Binder Programming

What the z/OS Linkage Editor Can Do for You

Session 09829

Barry.Lichtenstein@us.ibm.com

IBM Poughkeepsie

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*
- MVS

* 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.

Agenda

- Program Model
- APIs
- API Examples
- Binder Diagnostics
- Binder Options

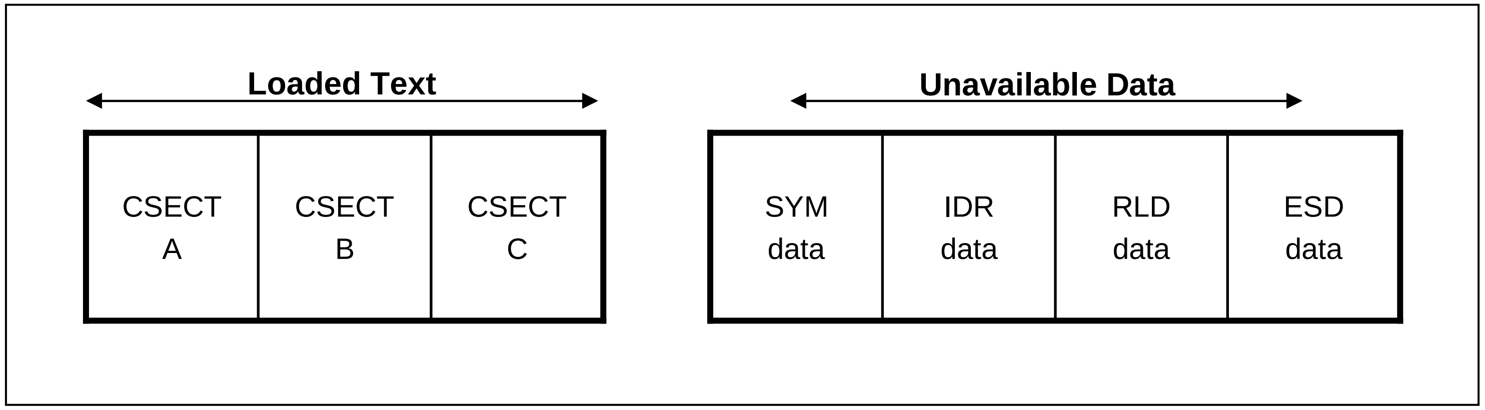
Program Model

- Load Module (LM)
  - Original MVS format
  - Fully documented in
    z/OS Program Management: Advanced Facilities
    Appendix: Load Module Formats

- Program Object (PO)
  - Exclusively created by binder
    - some areas known by loader
  - First introduced around 1991
  - Never to be documented format, requires APIs to access data

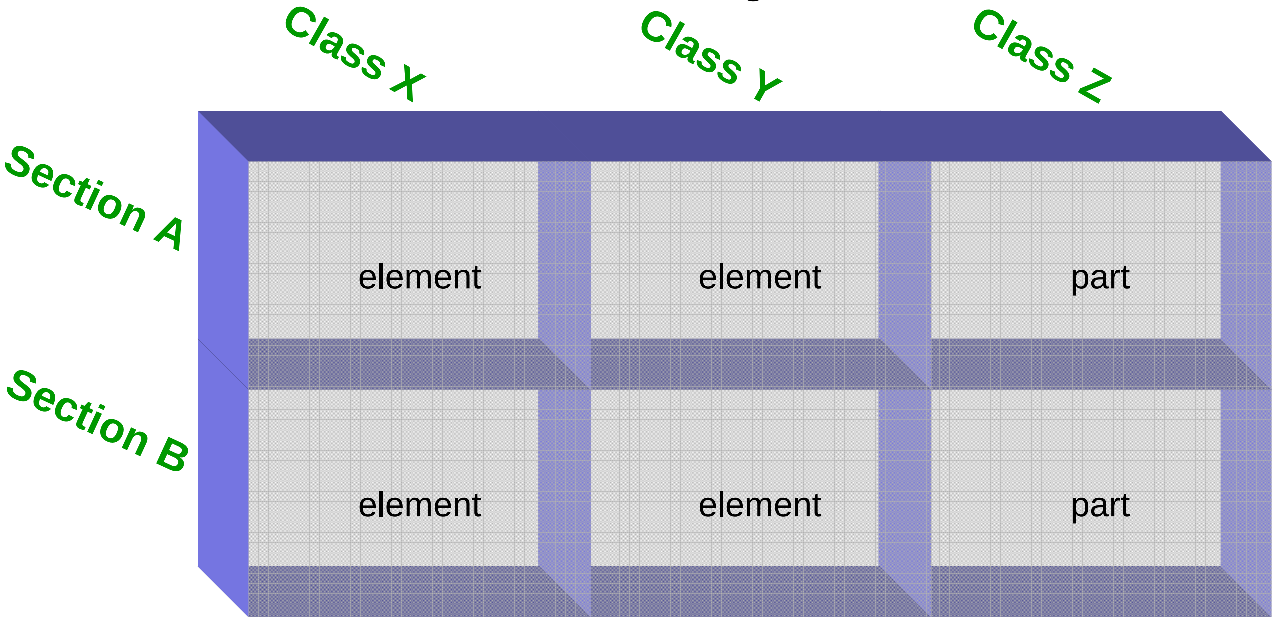
Program Model …

- Load Modules have a 1-D structure



Program Model …

- Program Objects have a 2-D model - ***class*** and ***section***
  - ***section*** is a generalization of ***csect***
  - ***class*** designates the type of data
  - a class name and section name together define an ***element*** or ***part***



Program Model …

- High Level Assembler (HLASM) instructions
  - Explicit instructions
- High Level Languages
  - Language Environment conforming
  - Implicit constructs

Program Model …

- HLASM CATTR – Establish a class (element)
  - Requires GOFF option
  - Assigns class attributes
    - Alignment
    - Content type (EXECUTABLE, NOTEXECUTABLE (data))
    - Loading behavior (initial load, DEFLOAD, NOLOAD)
    - PART – used for WSA data (variables)
      - *Implies MERGE binding rather than CONCATENATE*
      - *PRIORITY – associated with PART*
    - READONLY
    - REMOVABLE
    - Reusability (NOTREUS, REFR, RENT, REUS)
    - RMODE

Program Model …

- HLASM XATTR – External symbol attributes
  - Requires GOFF option
  - Assigns extended attributes
    - LINKAGE – OS | XPLINK
    - PSECT – referenced by R-con
    - REFERENCE – DIRECT|INDIRECT, CODE|DATA
    - SCOPE – SECTION|MODULE|LIBRARY|IMPORT|EXPORT

Program Model …

- Language Environment support for DLLs, XPLINK and LP64
  - HLASM macros CEEPDDA, CEEPLDA for variables
    - Requires GOFF option
  - C_WSA[64] is a Deferred Load (DEFLOAD) class
    - NORENT C or #pragma variable(,norent) -> C_STATIC or C_CODE
    - Also C_DATA for invariants

