

Binder for Beginners

Everything a z-OS Programmer Ever Wanted to Know about Binding but Was Afraid to Ask

Session 7692

Barry Lichtenstein
IBM Poughkeepsie
Barry.Lichtenstein@us.ibm.com



SHARE in Boston – August 2010
Copyright International Business Machines Corporation 2010

SHARE in Boston

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.

What is the binder?

- Wikipedia® under [linker \(computing\)](#): “In IBM mainframe environments such as OS/360 this program is known as a linkage editor.”
- In z/OS it is the program management binder

program management binder

- BCP exclusive base element
 - Wave 0
- z/OS system linker
- Related utilities
- Programming interfaces

program management binder ...

- The binder converts the output of language translators and compilers into an executable program unit ...
 - ... that can either be read directly into virtual storage for execution or stored
- The binder processes object modules, load modules and program objects...
 - *link-editing* or *binding* multiple modules into a single load module or program object
 - ... with contiguous virtual storage addresses

program management binder ...

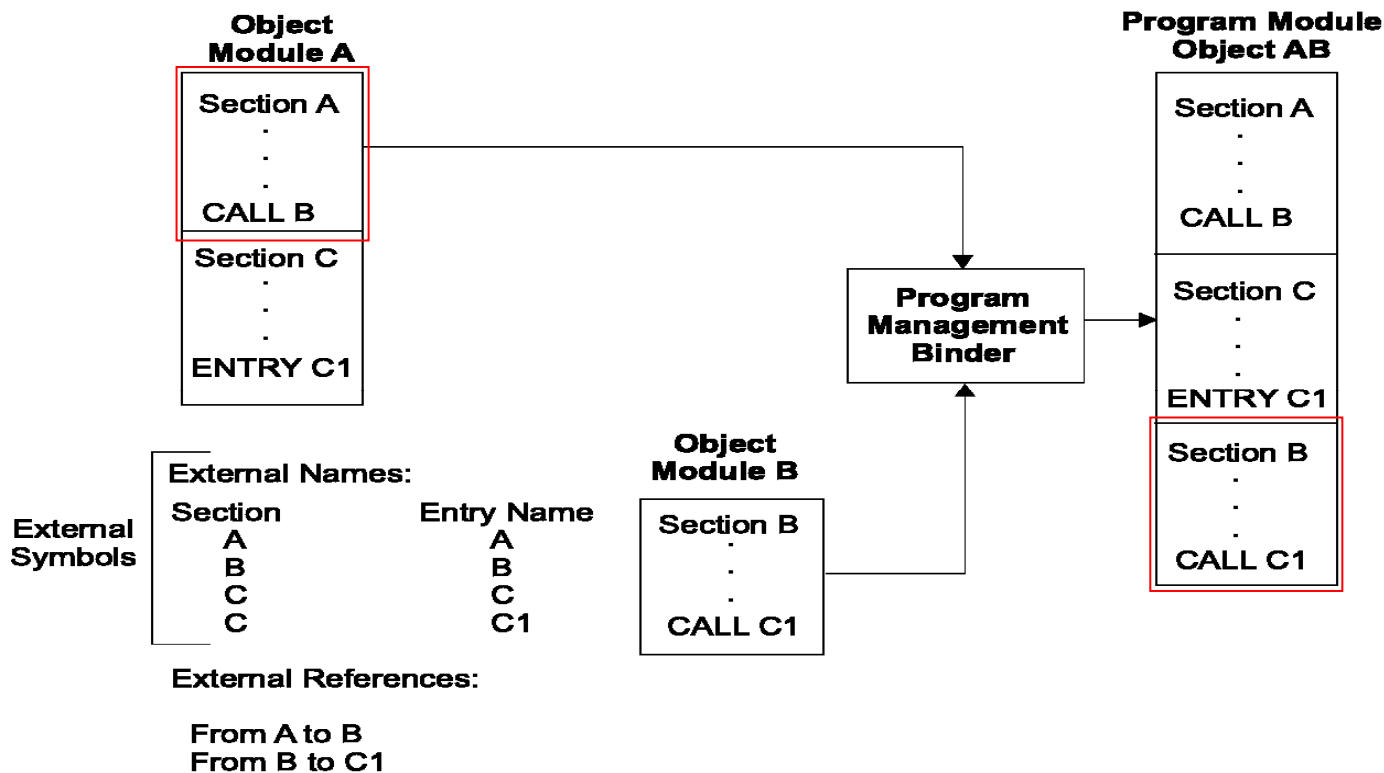
- Symbol resolution
 - all *external* symbol references which need to be satisfied
 - between all input parts
- Relocation
 - all modules combined, relocated relative to origin address
 - Zero (or start of segment)
 - final relocation is done by the loader
 - based on information created by the binder

program management binder ...



