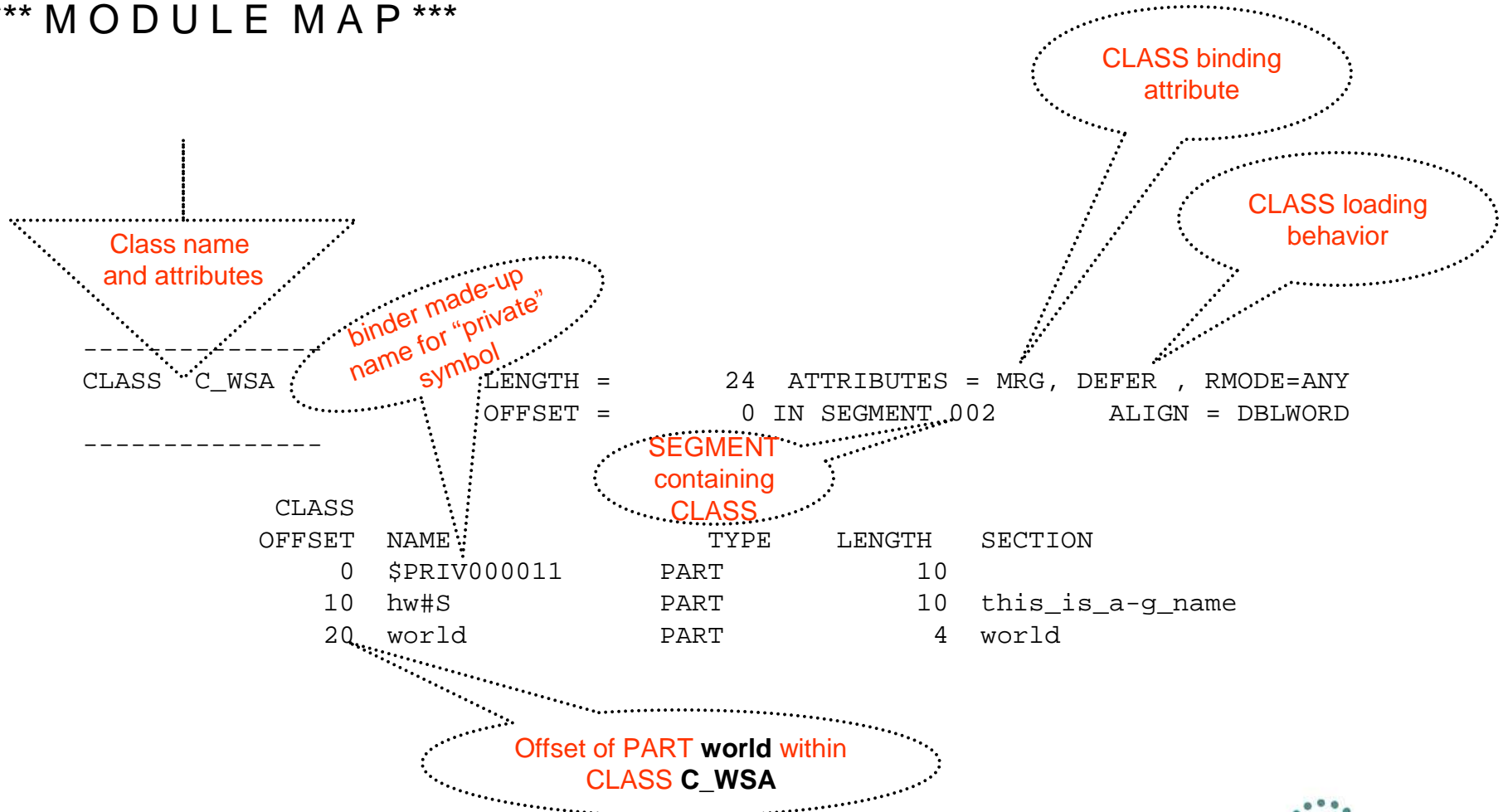
Offset of LABEL main within  
CLASS C\_CODE



# Options for SYSPRINT ... MAP ...



\*\*\* MODULE MAP \*\*\*



# Options for SYSPRINT ... XREF



## CROSS-REFERENCE TABLE

TEXT CLASS = C\_CODE

R E F E R E N C E			T A R G E T				
CLASS	OFFSET	SECT/PART (ABBREV)	ELEMENT OFFSET	TYPE	SYMBOL (ABBREV)	SECTION (ABBREV)	ELEMENT OFFSET CLASS NAME
	2C	CEESTART	2C	V-CON	CEEMAIN	CEEMAIN	0 C_DATA
	68	CEESTART	68	V-CON	CEEFMAIN	<b>\$UNRESOLVED</b>	
	6C	CEESTART	6C	V-CON	CEEBLLST	CEEBLLST	0 B_TEXT
	74	CEESTART	74	V-CON	CEEBETBL	CEEBETBL	0 B_TEXT
	78	CEESTART	78	V-CON	CEEROOTD	CEERootA	0 B_TEXT
	14C	this_is_a-g_name	CC	A-CON	CEESTART	CEESTART	0 C_CODE