Program Model …

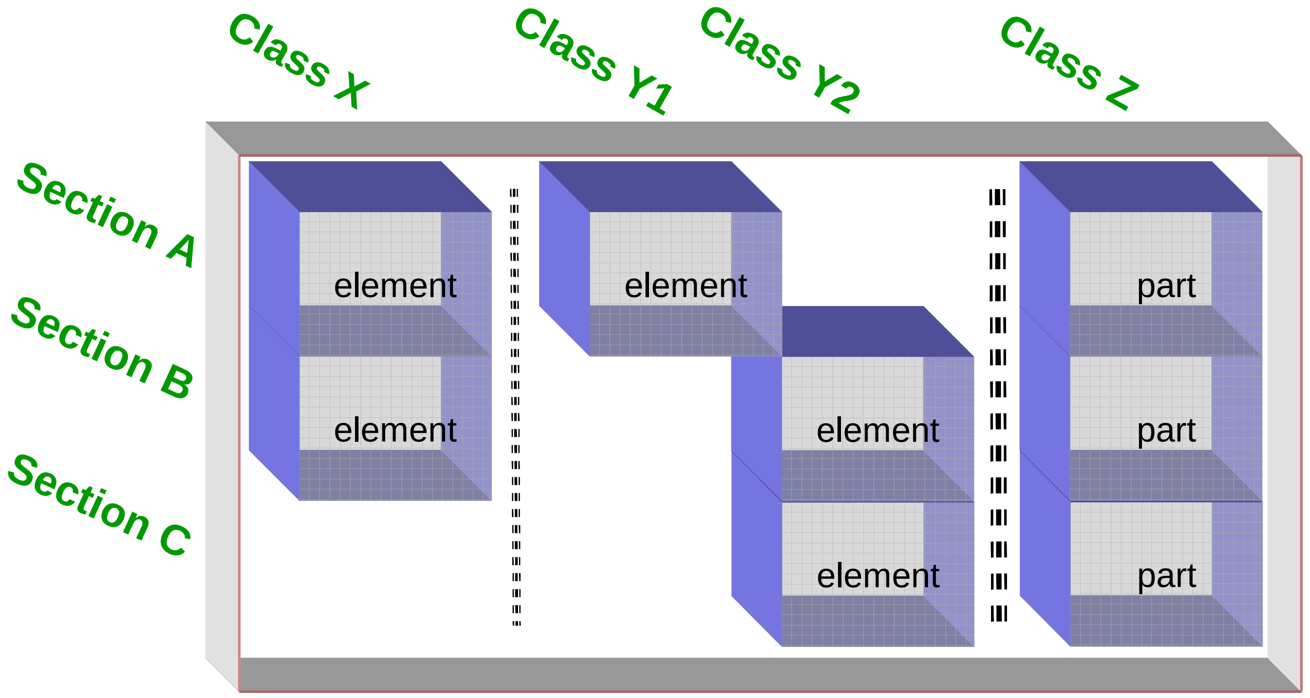
- Language Environment support for DLLs, XPLINK and LP64 …
    - global and static variables, also DLL descriptors, reside in class C_WSA[64]
    - RENT option – C, C++, COBOL, PL/I
        - DLL option – C, C++, COBOL
        - If NOXPLINK then GOFF option is optional (but preferred)

Program Model …

- Language Environment support for DLLs, XPLINK and LP64 …
  - PSECT (“environment”) associates the static part named “environment” with the section/label
    - This part contains static & unnamed variables (direct reference) and addresses of global variables (indirect reference)
    - This part is referenced by an R-con for the section/label name it was associated with (same as the V-con, not the part name itself)
    - C/C++ parts in the C_WSA[64] class…
      - ***#pragma csect(static,**name#S**)** for the PSECT part which is associated with the **#pragma csect(code,**name#C**)***
      - *same-named sections and parts for global (named) variables*

Program Model …

- Program Objects can have multiple segments
  - RMODE and loading behavior of classes used to segment



Binder APIs what for?

- *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*

Binder APIs …
what for? …

- *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 all executable modules formats*
  - *So need not code separately (PO vs. LM)*

Binder APIs …
comes in 3 flavors

- 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***

Binder APIs . . .
comes in 3 flavors . . .

- 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*

Binder APIs . . .
comes in 3 flavors . . .

- 3 – C/C++ DLLs
  - Not really a different flavor! *Introduced in z/OS V1R9*
  - 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)

Binder APIs . . .
comes in 3 flavors . . .

- 3 – C/C++ DLLs …
  - APIs in Dynamic Link Library (DLL)
    - **iewbndd.so**
    - **Iewbnddx.so** *— XPLINK introduced 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 introduced in z/OS V1R12*
  - Installs exclusively into UNIX file system

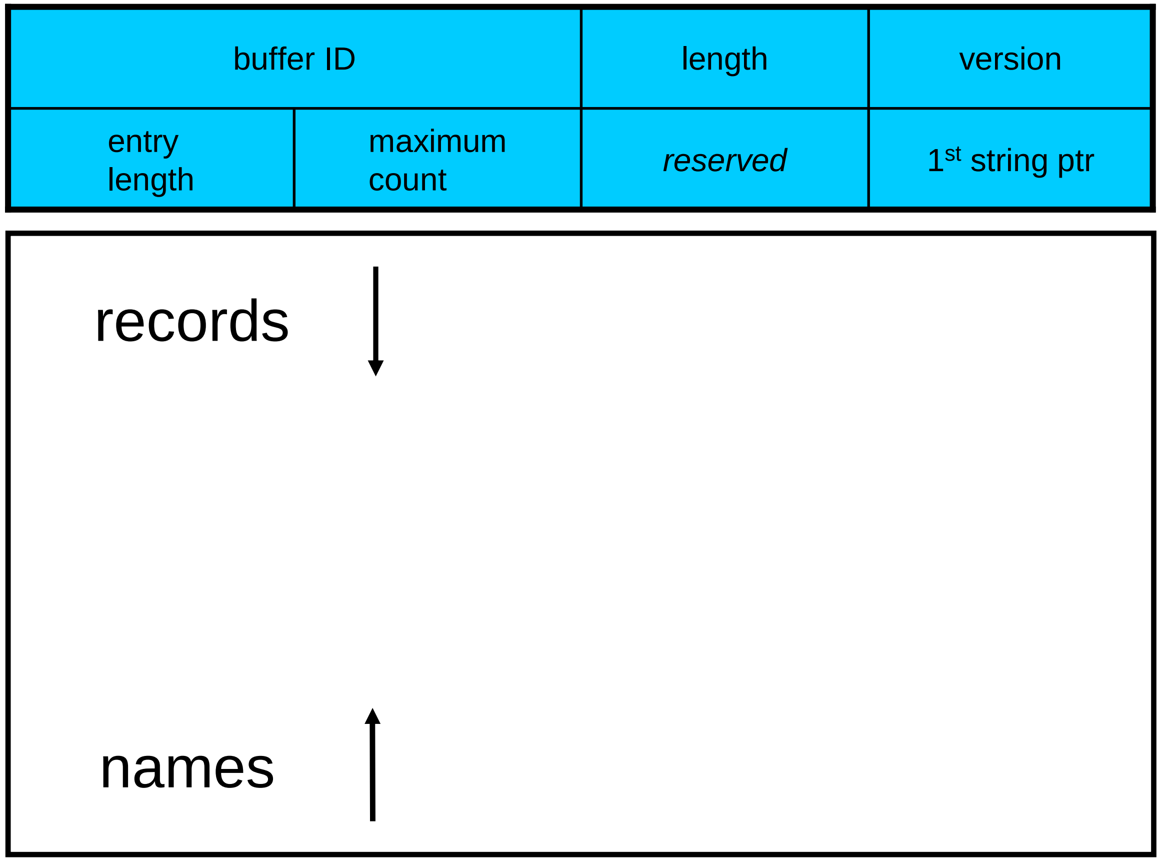
Binder APIs …
must have data buffers!

- Module data is returned in a buffer provided by the API caller
- IEWBUFF macro can help (but is not required)
- <u>Same buffer format used by both regular APIs and fast data APIs</u>
- 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

Binder APIs …
must have data buffers! …

- 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*

Binder APIs …
must have data buffers! …



Binder APIs… must have data buffers! …

- 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*

Binder APIs …
names …

- 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

API examples

- dumpClassText (C) *– compare to 'SYS1.SAMPLIB(IEWAPCCC)' introduced in z/OS V1R12!*
  - Single-source program which can be compiled for either regular or fast-data APIs
  - Uses GETN API to retrieve all binder class names and display information about them
  - Shows how to deals with new information available only in newer buffer formats
  - Optionally uses GETD API to retrieve and write out the text of the entire class (all elements)
- idModSect (C)
  - Adds an IDRU to the "module section" – binder section x'00000001' (can't be done with IDENTIFY!)
- addAlias (C)
  - Adds an ALIAS to an existing MVS program
  - Uses INTENT=ACCESS (this can't be done any other way)
- baGetE (assembler) *– introduced as 'SYS1.SAMPLIB(IEWAPBND)' in z/OS V1R12!*
  - Uses GETN API to retrieve all binder section names
  - Uses GETD API to write all ESD information
  - Shows use of the message exit