symbol resolution

- In Section A there is a call (reference) to B which will be statically linked to A
- Location of B relative to the call in Section A is determined at bind time
- Final relocation of entire executable program module determined at load time

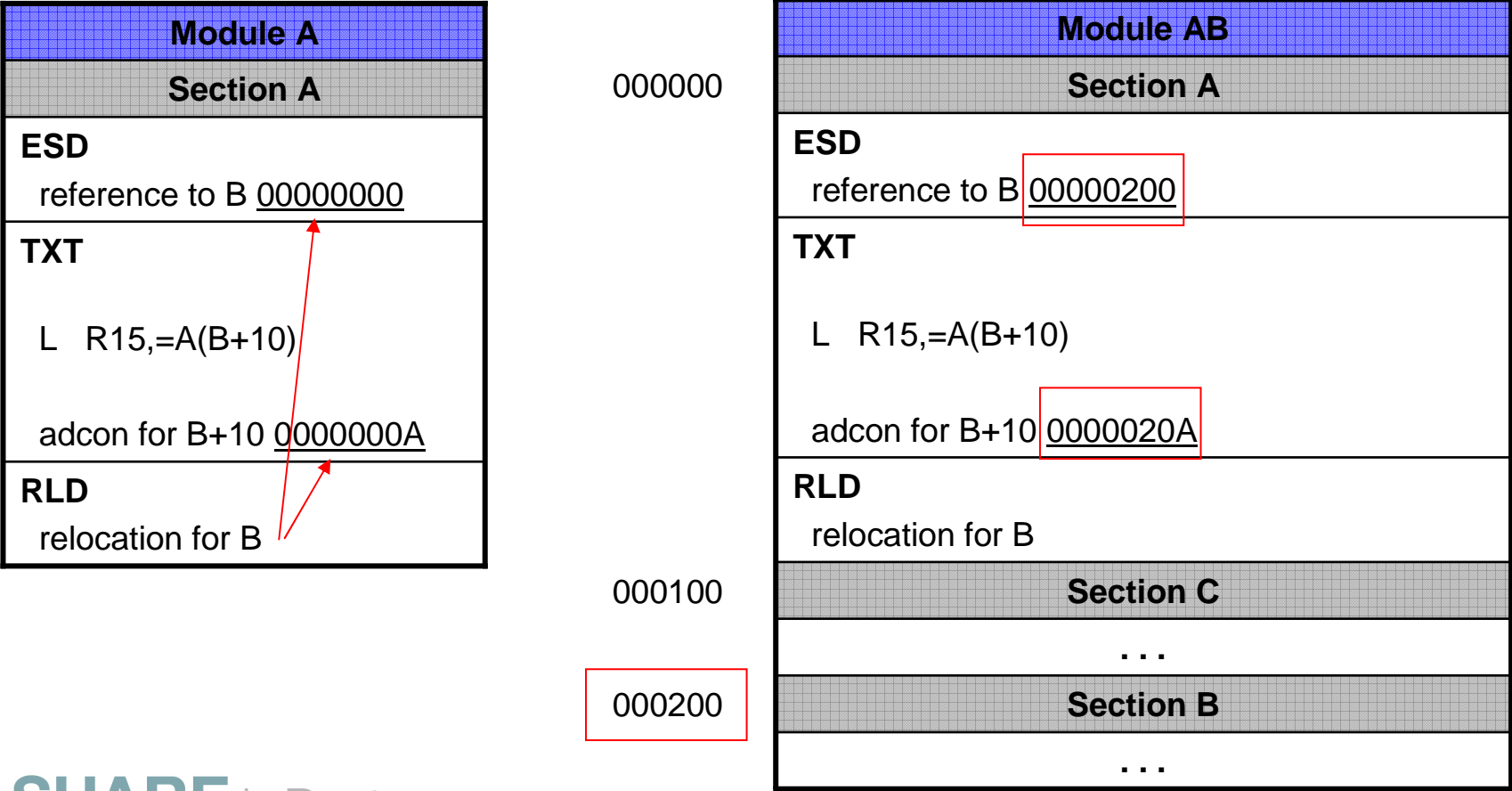


program management binder ...



relocation

- There is an External Symbol Dictionary (ESD) entry for the location of B
- There is an Relocation Dictionary (RLD) entry for the location in A to write the location of B
- What if B is unresolved ?