**All address constants in section CEESTART in CLASS C\_CODE**

**Location to which adcons in section CEESTART have resolved**

**We can see that section CEESTART begins CLASS C\_CODE**





# Options for SYSPRINT ... XREF ...



## CROSS-REFERENCE TABLE

Symbol world is a part... we know from the Module MAP...

Adcon at X'1C' in section hw#S refers to IMPORTED symbol printf. Location of printf not known until run-time.

```
TEXT CLASS = C_WSA
```

R E F E R E N C E			T A R G E T					
CLASS	OFFSET	SECT/PART(ABBREV)	ELEMENT OFFSET	TYPE	SYMBOL(ABBREV)	SECTION (ABBREV)	ELEMENT OFFSET	CLASS NAME
	10	hw#S	10	A-CON	world	\$PRIV000003	20	C_WSA
	18	hw#S	18	R-CON	printf			
	1C	hw#S	1C	V-CON	printf	<b>\$IMPORTED</b>		
	20	world	20	A-CON	this_is_a-g_name	this_is_a-g_name	0	C_CODE
	18	hw#S	18	A-CON				B_IMPEXP
	1C	hw#S	1C	V-CON	CEETHLOC	CEETLOCE	8	B_TEXT



## Options for SYSPRINT ...

- **INFO** about service level of binder
- **MSGLEVEL** of lowest severity messages to write
  - Default is all (0)
  - Suppresses text, no change to return code!
- **LISTPRIV** for a listing of “private code” sections
  - and if so make it an error (YES)
  - or just informational (INFORM)
- **SYMTRACE** *new!*
  - Messages for all instances of a named symbol during processing

# Behavior changing options

## LET my program be executable

- **LET=number**
  - “LET this be an executable, even if the return code is equal to or less than number”
  - EXECUTABLE is an attribute in the program and in the case of datasets, in the directory
    - NX in ISPF member list means “Not Executable”
    - Nothing to do with the UNIX execute permission
  - “LET” in batch means LET=8
    - Unspecified or “NOLET” means LET=4

# Behavior changing options ...

## Save a non-executable program

- **STORENX**

- STORENX controls whether the “Not Executable” program is saved
  - The default is NOREPLACE (same as NO)...
  - That means by default, a “Not Executable” program WILL BE SAVED if it does not already exist!
  - STORENX=NEVER
    - *Did not always exist, so not the default*

# Behavior changing options ...

## Execute an non-executable program

- What happens if I try to execute an NX program?
  - from batch

```
CSV016I REQUESTED MODULE STOREDNX IS NOT EXECUTABLE
CSV028I ABEND706-04 JOBNAME=BARYLR STEPNAME=GO
IEA995I SYMPTOM DUMP OUTPUT 467
SYSTEM COMPLETION CODE=706 REASON CODE=00000004
```

- from UNIX... usually you will see...

```
BARYLR [478] /u/barryl/binder/SHARE/SHARE116 $ ./a.out
IEWPLMH: ./a.out 14: FSUM7351 not found
```

- ...shell semantics for a failed spawn, to treat as a shell script
- as a DLL

```
CEE3512S An HFS load of module SNX.dll failed. The system return code was 000000130; the reason code was 053B006C.
From entry point main at compile unit offset +000000A8 at entry offset +000000A8 at address 20F1AA10.
```

## Behavior changing options ...

- **CASE**
  - Applies to option values, control statements and API parameters
  - **UPPER** – Default is to uppercase
  - **MIXED** – Preserve the input as-is
    - c89 default

# Program changing options

- **COMPAT**

- The “compatibility” level of the program
- Specified as z/OS releases
  - Or CURRENT
  - Or (older convention) as PM levels
- Each COMPAT release means the program can be fully functional on that release and above
  - May execute on prior releases but other things may not work
    - *Like rebind, IEBCOPY, AMBLIST...*