API example 1 – dumpClassText

```
/**********************************************************************/
/* dumpClassText pathname <classname>                                 */
/**********************************************************************/
#include <stdlib.h>
#include <env.h>
#include <stdio.h>
#include <string.h>
#include <strings.h>
#include <ctype.h>

#define STR(x)  STRX(x)  /* expand (x)       */
#define STRX(x) #x       /* and stringize it */

#if defined R9
#define _IEW_TARGET_RELEASE _IEW_ZOSV1R9_
#elif defined R10
#define _IEW_TARGET_RELEASE _IEW_ZOSV1R10_
#elif defined R11
#define _IEW_TARGET_RELEASE _IEW_ZOSV1R11_
#elif defined R12
#define _IEW_TARGET_RELEASE _IEW_ZOSV1R12_
#elif defined R13
#define _IEW_TARGET_RELEASE _IEW_ZOSV1R13_
#else
#define _IEW_TARGET_RELEASE _IEW_CURRENT_
#endif
#include <__iew_api.h>
```

API example 1 – dumpClassText …

```
/* context & functions */
#ifndef FD
  #define BAcntx   _IEWAPIContext
  #define BAgetN   __iew_getN
  #define BAopen   __iew_openW
  #define BAfile   __iew_includeName
  #define BAgetD   __iew_getD
  #define BAdone   __iew_closeW
  #define BAeod    __iew_eod
  #define BAgRC    __iew_get_return_code
  #define BAgRS    __iew_get_reason_code
#else
  #define BAcntx   _IEWFDContext
  #define BAgetN   __iew_fd_getN
  #define BAopen   __iew_fd_open
  #define BAfile   __iew_fd_startName
  #define BAgetD   __iew_fd_getD
  #define BAdone   __iew_fd_end
  #define BAeod    __iew_fd_eod
  #define BAgRC    __iew_fd_get_return_code
  #define BAgRS    __iew_fd_get_reason_code
#endif
```

API example 1 – dumpClassText …

```
void dumpTextHex(char * buf, int len, int tran, int records)
{
  int ii, jj, reclen;
  if (records) reclen=len/records; else reclen=0;
  for (ii=0; ii<len; ii++) {
    if (tran && (ii%32==0))
      printf("%.08X  ", ii);
    printf("%.02x", buf[ii]);
    if (tran) {
      if ((ii+1)%4==0) printf(" ");
      if ((ii+1)%32==0)  {
        printf("\n          ");
        for (jj=ii-31; jj<=ii; jj++) {
          if (reclen && ((jj)%reclen==0)) /* mark record */
            isgraph(buf[jj]) ? printf("^%01c", buf[jj]) : printf("^?");
          else
            isgraph(buf[jj]) ? printf("%02c", buf[jj]) : printf(" ?");
          if ((jj+1)%4==0) printf(" ");
        }
        printf("\n");
      }
    }
  }
  if (tran) {
    if (len%32!=0) {
      printf("\n          ");
      for (jj=len-(len%32); jj<len; jj++) {
        if (reclen && ((jj)%reclen==0)) /* mark record */
          isgraph(buf[jj]) ? printf("^%01c", buf[jj]) : printf("^?");
        else
          isgraph(buf[jj]) ? printf("%02c", buf[jj]) : printf(" ?");
        if ((jj+1)%4==0) printf(" ");
      }
      printf("\n");
    }
  }
  else
    printf("\n");
  return;
}
```

API example 1 – dumpClassText …

```
void print_name_entry(_IEWNameListEntry* _name, char *this_name)
{
  char temp_str[1025];
  temp_str[0] = '\0';

  if (((char*)_name->__bnl_name_ptr)[0] == 0) {
    sprintf(temp_str,"$PRIV%06X",*((int*)_name->__bnl_name_ptr));
  }
  else {
    sprintf(temp_str,"%.*s",_name->__bnl_name_chars,
            (char*)_name->__bnl_name_ptr);
  }

  if (this_name==NULL || strcmp(this_name, temp_str)==0) {

    printf("%-16s : elemCount=%4d, bindFlags=0x%02X, loadFlags=0x%02X, "
                   "clsLength=0x%.08X, segmentID=0x%.04X, segmentOffset=0x%.08X"
#if _IEW_TARGET_RELEASE >= _IEW_ZOSV1R10_
                                       ", align=0x%.02X, rmode=%.01X, namespace=0x%02X"
#endif
                                       "\n",
           temp_str,
           _name->__bnl_elem_count,
           _name->__bnl_bind_flags,
           _name->__bnl_load_flags,
           _name->__bnl_u1.__bnl_cls_length,
           _name->__bnl_segm_id,
           _name->__bnl_segm_off
#if _IEW_TARGET_RELEASE >= _IEW_ZOSV1R10_
                                ,
           _name->__bnl_attr.__bnl_align,
           _name->__bnl_attr.__bnl_rmode,
           _name->__bnl_namespace
#endif
                                  );
  }
}
```

API example 1 – dumpClassText …

```
void print_name_entry(_IEWNameListEntry* _name, char *this_name)
{
  char temp_str[1025];
  temp_str[0] = '\0';

  if (((char*)_name->__bnl_name_
    sprintf(temp
  }
  e
```

```
if (((char*)_name->__bnl_name_ptr)[0] == 0) {
     sprintf(temp_str,"$PRIV%06X",*((int*)_name->__bnl_name_ptr));
}
```

```
           _name
           _name-
           _name->__bnl_segm_off
#if _IEW_TARGET_RELEASE >= _IEW_ZOSV1R10_
                                    ,
           _name->__bnl_attr.__bnl_align,
           _name->__bnl_attr.__bnl_rmode,
           _name->__bnl_namespace
#endif
                                    );
  }
}
```

API example 1 – dumpClassText …

```
void print_class(BAcntx* _context, char* class) {

  _IEWNameListEntry *classes;
#ifndef FD
  unsigned int num_of_classes = 0;
#endif

  int i,count;

  /* loop on getN to get all classes */
  while ((count = BAgetN(_context,
                         _IEW_CLASS,
#ifndef FD
                         &num_of_classes,
#endif
                         &classes)) > 0)
  {
    if (classes)
    {
      /* process all CLASSes retrieved */
      for (i=0; i < count; i++)
      {
        print_name_entry(&classes[i], class);
      }
    }
    else
      fprintf(stderr,"print_class: API bug! count>0 but name buffer is NULL!\n");
  }

  if (BAeod(_context, (void **) &classes, "B_BNL"))
  {
    fprintf(stderr, " " STR(BAgetN) ": rc=%u, rs=<0X%.8X>.\n",
            BAgRC(_context),
            BAgRS(_context));
  }
}
```

API example 1 – dumpClassText …

```
void print_class(BAcntx* _context, char* class) {

   _IEWNameListEntry *classes;
#ifndef FD
   unsigned int num_o
#endif

   int i,cou

   /
   whi

   if (
   {

   }
}
```

```
  /* loop on getN to get all classes */
   while ((count = BAgetN(_context,
                          _IEW_CLASS,
  #ifndef FD
                          &num_of_classes,
  #endif
                          &classes)) > 0)
   {
     if (classes)
     {
       /* process all CLASSes retrieved */
       for (i=0; i < count; i++)
. . .
if (BAeod(_context, (void **) &classes, "B_BNL"))
```

API example 1 – dumpClassText …

```
int main(int argc, char **argv)
{

  char *path_name;
  char *class;
  long long reloc = 0;

  BAcntx *apiCntx, *retCntx;
#ifndef FD
  _IEWList* files_list;
#endif

  unsigned int rc, reason;
  void * txtbufp;

#ifndef FD
  /* initialize api flags */
  _IEWAPIFlags apiflags= {0};

  /* files list variables */
  char* files[] = { "PRINT " };
  char* files_val[] = { "./dumpClassText.out" };

  /* parms for __iew_openW() */
  char *parms="MAP=Y,XREF=Y,CASE=MIXED,TERM=Y,LIST=ALL";
#endif

  path_name = argv[1];
  class = argv[2];

#ifndef FD
  /* create file list for __iew_openW() */
  files_list = __iew_create_list(1,files,(void **)files_val);
  if (files_list == NULL)
  {
    fprintf(stderr, " create_list error: files list is null.\n");
    return 1;
  }
#endif
```