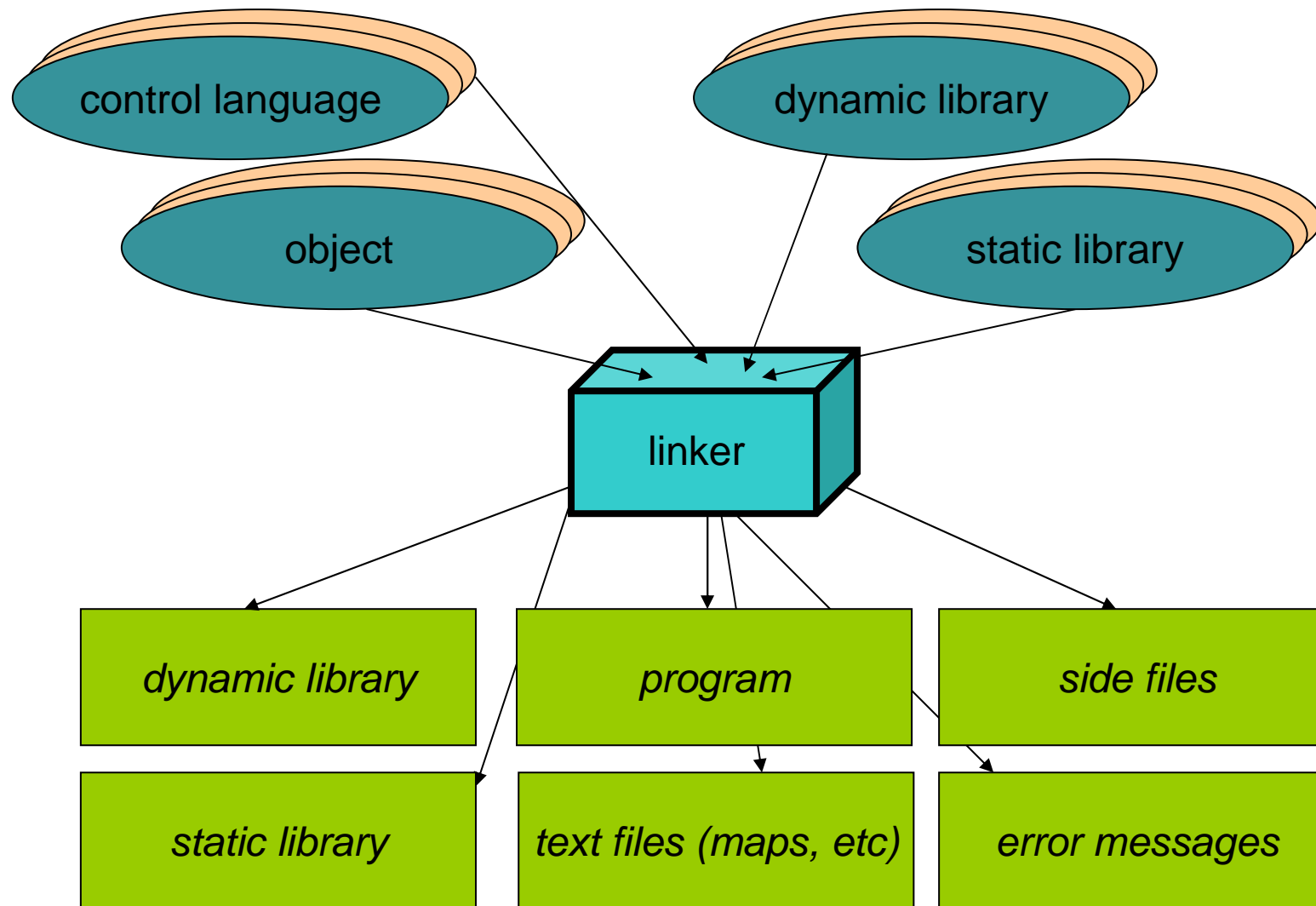
z/OS system linker

- Really not the linkage editor!
 - Application programming interface
 - ...
- PDSE, UNIX
 - program object format – PO (COMPAT(PMx))
 - exclusively binder
 - loaded by program management loader
- PDS
 - load module format
 - just like the linkage editor used to do
 - HEWLKED anybody?
 - loaded by program management loader (program fetch)

z/OS system linker...

