Oracle BRM Training

By Nedhya Sharma

AGENDA

1. BRM Flist & Storable Objects Continue..
2. BRM PCM Macros
3. BRM PIN Macros

BRM Flist & Storable Object

FLIST
Flist is a field value pair.

Example: If you take one record from the table,the table field/column name is the Field and the contents inside that column is Value.

account_t table fields:

poid_id0 | poid_type | poid_rev|account_no | business_type

When ever data is getting from database to flist, the column names for the table will be prefixed with PIN_FLD_<tablecolumn>

For the above table fields the Flist field names will be as follows:

PIN_FLD_POID
PIN_FLD_ACCOUNT_NO
PIN_FLD_BUSINESS_TYPE

We learnt about parent table and child tables. Example: account_t is a parent, account_nameinfo_t and account_phones_t are child tables.

The child table will be represented as ARRAY or SUBSTRUCT in the FLIST at 0 level, prefixing the PIN_FLD_<childtablename> removing the parent table name prefix. In the below example account_t is the parent table and account_nameinfo_t is the child table.

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 198552 116
0 PIN_FLD_CREATED_T      TSTAMP [0] (1262332886) Fri Jan  1 00:01:26 2010
0 PIN_FLD_MOD_T          TSTAMP [0] (1545044650) Mon Dec 17 03:04:10
2018
0 PIN_FLD_READ_ACCESS      STR [0] "L"
0 PIN_FLD_WRITE_ACCESS     STR [0] "L"
0 PIN_FLD_AAC_ACCESS       STR [0] ""
0 PIN_FLD_AAC_VENDOR       STR [0] ""
0 PIN_FLD_ACCESS_CODE1     STR [0] ""
0 PIN_FLD_ACCESS_CODE2     STR [0] ""
0 PIN_FLD_ACCOUNT_NO       STR [0] "0.0.0.1-198552"
0 PIN_FLD_NAMEINFO       ARRAY [1] allocated 20, used 19
1    PIN_FLD_ADDRESS         STR [0] "m"
1    PIN_FLD_CANON_COMPANY   STR [0] ""
1    PIN_FLD_CANON_COUNTRY   STR [0] "IN"
1    PIN_FLD_CITY            STR [0] "m"
1    PIN_FLD_COMPANY         STR [0] ""
1    PIN_FLD_CONTACT_TYPE    STR [0] "Account holder"
1    PIN_FLD_COUNTRY         STR [0] "in"
1    PIN_FLD_EMAIL_ADDR      STR [0] "m@m.in"
```

BRM Storable Classes

Important Points About Storable classes

1. Storable classes can have no or more than one child/extended class.
2. All parent storable class have POID_id0 field which is the primary key to identify the object.