## Program changing options ...

- **STRIPSEC/STRIPCL** to remove and list “unneeded” stuff
  - To see the “removed” report requires **MAP** option
  - **STRIPSEC=YES**
    - remove unneeded stuff
  - **STRIPSEC=PRIV** *new in z/OS V1R13 !*
    - just unneeded “private” stuff
  - **STRIPCL=YES**
    - Remove class marked as “removable”



# Program changing options ...

- **COMPRESS=YES (default is AUTO)**
  - Can significantly shrink size of program object on disk
  - **No Change** to size of in-storage program!
    - No Change to the program itself (loader / run-time data), only binder owned data
  - Distinguished in **Save Module Attributes (LIST output)**:

```

MODULE SIZE (HEX)      00002BFC
DASD SIZE (HEX)       0000D000 (this had been 00015000)
  
```

- Requires COMPAT(ZOSV1R7)

```

PROGRAM TYPE          PROGRAM OBJECT(FORMAT 4 OS COMPAT LEVEL
z/OS V1R7 )
  
```

- AUTOMatically happens, if beneficial, with this or later COMPAT level
  - *default is COMPAT(MIN)*
  - *will still execute back to ZOSV1R3*
    - *but no rebind, AMBLIST, ZAP, etc.*

## Program changing options ...

- **EDIT=NO**
  - *Permanently deletes* the data that COMPRESS would have compressed
  - Thus *limited* rebind, AMBLIST, ZAP, etc. *anywhere*

```
MODULE SIZE (HEX)    00002BFC
DASD SIZE (HEX)     00005000
```

- Limitation is binder based so:
  - *AMBLIST of LM works because it doesn't use binder*
  - *Binder supports limited processing of INTENT=ACCESS LM*

# Program changing options ...

- **FILL=xx**
  - All uninitialized areas (but not EXPANDED areas) will be set to this value
    - Some of the areas may be written to disk
    - Some “gaps” will only be “filled” when they are loaded
  - Program Object COMPAT=PM2 or later only! Else RC=4...
    - IEW2695W 4B37 OPTION SPECIFICATION FOR FILL IS NOT VALID FOR VERSION 1 PROGRAM OBJECT OR LOAD MODULE.
  - Intended as debugging aid (not to overcome poor programming!)
    - Also see Language Environment STORAGE options

## Program changing options ...

- **DYNAM=DLL** – Dynamic Link Library
  - exported symbols to SYSDEFSD as IMPORT control statements
  - Control information (visible in **MAP** and AMBLIST output, macros in ‘SYS1.MACLIB’)
    - IEWBLIT section B\_LIT class – Loader Information Table
    - IEWBCIE section B\_IMPEXP class – Import/Export table
- Language Environment high-level languages and High Level Assembler (LE provides macro)
- Execution requires Language Environment run-time support
  - Function “descriptors” enable dynamic linking
- Exploits deferred load C\_WSA[64] class
  - Writable / Static Area
  - LE controls unique instance for each “enclave” of execution
- Dynamic resolution follows all static resolution

## Program changing options ...

- **SIGN=YES** – Program Signing – *new in z/OS V1R11 !*
  - Digital signature is written into program object
    - Constructed based on program data
    - Becomes part of program
    - PDSEs supported only!
  - Requires SAF/RACF setup & services
    - Require keyring or PKCS #11 token to sign
    - Program must be identified as requiring digital signature for execution
      - ... *loader verifies correct digital signature prior to execution*
  - Cannot use traditional (SMP/E) service methodology since only signer can bind
    - Could use EDIT=NO

# Option-less output

- Written to if exists
- **IEWDIAG**
  - All messages, as if MSGLEVEL=0 and LIST=ALL
  - Useful when options cannot be passed (particularly API users)
- **IEWTRACE**
  - IBM service aid, shows key trace points throughout processing
  - TRACE option can limit range (default is ALL)
- **IEWDUMP**
  - IBM service aid, SNAP dump and binder formatted dump
  - Automatic on terminal (level 16) error
  - DUMP option can activate for specific ECODE (binder message or trace point)

## So what comes with the binder?

- Batch binder
- Batch binder loader
  
- Legacy batch linkage editor
- Legacy batch loader
  
- TSO invocations of the above
- UNIX **ld** command to invoke batch binder

# What else comes with the binder?

## Service aids

- **AMASPZAP** (Superzap)
  - Service aid to modify existing program objects
    - binders owns PO support, BCP service aids owns the LM
    - Can modify program text, but not change size, offsets, etc.
- **AMBLIST**
  - Service aid to list the contents of OBJ, GOFF, LM and PO
    - Fully deconstruct
    - PMAR, data and IDRs for programs
    - Segment map for POs
  - **amblis**t UNIX command