- Really, *the* one and only!
 - Batch LINKEDIT, IEWL, etc.
 - TSO LINK, LOAD, LOADGO
 - c89 (c++), cob2, pli, xlc (xLC)
- IEBCOPY
- ZAP
- AMBLIST
- batch loader (HEWLD[I[A]])
 - binder does it too
 - *IEWBLOAD/IEWBLODI/IEWBLDGO*
 - *single segment only (more later!)*

z/OS system linker ...



binder inputs

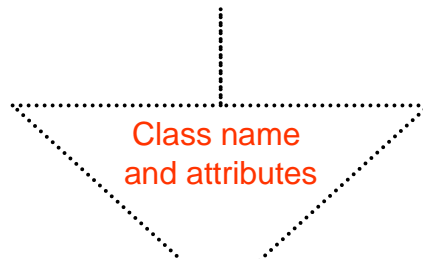
- SYSLIN
 - object modules
 - OBJ, XOBJ, GOFF
 - program modules (executables)
 - load modules
 - program objects
 - control statements

binder outputs

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

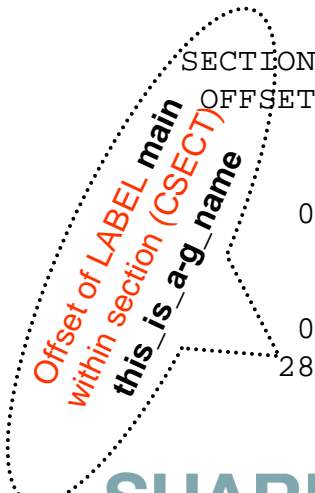
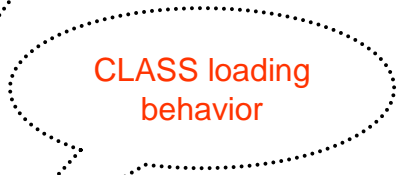
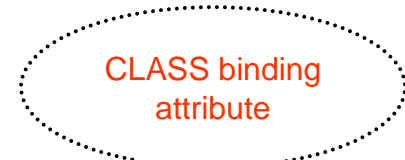
MAP