3. Subclasses are linked to parent/base classes in the table using obj_id0 field.
4. **obj_id0** field is mandatory in all subclasses. This field is linked with parent class **poid_id0** field.
5. Different storable classes are linked by having primary key of one storable class as foreign key in other parent classes.

   Example:

   /account is a storable class and **account_t** is the parent table for the storable class.

   poid_id0 of account_t is primary key.

   When an account is linked with other objects like **/bill** , table name **bill_t,** it will have **account_obj_id0** as foreign key.
6. **Sub tables will not have poid_id0.**
7. foreign key names will be **storableclassname_obj_id0.**

PCM & PIN MACROS

PCM Library

– Set of macros used across the Tiers

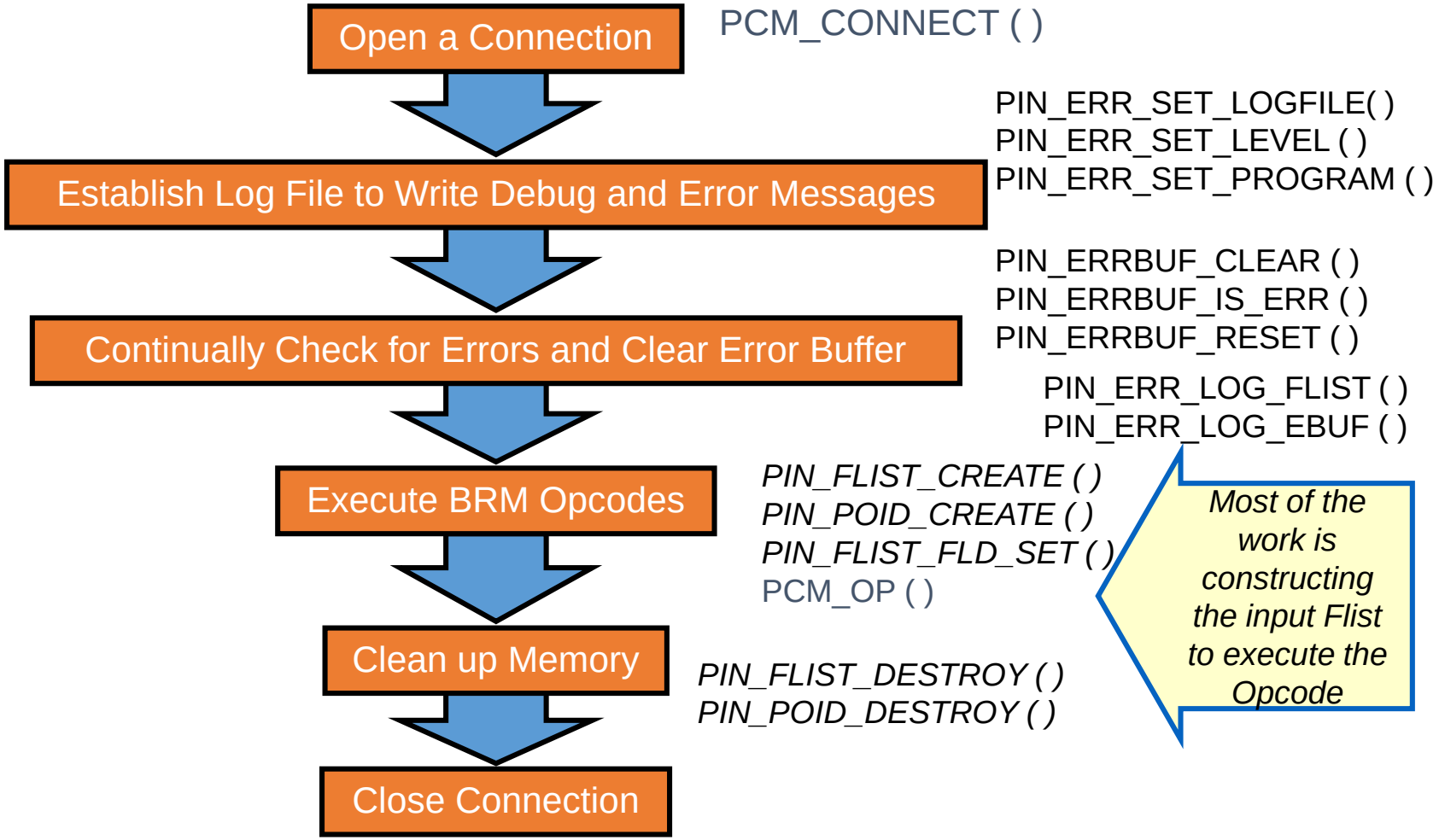
- Manage connections
- Execute Opcodes

PIN Library

– Set of macros and functions used locally within a Tier

- Memory Management and Manipulation of Flists
- Field Manipulation of Flists
- Error Management
- Memory Management and Manipulation of POIDs

PCM and PIN MACROS



PCM MACROS

Context Management PCM Macros

- **PCM_CONTEXT_OPEN( ) opens communication channel between Business Process Tier and Object Tier**
  - Lower level macro
  - Input parameters contain connection info
  - Returns context pointer

- **PCM_CONNECT( ) opens communication channel between Application Tier and Business Process Tier**
  - High level macro that reads connection info in pin.conf file
  - Ultimately executes PCM_CONTEXT_OPEN
  - Returns context pointer & database number.

  Example: PCM_CONNECT(&ctxp, &db_no, &ebuf);

- **PCM_CONTEXT_CLOSE( ) closes communication channel between *any* tiers**
  - Used regardless of how connection was opened
  - Input parameter contains context pointer of connection to close

  Example: PCM_CONTEXT_CLOSE(ctxp, 0, &ebuf);

PCM MACROS

PCM_OP( ) executes Opcodes

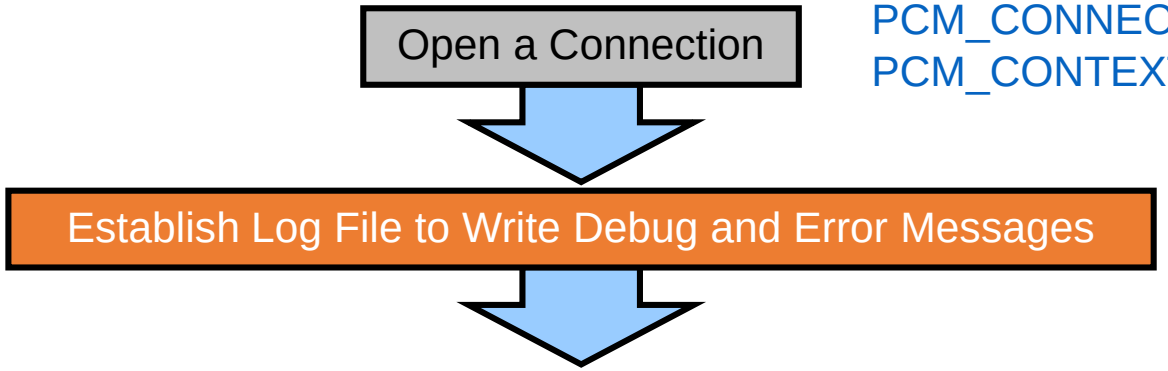
- Used for all Opcodes: FM Opcodes and Base Opcodes.
- Input parameter includes context pointer and flist pointer containing input data.

Example: PCM_OP(ctxp, PCM_OP_CREATE_OBJ, 0, input_flistp, &ret_flistp, &ebuf);