API example 1 – dumpClassText …

```
  /* open workmod session, load binder/fast data, create api context with */
  apiCntx = BAopen(
                   __TARGET_LIB__,
#ifndef FD
                   _IEW_ACCESS,
                   files_list,
                   NULL, /* exits list */
                   parms,
#endif
                   &rc,
                   &reason);
  if (apiCntx == NULL)
  {
    fprintf(stderr," " STR(BAopen) " error: rc=%u, rs=<0X%.8X>.\n",rc,reason);
    return 2;
  }
  else
  {
    fprintf(stderr," " STR(BAopen) ": rc=%u, rs=<0X%.8X>.\n",rc,reason);
  }

  /* read in the program */
  rc = BAfile(apiCntx,
              path_name,
              ""
#ifndef FD
              ,apiflags
#endif
             );
```

API example 1 – d mpC ssT t ...

```
/* open workm
apiCntx = BAo

#ifndef
```

```
/* open workmod session, load binder/fast data, create
apiCntx = BAopen(
                          __TARGET_LIB__,
#ifndef FD
                          _IEW_ACCESS,
                          files_list,
                          NULL, /* exits list */
                          parms,
#endif
                          &rc,
                          &reason);
  if (apiCntx == NULL)

. . .

 /* read in the program */
  rc = BAfile(apiCntx,
                     path_name,
                     ""
#ifndef FD
                     ,apiflags
#endif
                   );
```

API example 1 – dumpClassText …

```
if (rc)
{
  fprintf(stderr," " STR(BAfile) " error: rc=%u, rs=<0X%.8X>.\n",
          rc,BAgRS(apiCntx));
}
else
{
  fprintf(stderr," " STR(BAfile) ": rc=%u, rs=<0X%.8X>.\n",
          rc,BAgRS(apiCntx));

  print_class(apiCntx, class);

  if (class != NULL)
  {
    int records;
    /* loop thru getting all the "class" buffer until there is no more */
    while((rc = BAgetD(apiCntx,
                       class,
                       NULL, /* any sections: binder API, INTENT=ACCESS = workmod order, INTENT=BIND = address order */
                       &reloc, /* relocation address */
                       &txtbufp)) != 0)
    {
      records=rc;
      /* if structured size reported, adjust the length to dump all bytes */
           if (strcmp(class, "B_ESD") ==0) rc = rc * sizeof(_IEWESDEntry);
      else if (strcmp(class, "B_IDRB")==0) rc = rc * sizeof(_IEWBinderIDEntry);
      else if (strcmp(class, "B_IDRL")==0) rc = rc * sizeof(_IEWLanguageIDEntry);
      else if (strcmp(class, "B_IDRU")==0) rc = rc * sizeof(_IEWUserIDEntry);
      else if (strcmp(class, "B_IDRZ")==0) rc = rc * sizeof(_IEWAMASPZAPIDEntry);
      else if (strcmp(class, "B_MAP") ==0) rc = rc * sizeof(_IEWMapListEntry);
      else if (strcmp(class, "B_RLD") ==0) rc = rc * sizeof(_IEWRLDEntry);
      else records=0;
      dumpTextHex(txtbufp, rc, 1, records);
    };

    if (BAeod(apiCntx, &txtbufp, class)) {
      fprintf(stderr, " " STR(BAgetD) " failed - rc=%d, rs=<0X%.8X>\n", rc, BAgRS(apiCntx));
    }
  }
}
```

API example 1 – dumpClassText …

```
    if (rc)
    {
```

```
/* loop thru getting all the "class" buffer until there is no more */
    while((rc = BAgetD(apiCntx,

    . . .

      /* if structured size reported, adjust the length to dump all bytes

      else if (strcmp(class, "B_IDRB")==0) rc = rc * sizeof(_IEWBinderIDEn
```

```
      else if (strcmp(class, "B_IDRB")==0) rc = rc * sizeof(_IEWBinderIDEntry);
      else if (strcmp(class, "B_IDRL")==0) rc = rc * sizeof(_IEWLanguageIDEntry);
      else if (strcmp(class, "B_IDRU")==0) rc = rc * sizeof(_IEWUserIDEntry);
      else if (strcmp(class, "B_IDRZ")==0) rc = rc * sizeof(_IEWAMASPZAPIDEntry);
      else if (strcmp(class, "B_MAP") ==0) rc = rc * sizeof(_IEWMapListEntry);
      else if (strcmp(class, "B_RLD") ==0) rc = rc * sizeof(_IEWRLDEntry);
      else records=0;
      dumpTextHex(txtbufp, rc, 1, records);
    };

    if (BAeod(apiCntx, &txtbufp, class)) {
      fprintf(stderr, " " STR(BAgetD) " failed - rc=%d, rs=<0X%.8X>\n", rc, BAgRS(apiCntx));
    }
  }
}
```

API example 1 – dumpClassText …

```
#ifndef FD
  /* set api flags for __iew_closeW() */
  apiflags.__protect = 1;
#endif

  /* close workmod session, delete api context */
  retCntx = BAdone(apiCntx,
#ifndef FD
                   apiflags,
#endif
                   &rc,
                   &reason);
  if (retCntx)
  {
    fprintf(stderr," " STR(BAdone) ": ERROR, context is not NULL. \n");
  }
  fprintf(stderr," " STR(BAdone) ": rc=%d, rs=<0X%.8X>\n",
          rc, reason);

  return 0;
}
```

API example 2 – idModSect

```
/***********************************************************************/
/* idModSect pathname juldate IDRstring                               */
/*                                                                     */
/***********************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <__iew_api.h>

_IEWUserIDEntry *myIDRUp = 0;

int main(int argc, char **argv) {

  char *path_name, *id_julian, *id_string;

  char modlvlsect_name[] = "\x00\x00\x00\x01";
  char modlvlsect_str[10];

  long long reloc = 0;

  _IEWAPIContext *apiCnxt, *retCnxt;
  _IEWList* files_list;

  unsigned int rc, reason;

  /* initialize api flags */
  _IEWAPIFlags apiflags= {0};

  /* files list variables */
  char* files[] = { "PRINT " };
  char* files_val[] = { "./idModSect.out" };

  /* parms for __iew_openW() */
  char *parms="MAP=Y,XREF=Y,CASE=MIXED,TERM=Y,LIST=ALL";
```

API example 2 – idModSect …

```
path_name = argv[1];
id_julian = argv[2];
id_string = argv[3];

/* create file list for __iew_openW() */
files_list = __iew_create_list(1,files,(void **)files_val);
if (files_list == NULL)
{
  fprintf(stderr, " create_list error: files list is null.\n");
  return 1;
}

/* open workmod session, load BINDER, create api context with */
apiCnxt = __iew_openW(__TARGET_LIB__,
                      _IEW_ACCESS,
                      files_list,
                      NULL, /* exits list */
                      parms,
                      &rc,
                      &reason);
if (apiCnxt == NULL)
{
  fprintf(stderr," openW error: apiCnxt is null.\n");
  fprintf(stderr," openW error: rc=%u, rs=<0X%.8X>.\n",rc,reason);
  return 2;
}

/* set api flags for __iew_includeName() */
apiflags.__imports = 1;
apiflags.__aliases = 1;
apiflags.__attrib = 1;
```

API example 2 – idModSect …

```
/* read in program object via __iew_includeName() */
rc = __iew_includeName(apiCnxt,path_name,"",apiflags);
if (rc)
{
  fprintf(stderr," includeName error: apiCnxt is null.\n");
  fprintf(stderr," includeName error: rc=%u, rs=<0X%.8X>.\n",
          rc,__iew_get_reason_code(apiCnxt));
}
else
{

  /* string version of binder module level section x'00000001' */
  __iew_api_name_to_str(modlvlsect_name,4,modlvlsect_str);

  /* dummy getD to get a buffer */
  rc = __iew_getD(
             apiCnxt,
             "B_IDRU",
             modlvlsect_str,
             &reloc, /* relocation address */
             (void **) &myIDRUp);
  if (rc != 0) { /* no data returned */
    /* RC=4 RSN=83000800 means End of Data, RC=4 RSN=83000801 means nothing to return */
    /* we only care about more severe errors, no data or end of data is expected  */
    /* (we just needed a buffer of the right kind), so we continue on...          */
    fprintf(stderr, "getD failed - RC=%d, RS=<0X%.8X>\n", __iew_get_return_code(apiCnxt), __iew_get_reason_code(apiCnxt));
  }

  memcpy(myIDRUp->__idu_create_date,id_julian,7);
  myIDRUp->__idu_data_chars = (unsigned short) strlen(id_string);
  memcpy(myIDRUp->__idu_data, id_string, strlen(id_string));

  apiflags.__enddata = 1;
  rc = __iew_putD(
             apiCnxt,
             "B_IDRU",
             modlvlsect_str,
             (void **) &myIDRUp,
             1, /* count */
             0, /* cursor */
             apiflags);
```