*** MODULE MAP ***



CLASS C_CODE

LENGTH = 160 ATTRIBUTES = CAT, LOAD, RMODE=ANY
 OFFSET = 0 IN SEGMENT_001 ALIGN = DBLWORD

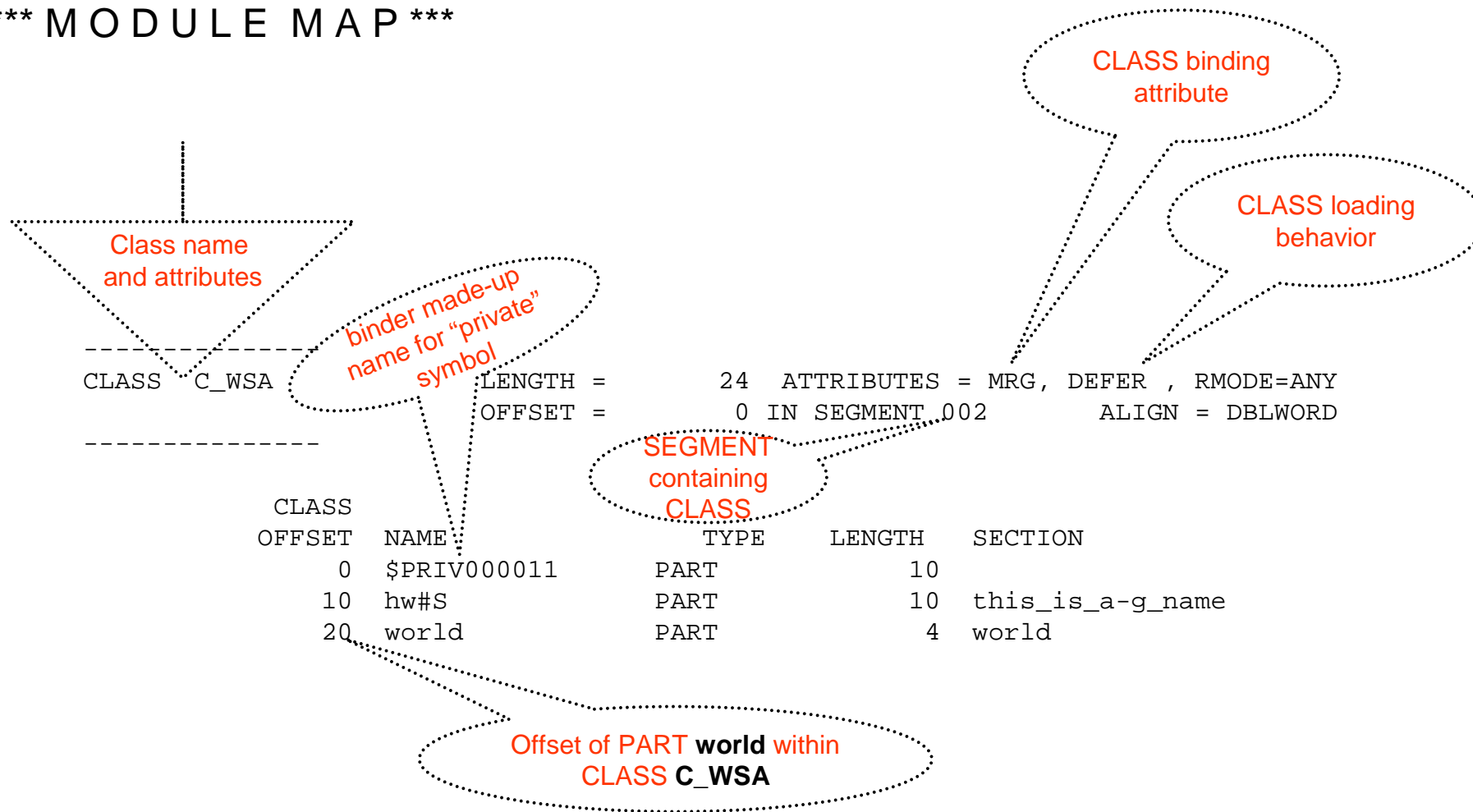


SECTION OFFSET	CLASS OFFSET	NAME	TYPE	LENGTH	DDNAME	SOURCE SEQ	MEMBER
0	0	CEESTART	CSECT	7C	/0000001	01	
0	0	CEESTART	LABEL				
80	80	this_is_a-g_name	CSECT	E0	/0000001	01	
0	80	this_is_a-g_name	LABEL				
28	A8	main	LABEL				



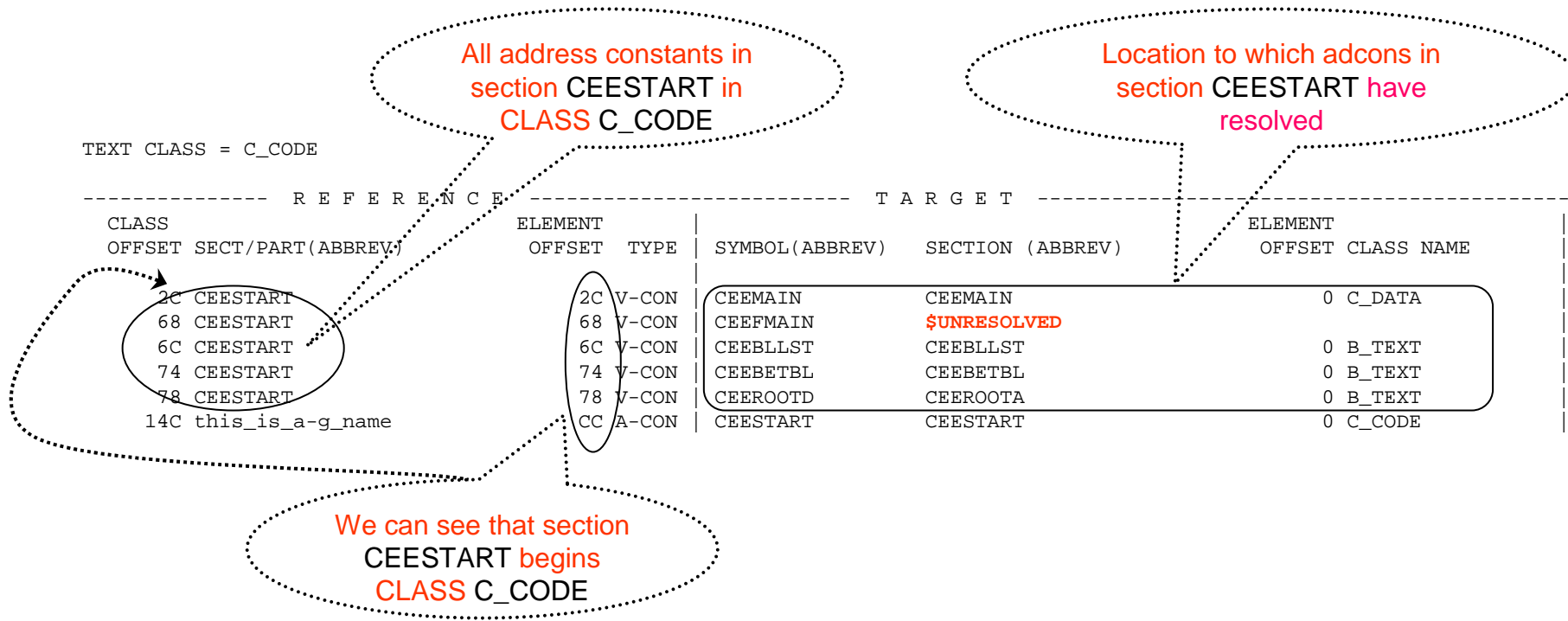
MAP ...