# What else comes with the binder?

## Binder APIs

- copy
  - IEBCOPY
  - cp, mv
- bind
  - write your own binder!
    - could have a direct-to-program compiler
    - c89 uses binder APIs
    - ld calls batch binder program

# What else comes with the binder?

## Binder APIs ...

- edit without rebinding
  - superZAP (change text so long as length is same)
  - change AMODE, RMODE, entry point, reusability attributes
  - add or delete aliases or IDRUs
- extract data
  - AMBLIST
  - Debuggers
  - Performance analyzers
  - nm
- regular APIs support both executable modules formats
  - So need not code separately (PO vs. LM)

# What else comes with the binder?

## Binder APIs ...

- 1 - Regular (original)
  - Establish dialog with binder (IEWBIND) and create one or more workmods under dialog
  - APIs have a version number indicative of parameter list and functionality
    - Default is Version 1 – don't use it!
  - Binder converts all executables into an internal format called ***workmod***

# What else comes with the binder?

## Binder APIs ...

- 2 - Fast Data Access
  - Only for Program Objects (Load Module format documented)
  - No **workmod** is created thus processing is streamlined
  - Read-Only access (cannot make ANY modifications!)
  - There are two interfaces
    - Request code interface
      - *Introduced in z/OS V1R5*
      - *Simplified parameter list*
      - *More dialog-like (as 'regular' API)*
      - *More functionality*
      - *As of z/OS V1R9 it is completely rewritten and internally an AMODE=64 program*
    - Unitary interface (original)
      - *Macro (IEWBFDA) provided for access and to simplify coding parameters*
      - *Limited functionality (comparable to GD request code only)*
      - *Functionally stabilized*

# What else comes with the binder?

## Binder APIs ...

- 3 – C/C++ DLLs
  - Not really a different flavor!
  - Simplified C interfaces to both regular APIs and fast data access APIs
  - Simplifies management of binder (loading modules, creating buffers)
    - oriented to buffer data (records) returned
  - Provides extra utility interfaces
    - Create lists needed by some API calls
    - Test for end-of-data on get calls
    - Get Return/Reason codes (new APIs)
    - Get/Set cursor
  - Uses *contexts* – for regular APIs this represents workmod+dialog (no facility for multiple workmods in a single dialog)

# What else comes with the binder?

## Binder APIs ...

- 3 – C/C++ DLLs ...
  - APIs in Dynamic Link Library (DLL)
    - **iewbndd.so**
    - **iewbnddx.so** — *XPLINK new in z/OS V1R12*
  - C/C++ header file provides buffer structures, API prototypes and other needed data types – **\_\_iew\_api.h**
  - Side file links with application to access DLL
    - **iewbndd.x**
    - **iewbnddx.x** — *XPLINK new in z/OS V1R12*
  - Installs into UNIX file system (/usr/lib, header in /usr/include)
  - Installs into datasets (SYS1.SIEAMIGE and SYS1.SIEASID)

*new!*

# What else comes with the binder?

## Binder APIs ...

- Module data is returned in a buffer provided by the API caller
- IEWBUFF macro can help (but is not required)
- Same buffer format used by both regular APIs and fast data APIs
- Buffers have version numbers indicative of buffer format
  - **Until z/OS V1.10** regular APIs required matching version numbers
  - Version numbers are ubiquitous
- The buffer ID must be consistent with the type of data being requested
  - For example, the buffer ID for ESDs is IEWBESD