API example 2 – idModSect

```
/* read in program object via __iew_includ
rc = __iew_includeName(apiCnxt,path_name,"
if (rc)
{
  fprintf(stderr," includeNa
  fprintf(stderr," includeNam
          rc,__iew_get_reason
}
else
{

  /* string
  __ie

  /* dummy g
  rc = __iew_


  if (

    /
    fprin
  }

  myIDRU
  memcpy(myID

  apiflag
  rc =
             ap
             "B_IDRU
             modlv
             (vo
             1
             0, /* cursor */
             apiflags);
```

```
/* string version of binder module level section x'
__iew_api_name_to_str(modlvlsect_name,4,modlvlse

/* dummy getD to get a buffer */
rc = __iew_getD(
           apiCnxt,
           "B_IDRU",
           modlvlsect_str,
           &reloc, /* relocation address */
           (void **) &myIDRUp);

. . .

apiflags.__enddata = 1;
   rc = __iew_putD(
           apiCnxt,
           "B_IDRU",
           modlvlsect_str,
           (void **) &myIDRUp,
           1, /* count */
           0, /* cursor */
           apiflags);
```

API example 2 – idModSect …

```
 if (rc != 0) {
      fprintf(stderr, "putD failed - RC=%d, RS=<0X%.8X>\n", rc, __iew_get_reason_code(apiCnxt));
    }
    else { /* putD worked, now save it back */
      apiflags.__replace = 1;
      rc = __iew_saveW(
                 apiCnxt,
                 path_name,
                 "",
                 apiflags);
      if (rc != 0) {
        fprintf(stderr, "saveW failed - RC=%d, RS=<0X%.8X>\n", rc, __iew_get_reason_code(apiCnxt));
      }
    }

  }

  /* set api flags for __iew_closeW() */
  apiflags.__protect = 1;

  /* close workmod session, delete api context */
  retCnxt = __iew_closeW(apiCnxt, apiflags, &rc, &reason);
  if (retCnxt)
  {
    fprintf(stderr," closeW: ERROR, context is not NULL. \n");
  }

  return 0;
}
```

API example 3 – addAlias

```
BARRYL [438] /u/barryl/binder/SHARE/SHARE117/samples $  addAlias
Usage: addAlias [-lv -a amode -e ename] dataset(member) alias[,ename[,amode]]...

 where:

   dataset may be DD:ddname.  member must always be specified.
   o If running POSIX(ON) dataset must begin with //
   o member is the existing member name to which the alias will be added.

   -a amode
     Addressing mode of addes aliases (24, 31, 64, ANY, MIN).
     o The default is addressing mode of the external symbol associated with the alias.

   -l - lowercase
     member, alias and entry names are taken 'as-is'.
     o The default is that they are uppercased.

   -e ename
     External symbol name to associate with alias.
     o If it is the name of an entry point within the program, that name is used as the entry point for the alias.
     o If it is not an entry point name, but another external name such as a pseudoregister or an unresolved external
       reference, the main entry point is used as the entry point for the alias.
     o If the symbol you specify is not defined in the program, the alias is not created and an error will be reported.
     o The default when not specifed is 'alias'.

   -v - verbose mode
     Extra (binder) output goes to stderr / SYSOUT.
     o The default is that only addAlias error messages go to stderr / SYSOUT.

   alias[,ename[,amode]] is the new alias name, and optionally an overriding ename and/or amode
     The entry point associated with the alias is determined as follows:
     o If 'ename' is specified, execution begins at that entry point.
     o If the alias symbol matches an entry name within the program, execution begins at that entry point.
     o If the alias symbol does not match an entry name within the program, execution begins at the main entry point.
     If ename or amode are specified here on an alias, the effect is only for this one alias;
     the ename and amode options (or their defaults) are in effect when unspecified here.
     amode can be specified here without ename, as in 'myalias,,24'.
```

API example 3 – addAlias …

```
/* Now add the new ALIASes */
   do {
     char *enamea, *amodea;

     alias = argv[optind++];
     if (!flags.lc)
     {
       int ii; /* might have local ename & amode in there too! */
       for (ii=0; ii<strlen(alias); ii++) alias[ii]=(char) toupper(alias[ii]);
     }

     /* set globals */
     enamea = ename;
     amodea = amode;

     /* check for locals */
     /*  external symbol */
     pos = strchr(alias, ',');
     if (pos)
     {
       *pos++ = '\0';
       if (*pos!=',' && *pos!='\0')
         enamea = pos;
       pos = strchr(pos, ',');
       if (pos)
       {
         *pos++ = '\0';
         if (*pos!=',' && *pos!='\0')
           amodea = pos;
     }
   }
```

API example 3 – addAlias …

```
rc = __iew_addA(
               apiCnxt,
               alias,
               enamea,
               _IEW_ALIAS,
               amodea);
    if (rc==0 || rc==4) /* Next addA worked, but maybe worth saying something ... */
    {
      arc = MAX(arc, rc); /* remember to return if all else works (might be 4) */
      if (rc != 0)
      {
        unsigned int rsn;
        rsn = __iew_get_reason_code(apiCnxt);
        fprintf(stderr, thisPgm ": addA warning: rc=%u, rs=<0X%.8X>",
                rc, rsn);
        switch (rsn)
        {
          case 0x83000711:
            fprintf(stderr, ": Alias name %s has already been assigned. This request will
 replace the previous request for this alias name.\n", alias);
            break;
          case 0x8e000114:
            fprintf(stderr, ": An attempt to change module attributes failed. Symbol %s
 added to the list of aliases.\n", alias);
            break;
          default:
            fprintf(stderr, ".\n");
            break;
        }
      }
    }
  } while ((rc==0 || rc==4) && (optind<argc));
```

API example 3 - add Alia ...

```
rc = __iew_addA(
               apiCnx
               lias,

               _I

 if (rc==0                                            ... */
 {
   ar
   if (r

     u

     {

replace the                                                will

       ase 0x
         fpr
added to the         of                        ed. Symbol %s
         break;
       default:
         fprintf(stderr,      n");
         break;
     }
   }
 }
} while ((rc==0 || rc==4) && (optind<argc));
```

rc = __iew_addA(
apiCnxt,
alias,
enamea,
_IEW_ALIAS,
amodea);

API example 3 – addAlias …

```
if (rc==0 || rc==4) /* All the addA's worked, now save it back */
   {
     apiflags.__replace = 1;
     rc = __iew_saveW(
                apiCnxt,
                path_name,
                member,
                apiflags);
     if (rc == 0)
     {
       if (flags.dbg) fprintf(stderr, thisPgm ": saveW: rc=%u,
 rs=<0X%.8X>.\n", rc, __iew_get_reason_code(apiCnxt));
     }
     else /* saveW failed */
     {
       fprintf(stderr, thisPgm ": saveW failed - rc=%d, rs=<0X%.8X>\n", rc,
 __iew_get_reason_code(apiCnxt));
     }
   }
```

API example 3 - add Alia ...

```
if (rc==0 ||     =4)                                          ack
   {
     apif
     rc =



     if (

       if
  rs=<0X
     }
     else     sav
     {
       fprin   (stde                                    <0X%.8X>\n", rc,
  __iew_get_reason_     (ap
     }
   }
```

apiflags.__replace = 1;
rc = __iew_saveW(
apiCnxt,
path_name,
member,
apiflags);