*** MODULE MAP ***



XREF

CROSS-REFERENCE TABLE



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                                     ELEMENT                                     ELEMENT
OFFSET SECT/PART(ABBREV)                 OFFSET  TYPE  SYMBOL(ABBREV)  SECTION (ABBREV)  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          $IMPORTED
20 world                                   20 A-CON | this_is_a-g_name  this_is_a-g_name  0 C_CODE
18 hw#S                                     18 A-CON |
1C hw#S                                     1C V-CON | CEETHLOC        CEETLOCE          8 B_TEXT
  
```

binder outputs ...

- 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**
 - Headings there even if empty
 - Due to external reference ESD entry from object module

all about AUTOCALL

- SYSLIB ddname
- AUTOCALL control statements
- LIBRARY control statements

- PDSs and PDSEs
 - “C370LIB” Object Libraries
- UNIX archive files

- Traditionally
 - The unresolved symbol name is searched for as the member name
 - The expectation is that the member, if found, will contain the symbol

- Object Libraries and UNIX archives extend this
 - They have their own directories of defined symbol names

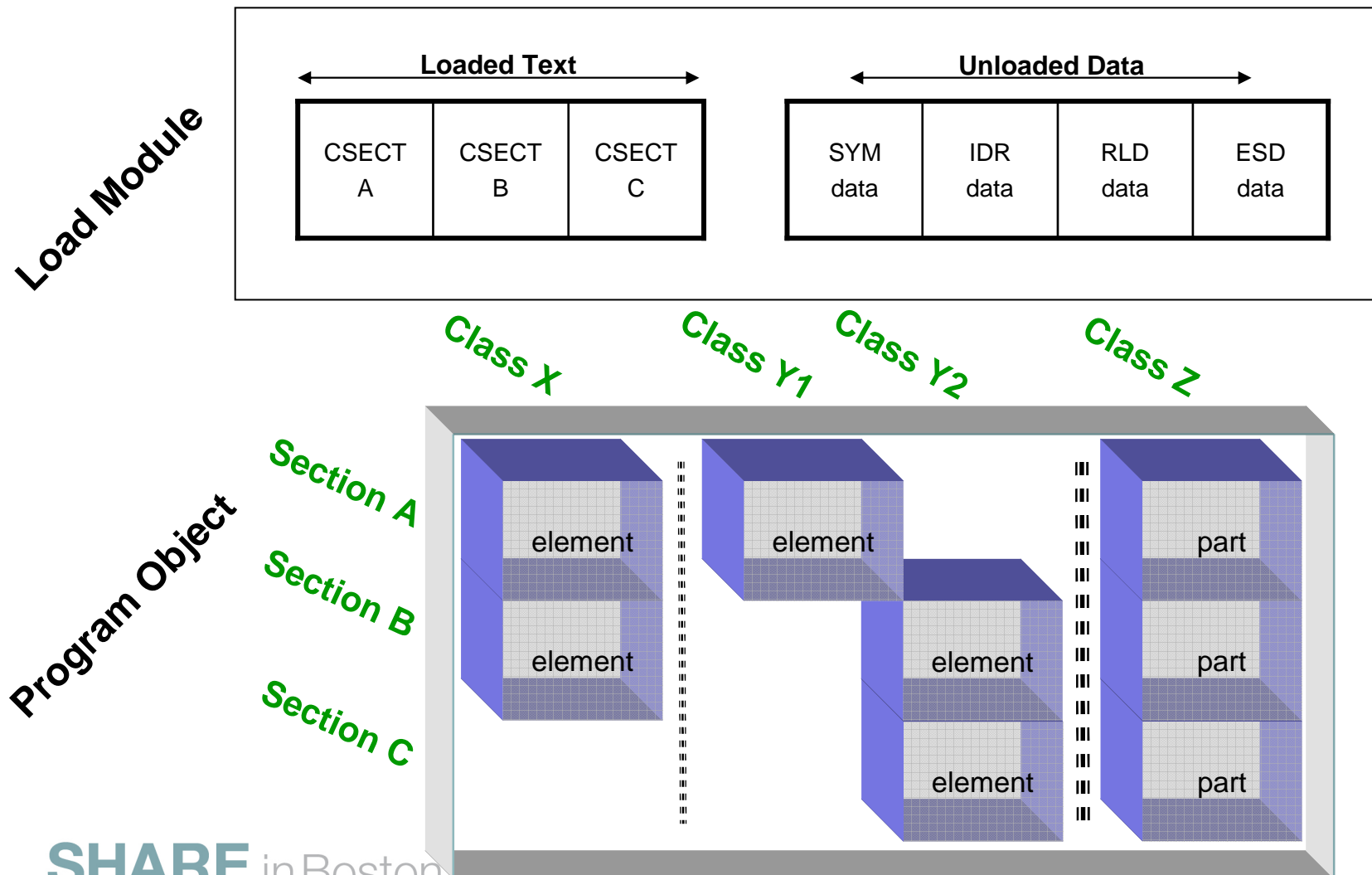
all about AUTOCALL ...

- CALL (default) or NCAL
 - CALL=YES or NOCALL or CALL=NO
- SYSLIB ddname
 - Concatenation of data sets
 - All kinds – object modules, load modules, program objects
 - Searched for only after reading all SYSLIN input

all about AUTOCALL ...

- AUTOCALL
 - UNIX “incremental”
 - Use this library right now
 - Then, forget about it!
- LIBRARY
 - Qualified with member/symbol for:
 - additional CALL (like INCLUDE but only if unresolved)
 - restricted NO-CALL
 - NEVER-CALL
 - Unqualified support added for UNIX final autocall
 - couldn't do it with SYSLIB contatenation
 - Searched in order just before SYSLIB

load module vs. program object



load module vs. program object ...

- Load modules
 - Single-dimensional
 - Documented format
 - Format never to be (substantially) be changed
- Program objects
 - Multi-dimensional
 - Class vs. section
 - Format never to be documented
 - Changes regularly – COMPAT(PMnn) levels and zOSVnRn sublevels
 - *Currently 8 levels and sublevels*

other interesting / useful stuff

- Options for even more 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)
 - **STRIPSEC/STRIPCL** to remove and list “unneeded” stuff
 - To see the “removed” report requires **MAP** option

other interesting & useful stuff ...

- **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 interesting & useful stuff ...

- COMPRESSion=YES

- 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 COMPAT level
 - *default is COMPAT(CURRENT)*
 - *will still execute back to ZOSV1R3*
 - *but no rebind, AMBLIST, ZAP, etc.*

- EDIT=NO

- *Permanently deletes* the data that COMPRESS would have compressed
- So no rebind, AMBLIST, ZAP, etc. *anywhere*

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