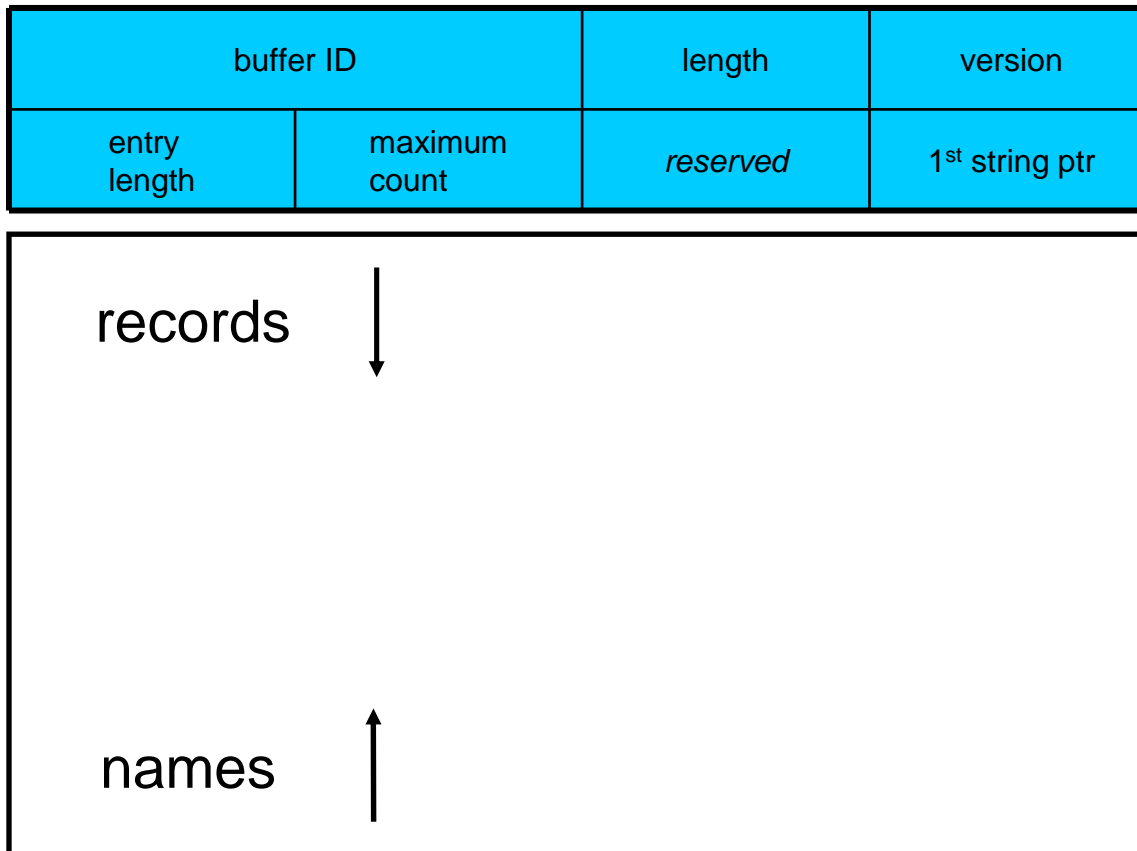
# What else comes with the binder?

## Binder APIs ...

- Earlier buffer versions may not contain all information available from later PO formats
  - APIs will attempt to convert data to a format compatible with the buffer version
  - In some cases the conversion cannot be performed and the request will fail.
  - The most likely scenario in which this would happen is using a version 1 ESD buffer to retrieve information from PO format PO2 or greater with multiple text classes
    - *The differences between later PO versions are much smaller*



# What else comes with the binder? Binder APIs ...



# What else comes with the binder?

## Binder APIs ...

- IEWBUFF usage
  - Must specify BUFFER TYPE
    - ESD, RLD, NAME, TEXT etc.
  - Must specify FUNCTION
    - *MAPBUF* - generate buffer mapping for selected buffer type
    - *GETBUF* - acquire storage for buffer
    - *INITBUF* - initialize buffer header
    - *FREEBUF* - release storage acquired via *GETBUF*
  - MAPBUF must be used first since it specifies the buffer size used by GETBUF and values to be inserted in the buffer header.
    - *Buffer size can be specified as SIZE (record count) or BYTES*
    - *Should specify version number (VERSION). Default is version 1 - probably NOT what you want*

# What else comes with the binder?

## Binder APIs ...

- Class name are limited to 16 bytes
- Other ESD names are limited to 32K-1 bytes
- Binder generated names, demangle named and abbreviated names as they appear in the printed output are not how they look in the program
  - You *must* use the *real internal name* in the API
  - C/C++ APIs work with strings representing binder generated names
    - `__iew_api_name_to_str`
- Binder-generated names for sections and symbols are 4-byte binary numbers
  - Printed as `$PRIVxxxxxx`, where `xxxxxx` is the hexadecimal representation of the binary number
- C++ mangled names are used directly as is
  - no demangling provided by APIs

## binder documentation

- SA22-7643 - z/OS MVS Program Management:  
User's Guide and Reference

options & control  
statements

- SA22-7644 - z/OS MVS Program Management:  
Advanced Facilities

binder **APIs**

- GA22-7589 - z/OS MVS Diagnosis:  
Tools and Service Aids

**AMBLIST** and **SPZAP**

- SA22-7782 - z/OS TSO/E Command Reference

**LINK** and **LOADGO**

- SA22-7802 - z/OS UNIX System Services  
Command Reference

**c89** and **ld**



# Using the Binder to assemble your application together

[Barry.Lichtenstein@us.ibm.com](mailto:Barry.Lichtenstein@us.ibm.com)

IBM

August 2013

Session #14104