API example 4 – baGetE

```
**********************************************************************
*                                                                    *
* LICENSED MATERIALS - PROPERTY OF IBM                               *
*                                                                    *
* 5694-A01                                                           *
*                                                                    *
* COPYRIGHT IBM CORP. 1977, 2010                                     *
*                                                                    *
* STATUS = HPM7770                                                   *
*                                                                    *
**********************************************************************
*                                                                    *
*                      SAMPLE BINDER PROGRAM                         *
*                                                                    *
*  FUNCTION: Example application which includes a module and prints *
*      its ESD records using the Binder call interface functions     *
*      INCLUDE, GETN, and GETE.                                      *
*                                                                    *
*  PROCESSING:                                                       *
*      Broken up into these steps, and referred to by these numbers  *
*      throughout:                                                   *
*                                                                    *
*      A. Initialization:                                            *
*                                                                    *
*      B. Main processing:                                           *
*                                                                    *
*      C. Finishing up:                                              *
*                                                                    *
*  CONSTANTS, VARIABLES, BUFFER MAPPINGS AND MESSAGE EXIT ROUTINES:  *
*                                                                    *
**********************************************************************
```

API example 4 – baGetE

```
***********************************************************************
* PROGRAM INITIALIZATION                                              *
***********************************************************************
BAGETE   CSECT
         PRINT GEN
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         SAVE  (14,12)
         BASR  R12,0              Get 31-bit base even in 24-bit mode
         USING *,R12
         ST    R12,MESSAGE+4      Save program base for message exit
         LA    R15,SAVE
         ST    R13,SAVE+4
         ST    R15,8(,13)
         LR    R13,R15
         SPACE
         MVC   FREEBFR,ZERO       No buffers to FREEBUF yet
         MVC   CLSDCB,ZERO        No DCB to close yet
         MVC   ENDDLG,ZERO        No Dialog to end yet
```

API example 4 – baGetE …

```
**********************************************************************
*                2. Open output data set                             *
*                                                                    *
* This logic opens the output file.                                  *
*****                                                            *****
         OPEN  (MYDCB,OUTPUT)      Open output data set
         LTR   R15,R15             Successful?
         BNZ   ERREXIT             Exit if not
         MVC   CLSDCB,FOUR         We must CLOSE our DCB on exit
         SPACE
**********************************************************************
*                3. Obtain and initialize binder buffers             *
*                                                                    *
* These specifications of the IEWBUFF macro obtain storage for the   *
* ESD and NAMES buffers and initialize them. Mapping DSECTs for      *
* the buffers are provided at the end of the program.                *
*****                                                            *****
       IEWBUFF FUNC=GETBUF,TYPE=ESD
       IEWBUFF FUNC=GETBUF,TYPE=NAME
       IEWBUFF FUNC=INITBUF,TYPE=ESD
       IEWBUFF FUNC=INITBUF,TYPE=NAME
         MVC   FREEBFR,FOUR        We must FREEBUF our buffers on exit
         SPACE
```

API example 4 – baGetE …

```
**********************************************************************
*                2. Open output data set                             *
*                                                                    *
* This logic opens the output file.
*****
         OPEN  (MYDCB,OUTPUT)
         LTR   R15,R15
         BNZ
         MVC   CL

***********
*
*
* Thes
* ESD and
* the bu
*****
        IEWB
        IEWBU
        IEWBUFF F
        IEWBUFF FUNC=IN
         MVC   FREEBF
         SPACE
```

```
IEWBUFF FUNC=GETBUF,TYPE=ESD
IEWBUFF FUNC=GETBUF,TYPE=NAME
IEWBUFF FUNC=INITBUF,TYPE=ESD
IEWBUFF FUNC=INITBUF,TYPE=NAME
```

API example 4 – baGetE …

```
**********************************************************************
*               4. Start Dialog, specifying lists                    *
*                                                                    *
* The STARTD call establishes a dialog with the binder. It is always *
* required and sets the dialog token for use in subsequent binder    *
* calls. The dialog token must be initialized to binary zero before  *
* its usage.                                                         *
*                                                                    *
* The example uses all three list parameters on the STARTD call:     *
*  - FILES allows us to assign a ddname to the binders print file.   *
*    Note that when using the binder API, any required binder files  *
*    (those whose ddnames do not appear on binder control statements *
*    or as macro parameters) must have ddnames assigned in this way. *
*  - EXITS allows us to specify a message exit routine that receives *
*    control, in this case, if the message severity is greater than  *
*    0. The exit routine appears at the end of this program.         *
*  - OPTIONS allow us to specify one or more options that will apply *
*    throughout the binder dialog. In this example, option TERM is   *
*    set to Y.                                                       *
*****                                                            *****
         MVC   DTOKEN,DZERO Clear dialog token
               IEWBIND FUNC=STARTD,                                    +
               RETCODE=RETCODE,                                        +
               RSNCODE=RSNCODE,                                        +
               DIALOG=DTOKEN,                                          +
               FILES=FILELIST,                                         +
               EXITS=EXITLIST,                                         +
               OPTIONS=OPTLIST,                                        +
               VERSION=4
         CLC   RSNCODE,ZERO       Check the reason code
         BNE   ERREXIT            Exit if not zero
         MVC   ENDDLG,FOUR        We must ENDDIALOG on exit
         EJECT
```

API example 4 – baGetE …

```
**********************************************************************
*               5. Create a Workmod with Intent ACCESS               *
*                                                                    *
* This logic creates a binder workmod with INTENT=ACCESS. The dialog *
* token, DTOKEN, is a required input parameter. The workmod token,   *
* WTOKEN, is set by the binder for use on subsequent calls. The      *
* workmod token must be initialized to binary zero prior to the      *
* CREATEW call.                                                      *
*****                                                           *****
         MVC   WKTOKEN,DZERO      Clear workmod token
       IEWBIND FUNC=CREATEW,                                           +
               RETCODE=RETCODE,                                        +
               RSNCODE=RSNCODE,                                        +
               WORKMOD=WKTOKEN,                                        +
               DIALOG=DTOKEN,                                          +
               INTENT=ACCESS,                                          +
               VERSION=4
         CLC   RSNCODE,ZERO       Check the reason code
         BNE   ERREXIT            Exit if not zero
         EJECT
**********************************************************************
*               6. Set the list option to ALL                        *
*                                                                    *
* SETO is used to set the LIST option to ALL. Since the workmod token*
* is provided on the macro, LIST is set at the workmod level and is  *
* valid only until the workmod is reset.                             *
*****                                                           *****
       IEWBIND FUNC=SETO,                                              +
               RETCODE=RETCODE,                                        +
               RSNCODE=RSNCODE,                                        +
               WORKMOD=WKTOKEN,                                        +
               OPTION=LIST,                                            +
               OPTVAL=ALL,                                             +
               VERSION=4
         CLC   RSNCODE,ZERO       Check the reason code
         BNE   ERREXIT            Exit if not zero
         EJECT
```

API example 4 – baGetE …

```
**********************************************************************
*                        MAIN PROGRAM                                *
**********************************************************************
**********************************************************************
*                7. Include a module (IFG0198N)                      *
*                                                                    *
* This step includes member IFG0198N from library LPALIB, using      *
* ddname and member name to identify the module to be included.      *
*****                                                            *****
       IEWBIND FUNC=INCLUDE,                                          +
               RETCODE=RETCODE,                                       +
               RSNCODE=RSNCODE,                                       +
               WORKMOD=WKTOKEN,                                       +
               INTYPE=NAME,                                           +
               DDNAME=INCLLIB,                                        +
               MEMBER=MODNAME,                                        +
               VERSION=4
         CLC   RSNCODE,ZERO        Check the reason code
         BNE   ERREXIT             Exit if not zero
         EJECT
```