other interesting & useful stuff ...

- DLL – Dynamic Link Library
 - DYNAM=DLL
 - 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

other interesting & useful stuff ...

- Program Signing (introduced in z/OS V1R11)
 - SIGN=YES
 - 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

common problems & helpful tips

- Mixed-case input

IEW2456E 9207 SYMBOL myfunc UNRESOLVED. MEMBER COULD NOT BE INCLUDED FROM THE DESIGNATED CALL LIBRARY.

- Traditional names (from OBJ) are uppercase
 - Compatibility dictates the default CASE=UPPER
- Affects options values and control statement symbols
 - Option names and control statement keywords are case insensitive
 - *INCLUDE, include, Include*
- Most often an issue for IMPORT control statements (DLLs)
- Recommendations
 - CASE=MIXED
 - *Import Code,a.dll,myFunc*
 - 'quote_name'
 - *INCLUDE '/u/barryl/C/hello.o'*
 - *include PDSELIB('hello')*

common problems & helpful tips ...

- Long symbol names
 - Member names (at least in PDSs) are 8 characters
 - Problem introduced when building the object modules...
 - C/C++ LONGNAME option required
 - ... or when creating the (object) libraries
 - UNIX archive libraries manage their own internal directory
 - data set based (“C370LIB”) Object Libraries have a special directory member
 - @ @DC370\$, @ @DC390\$

common problems & helpful tips ...

- Long symbol names ...

IEW2459W 9206 INCLUDED **MEMBER s1** FAILED TO RESOLVE REFERENCE.

IEW2497W 9229 THE **SYMBOL s1** WAS EXPECTED TO BE RESOLVED BY INCLUDING **MEMBER SUB3** FROM THE LIBRARY DEFINED BY **DDNAME C8961**

- Worst case scenario!
 - Replacement object module incorrectly built (perhaps NOLONGNAME)
 - Directory member was previous built and not updated
- **IEW2497W is new for R12**
- Module already included, may resolve other symbols!

common problems & helpful tips ...

- Unresolved but it's there?
 - DYNAM=DLL may be required!
 - If “definition” is on IMPORT statement
 - *Otherwise binder processes IMPORTs but silently ignores them*

common problems & helpful tips ...

- Where did *that* come from?
 - Modules brought in by autocall
 - Turning on LIST=ALL
 - **New in R12**

IEW2340I 1036 MEMBER NAME CEEROOTD IN THE LIBRARY DEFINED BY DDNAME SYSLIB IS BEING INCLUDED TO RESOLVE REFERENCE TO CEEROOTD

IEW2308I 1112 SECTION CEERROOTA HAS BEEN MERGED.

- Especially for archives & C370LIBs

common problems & helpful tips ...

- for situations where options cannot otherwise be passed
 - particularly API based program
 - IEWPARMS
 - *like OPTIONS*
 - IEWDIAG
 - *like SYSTERM with LIST=ALL, MSGLEVEL=0*

common problems & helpful tips ...

- AMBLIST
 - LISTOBJ – all object modules
 - LISTIDR – all identification records; user IDENTIFY, language, binder, zap (EDIT=YES required)
 - LISTLOAD – all program modules (EDIT=YES required!)
 - Like binder MAP and XREF and more!
 - PMAR (partially) decoded and (fully) dumped
 - MODLIST
 - *Section / Class information ...*
 - *... including TEXT*
 - *Merge class part initializers decoded*
 - *IEWBCIE / B_IMPEXP decoded*
 - MAP
 - *SEGMENT map*
 - *Numerical MAP*
 - XREF
 - *SEGMENT map*
 - *Numerical MAP and XREF*
 - *Alphabetical MAP and XREF*
 - **New in R12 – AMBLIST LISTLOAD ebcdic translation for load modules**

more advanced stuff!

- IEWTRACE ddname – TRACE option
 - binder internal trace table
 - shows function entry / exit and other key processing points
 - shows ECODEs (part of which is 4 character code after message number)
 - can filter entries with TRACE=(start,end) or selectively TRACE='c[c...]'
- IEWDUMP – DUMP option
 - if allocated, automatically written upon terminal binder error or program check or abend
 - can be forced with DUMP option specifying ecode
 - binder continues processing for non-terminating condition
 - binder takes SNAP of binder storage and then formats key internal structures
- note: these diagnostics are normally used only for IBM problem determination
- limited information provided in program management documentation

more advanced stuff! ...

- User exits – EXIT option
 - provide module exit name
 - MESSAGE
 - filter all messages of specified severity or higher
 - prevent or allow the message to print
 - no effect on final return code of binder
 - SAVE
 - notification of each primary (member) name and alias name to be saved
 - request retry for certain failures
 - INTFVAL (Interface Validation)
 - after all input processing, including autocall
 - examine all references (resolved and unresolved) for each section
 - can allow unresolved, can change resolution to another symbol or glue
 - default exit can result in error if target & reference disagree in
 1. *ESD signature fields*
 2. *XPLINK attributes*
 3. *AMODE(64) mismatch*
 4. *Namespaces (like code (instructions) vs. data)*
 5. *Certain class attributes (like catenate vs. merge)*

more advanced stuff! ...

- Application Programming Interfaces (APIs)
 - data is input or output via buffers unique to each type of data
 - for example, ESDs
 - IEWBUFF macro can simplify creating buffers
 - *allocate, initialize, map and delete buffers*
 - *not required*
 - regular binder APIs
 - IEWBIND macro
 - *not required*
 - fast data access
 - for program objects only
 - *faster due to direct access, bypass workmod conversion*
 - request code interface
 - *obsoleted IEWBFDA macro “unitary” interface*
 - C APIs
 - NOXPLINK and XPLINK (new in R12!)
 - buffers in a header, C language oriented structures
 - simplifies access by automatically managing buffers for you
 - both regular API and fasta data access functions provided

program management documentation

- SA22-7643 - z/OS MVS Program Management:
User's Guide and Reference **for options & control statements**
- SA22-7644 - z/OS MVS Program Management:
Advanced Facilities **for binder APIs**
- GA22-7589 - z/OS MVS Diagnosis:
Tools and Service Aids **for AMBLIST and SPZAP**
- SA22-7782 - z/OS TSO/E Command Reference **for LINK and LOADGO**
- SA22-7802 - z/OS UNIX System Services
Command Reference **for c89 and ld**