API example 4 – baGetE …

```
**********************************************************************
*                8. Get all section names from workmod               *
*                                                                    *
* The GETN call retrieves from the workmod the names of all sections *
* in module IFG0198N. Names are returned in the names buffer,        *
* IEWBBNL, and COUNTN is set to the number of names returned. TCOUNT *
* is set to the total number of names in the module, regardless of   *
* size of the buffer. For this example, the two counts should be the *
* same. The size of the buffer is controlled by the second IEWBUFF   *
* macro in step 18, which specifies SIZE=50. This provides space     *
* for up to 50 names. Since IFG0198N has fewer than 50 sections, the *
* GETN request reaches end of file before filling the buffer. That is*
* why it ends with return code 4, and why TCOUNT and COUNTN are the  *
* same.                                                              *
*****                                                            *****
         MVC   CURSORN,ZERO
       IEWBIND FUNC=GETN,                                              +
               RETCODE=RETCODE,                                        +
               RSNCODE=RSNCODE,                                        +
               WORKMOD=WKTOKEN,                                        +
               AREA=IEWBBNL,                                           +
               CURSOR=CURSORN,                                         +
               COUNT=COUNTN,                                           +
               TCOUNT=TCOUNT,                                          +
               NTYPE=S,                                                +
               VERSION=4
               CH    R15,=H'4'           RC=4 means have all names
         BE    GETNOKAY
         BH    ERREXIT             Any higher is an error
         PUT   MYDCB,MSG2MANY      RC=0: Too many sections
GETNOKAY EQU   *
         EJECT
```

API example 4 – baGetE …

```
***********************                    ******            *****
*          8. G
*
```

```
    IEWBIND FUNC=GETN,                                       +
            RETCODE=RETCODE,                                 +
            RSNCODE=RSNCODE,                                 +
            WORKMOD=WKTOKEN,                                 +
            AREA=IEWBBNL,                                    +
            CURSOR=CURSORN,                                  +
            COUNT=COUNTN,                                    +
            TCOUNT=TCOUNT,                                   +
            NTYPE=S,                                         +
            VERSION=4
```

```
               NTYPE
               VERSIO
               CH    R15,=H'4'          RC=4     s have a      ames
         BE    GETNOKAY
         BH    ERREXIT            Any higher is an error
         PUT   MYDCB,MSG2MANY     RC=0: Too many sections
GETNOKAY EQU   *
         EJECT
```

API example 4 – baGetE …

```
**********************************************************************
*               9. Get ESD data for each name returned by GETN       *
*****                                                            *****
         L     R5,COUNTN             Number of sections
LOOP1    L     R3,BNL_NAME_PTR       Extract section name
         LH    R2,BNL_NAME_CHARS
         STH   R2,SECTION
         LA    R4,SECTION
         BCTR  R2,0
         EX    R2,MOVESEC
         MVC   CURSORD,ZERO          Reset cursor
       IEWBIND FUNC=GETD,                                              +
               RETCODE=RETCODE,                                        +
               RSNCODE=RSNCODE,                                        +
               WORKMOD=WKTOKEN,                                        +
               CLASS=B_ESD,                                            +
               SECTION=SECTION,                                        +
               AREA=IEWBESD,                                           +
               CURSOR=CURSORD,                                         +
               COUNT=COUNTD,                                           +
               VERSION=4
         CLC   RSNCODE,ZERO
         BE    GETDOKAY
         CLC   RETCODE,FOUR          Last buffer
         BE    GETDOKAY
         CLC   RETCODE,EIGHT         No data for item
         BNE   ERREXIT
GETDOKAY EQU   *
         L     R4,COUNTD             Number of ESD entries in buffer
         LTR   R4,R4                 Skip empty section
         BZ    NEXTSECT
         LA    R7,ESDH_END           First record in ESD buffer
         SH    R7,=H'4'              Leave space for length info
         L     R0,ESDH_ENTRY_LENG
         AH    R0,=H'4'
         SLL   R0,16                 Convert to LLBB form
LOOP2    DS    0H
         ST    R0,0(,R7)
         PUT   MYDCB,(R7)            Write ESD to output data set
         L     R0,0(,R7)
         A     R7,ESDH_ENTRY_LENG Move to next ESD in this section
         BCT   R4,LOOP2
NEXTSECT A     R9,BNLH_ENTRY_LENG Move to next section name
         BCT   R5,LOOP1
         SPACE
```

API example 4 – baGetE …

```
**********************************************************************
*              9. Get ESD data for each name returned by GETN       *
*****                                                           *****
         L     R5,COUNTN           Number of sections
LOOP1    L     R3,BNL_NAME_PTR     Extract section name
         LH    R2,BNL_NAME_CHARS
         STH   R2,SECTION
         LA    R4,SECTION
         BCTR  R2,0
         EX    R2,MOVESEC
         MVC   CURSORD,ZERO        Reset cursor
       IEWBIND FUNC=GETD,                                             +
               RETCODE=RETCODE,                                       +
               RSNCODE=RSNCODE,                                       +
               WORKMOD=WKT
               CLASS=B_ESD
```

```
IEWBIND FUNC=GETD,                                                   +
        RETCODE=RETCODE,                                             +
        RSNCODE=RSNCODE,                                             +
        WORKMOD=WKTOKEN,                                             +
        CLASS=B_ESD,                                                 +
        SECTION=SECTION,                                             +
        AREA=IEWBESD,                                                +
        CURSOR=CURSORD,                                              +
        COUNT=COUNTD,                                                +
        VERSION=4
```

```
NEXTSECT A
         BCT
         SPACE
```

API example 4 – baGetE …

```
***********************************************************************
*              10. Done processing - delete workmod                   *
*                                                                     *
* DELETEW removes the workmod from binder storage. PROTECT=YES, the   *
* default, merely indicates that the delete should fail if the        *
* workmod has been altered by the dialog. Since INTENT=ACCESS, no     *
* alteration was possible, and PROTECT=YES is ineffective.            *
*****                                                             *****
       IEWBIND FUNC=DELETEW,                                            +
               RETCODE=RETCODE,                                         +
               RSNCODE=RSNCODE,                                         +
               WORKMOD=WKTOKEN,                                         +
               PROTECT=YES,                                             +
               VERSION=4
         CLC   RSNCODE,ZERO
         BNE   ERREXIT
         SPACE
***********************************************************************
*              11. End dialog                                         *
*                                                                     *
* ENDD ends the dialog between the program and the binder, releasing *
* any remaining resources, closing all files, and resetting the       *
* dialog token to the null value.                                     *
*****                                                             *****
       IEWBIND FUNC=ENDD,                                               +
               RETCODE=RETCODE,                                         +
               RSNCODE=RSNCODE,                                         +
               DIALOG=DTOKEN,                                           +
               VERSION=4
         CLC   RSNCODE,ZERO
         BNE   ERREXIT
         SPACE
***********************************************************************
*              12. FREEBUF (Release) our buffer storage               *
*****                                                             *****
FREEBUFS IEWBUFF FUNC=FREEBUF,TYPE=ESD
         IEWBUFF FUNC=FREEBUF,TYPE=NAME
```

API example 4 – baGetE …

```
**********************************************************************
*              13. Close output dataset                              *
*****                                                            *****
CLOSEDCB CLOSE (MYDCB)
      FREEPOOL MYDCB
         SPACE
**********************************************************************
*              14. Return to operating system                        *
*****                                                            *****
NORMEXIT EQU   *
         LA    R15,0              Set a reason code of zero
         B     EXIT
ERREXIT  EQU   *
         CLC   FREEBFR,FOUR       Do we need to FREEBUF our buffers?
         BNE   CHECKDLG
       IEWBUFF FUNC=FREEBUF,TYPE=ESD
       IEWBUFF FUNC=FREEBUF,TYPE=NAME
CHECKDLG CLC   ENDDLG,FOUR        Do we need to end the Dialog?
         BNE   CHECKDCB
*        Ending the dialog also deletes the workmod
       IEWBIND FUNC=ENDD,                                              +
               RETCODE=RETCODE,                                        +
               RSNCODE=RSNCODE,                                        +
               DIALOG=DTOKEN,                                          +
               PROTECT=NO,                                             +
               VERSION=4
CHECKDCB CLC   CLSDCB,FOUR        Do we need to CLOSE and FREE our DCB?
         BNE   SETRSN
         CLOSE (MYDCB)
      FREEPOOL MYDCB
SETRSN   L     R15,RSNCODE
EXIT     L     R13,SAVE+4
        RETURN (14,12),RC=(15)
```

API example 4 – baGetE …

```
**********************************************************************
*                       PROGRAM CONSTANTS                            *
**********************************************************************
DZERO    DC    2F'0'
ZERO     DC    F'0'
FOUR     DC    F'4'
EIGHT    DC    F'8'
MOVESEC  MVC   2(0,R4),0(R3)
MSG2MANY DC    Y(MSG2MZ-*,0),C'TOO MANY SECTIONS TO DISPLAY'
MSG2MZ   EQU   *
**********************************************************************
*              15. Variable length string constants                  *
*****                                                            *****
B_ESD    DC    H'5',C'B_ESD'      Class name
ALL      DC    H'3',C'ALL'        LIST option value
INCLLIB  DC    H'6',C'LPALIB'     Include library
LIST     DC    H'4',C'LIST'       LIST option keyword
MODNAME  DC    H'8',C'IFG0198N'   Member name
TERM     DC    H'4',C'TERM'       TERM option keyword
Y        DC    H'1',C'Y'          TERM option value
**********************************************************************
*              16. STARTD list specifications                        *
*****                                                            *****
FILELIST DS    0F                 ddname specifications
         DC    F'1'               Number of list entries
         DC    CL8'PRINT',F'8',A(PRINTX) Assign print file ddname
PRINTX   DC    CL8'SYSPRINT'      The ddname
         SPACE
OPTLIST  DS    0F                 Global options specifications
         DC    F'1'               Number of list entries
         DC    CL8'TERM',F'1',A(YX) Set TERM option
YX       DC    C'Y'               TERM option value
EXITLIST DS    0F                 User exit specifications
         DC    F'1'               Number of list entries
         DC    CL8'MESSAGE',F'12',A(MESSAGE) Specify MESSAGE exit
MESSAGE  DC    A(MSGEXIT)         Exit routine entry point
         DC    AL4(0)             Base address for exit routine
         DC    A(FOUR)            Take exit for severity >= 4
```

API example 4 – baGetE …

```
**********************************************************************
*                        WORKING STORAGE                             *
**********************************************************************
SAVE     DS    18F                 Register save area
SAVE2    DS    18F                 Another for the exit routine
SAVE13   DS    F                   Register 13 save
COUNTD   DS    F                   Number of ESD records returned
COUNTN   DS    F                   Number of section names
CURSORD  DS    F                   Cursor value for GETD call
CURSORN  DS    F                   Cursor value for GETN call
DCB@     DS    F                   DCB for output file
DTOKEN   DS    CL8                 Dialog Token
RETCODE  DS    F                   General return code
RSNCODE  DS    CL4                 General reason code
SECTION  DS    H,CL8               Section Name for GETD
TCOUNT   DS    F                   Total number of sections
WKTOKEN  DS    CL8                 Workmod Token
MSGLEN   DS    F
MSG      DC    80C'0'              Put message buffer
FREEBFR  DS    F                   Indicator for FREEBUFing our buffers
*                                  on exit, if they were GETBUFfed.
CLSDCB   DS    F                   Indicator for closing our DCB
ENDDLG   DS    F                   Indicator for ENDDing the Dialog
**********************************************************************
*              17. DCB for output file                               *
*****                                                            *****
MYDCB    DCB   DSORG=PS,MACRF=PM,RECFM=VB,LRECL=300,DDNAME=MYDDN
*****                                                            *****
**********************************************************************
*              18. NAMES and ESD Buffer Mappings.                    *
*****                                                            *****
       IEWBUFF FUNC=MAPBUF,TYPE=ESD,SIZE=50,                           +
               HEADREG=6,ENTRYREG=7,VERSION=4
       IEWBUFF FUNC=MAPBUF,TYPE=NAME,SIZE=50,                          +
               HEADREG=8,ENTRYREG=9,VERSION=4
         LTORG
```

API example 4 – baGetE …

```
**********************************************************************
*                        MESSAGE EXIT ROUTINE                        *
*                                                                    *
*        This exit routine merely prints out a message as an example *
*        of how the print exit could be used, not how it should      *
*        be used.                                                    *
**********************************************************************
**********************************************************************
*              19. Message Exit Routine                              *
*                                                                    *
*        Note: This routine will always be entered in AMODE(31).     *
*        If AMODE(24) is required, capping code must be added.       *
*****                                                            *****
MSGEXIT  EQU   *
         SAVE  (14,12)
         L     R12,0(,R1)         Get address of user data
         L     R12,0(,R12)        Get user data(pgm base register)
         L     R4,28(,R1)         Get address of exit return code
         XC    0(4,R4),0(R4)      Set exit return code to zero
         L     R3,4(,R1)          Get address of address of msg buf
         L     R3,0(,R3)          Get address of message buffer
         LH    R1,0(,R3)          Length of the message
         LA    R0,L'MSG
         CR    R1,R0
         BNL   MSGX2
         LR    R1,R0              But limited to buffer length
MSGX2    DS    0H
         LA    R0,4(,R1)          Length+4 for QSAM
         SLL   R0,16              Convert to LLBB form
         ST    R0,MSGLEN
         BCTR  R1,0               Length-1 for Execute
         EX    R1,MOVEMSG         Put all we can in the buffer
         LA    R3,MSGLEN
         ST    R13,SAVE13         Save input save area address
         LA    R13,SAVE2          Save area for PUT
         PUT   MYDCB,(R3)         Write message to data set
         L     R13,SAVE13         Restore save area register
        RETURN (14,12)            Return to binder
*
MOVEMSG  MVC   MSG(0),2(R3) Executed above
         END BAGETE
```

Binder Diagnostics

- **Documented in z/OS Program Management: User's Guide**
- **Designed for IBM and API writers:**
  - Error messages (IEWDIAG)
    - Helpful diagnostic device for API users, captures messages as if MSGLEVEL=0, LIST=ALL
    - Especially useful if there is no way to turn on options to write to SYSPRINT
- **Designed for IBM to debug binder problems:**
  - Internal Trace Table (IEWTRACE)
    - ALL or ECODEs (start[,end]) or subcomponent ID(s)
  - Formatted Internals Dump (IEWDUMP)
    - Logic error, Program Check, Abend, or ECODE
  - XOBJ-to-GOFF conversion (IEWGOFF)

Binder options

- Installation options from IEWBODEF
  - Customizing procedure in z/OS Program Management: Advanced Facilities Appendix. Establishing Installation Defaults
  - Great care must be taken since all binder use is affected!
    - Though IEWBODEF links into batch interface module IEWBLINK, it is processed by the binder regular APIs!
- Primary invocation options, from one of the following:
  - The PARM field of the JCL EXEC statement
  - The first parameter passed to IEWBLINK, IEWBLOAD, etc.
  - STARTD API PARMS= parameter string
  - Note: These PARMS may contain the OPTIONS option
- The IEWPARMS DD statement *– introduced in z/OS V1R11 !*

. . .

Binder options …

. . .

- STARTD API OPTIONS= options list
- IEWBIND_OPTIONS environment variable options string (passed via STARTD API ENVARS= environment variables)
- Dynamic option changes:
  - The SETOPT control statement parameter string
  - SETO API PARMS= parameter string
  - SETO API OPTION= name OPTVAL= value

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**

Program Object format features

- ZOSV1R1 / ZOSV1R2 (Format 3)
  - Prelinker-less binding of C/C++ & DLLs
  - XPLink
- ZOSV1R3 / ZOSV1R4 (Format 4 Compat z/OS V1R3)
  - AMODE=64
  - symbol length > 1024 (EDIT=YES)
- ZOSV1R5 / ZOSV1R6 (Format 4 Compat z/OS V1R5)
  - RMODE=64 WSA…. So, C/C++/LE 64-bit
- ZOSV1R7 (Format 4 Compat z/OS V1R7)
  - Relative-Immediate across compilation units (GOFF)
  - Compression
- ZOSV1R8 / ZOSV1R9 (Format 5 Compat z/OS V1R8)
  - Relative-Immediate across segments & in OBJ/LM
- ZOSV1R10 / ZOSV1R11/ZOSV1R12 (Format 5 Compat z/OS V1R10)
  - Save extended IDRL timestamp (in addtion to datestamp)
  - QY-con RLD (RXY instruction displacement)
- ZOSV1R13 (Format 5 Compat z/OS V1R13)