```
PCM_OP(ctxp, opcode, flags, input_flistp, output_flistpp, ebuf )
```

ctxp	Context pointer from opening connection
opcode	Operation to perform; FM, Base or custom opcode
flags	Optional flags per Opcode specification
input_flistp	Pointer to input data per Opcode specification
output_flistpp	Address of pointer to return data
ebuf	Pointer to error buffer

PCM MACROS



PCM_CONNECT ( )
PCM_CONTEXT_OPEN( )

```
pcm_context_t   *ctxp = NULL;
int64           db_no;
pin_errbuf_t    ebuf;

/*
 * Open the connection
 */
PCM_CONNECT(&ctxp, &db_no, &ebuf);
```

1. Follow syntax in Online Docs

2. Declare variables

PCM MACROS - Using the Database Number

```
pcm_context_t *ctxp = NULL;
int64            db_no;
pin_errbuf_t     ebuf;
poid_t           *service_pdp = NULL;

/*
 * Open the connection
 */
PCM_CONNECT(&ctxp, &db_no, &ebuf);

/*
 * Create the poid
 */
service_pdp = PIN_POID_CREATE(db_no, "/service/ip", -1, &ebuf);
```

- PCM_CONNECT returns the database number of the BRM database
  - It is typically used when creating POIDs

PIN MACROS – Creating Flist

```
pin_flist_t     *input_flistp = NULL;
pin_errbuf_t    ebuf;

/*
 * Create the Flist
 */
input_flistp = PIN_FLIST_CREATE(&ebuf);
```

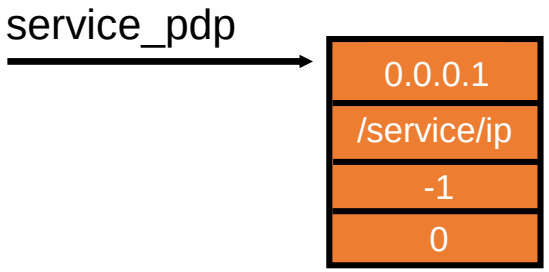
- PIN_FLIST_CREATE:
  - Preallocates 20 field/value pair locations
  - Dynamically allocates more memory as needed
  - Unallocates unused memory
  - Returns a pointer to a “NULL” Flist

PIN MACROS – Creating POID

```
pin_flist_t     *input_flistp = NULL;
pin_errbuf_t    ebuf;
poid_t          *service_pdp = NULL;
int64           db_no;

/*
 * Create the Flist
 */
input_flistp = PIN_FLIST_CREATE(&ebuf);

/*
 * Create the poid and put on Flist
 */
service_pdp = PIN_POID_CREATE(db_no,
          "/service/ip", -1, &ebuf);
```



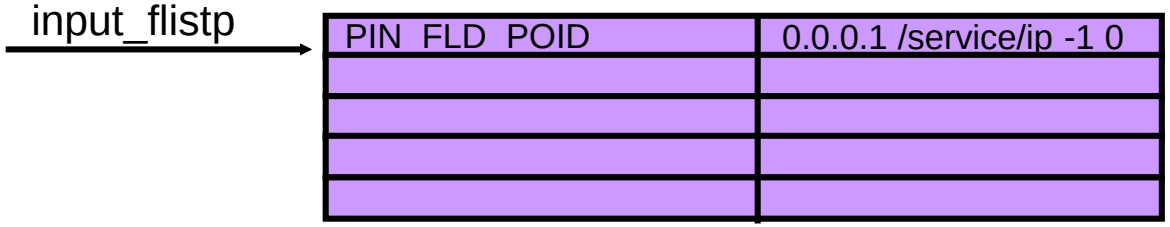
- `PIN_POID_CREATE` allocates memory
  - 4 memory locations to store database #, object type, object ID, and rev level
- Spec requires partial POID to indicate the type of object to create
  - Database # and object type must be valid values
  - Object ID is ignored, so use dummy value (-1 indicates the system should generate the object ID)

PIN MACROS – Set POID on Flist

```
pin_flist_t       *input_flistp = NULL;
pin_errbuf_t      ebuf;
poid_t            *service_pdp = NULL;
int64             db_no;

/*
 * Create the Flist
 */
input_flistp = PIN_FLIST_CREATE(&ebuf);

/*
 * Create the poid and set on Flist
 */
service_pdp = PIN_POID_CREATE(db_no,
          "/service/ip", -1, &ebuf);
PIN_FLIST_FLD_SET(input_flistp,
          PIN_FLD_POID, service_pdp, &ebuf);
```



- `PIN_FLIST_FLD_SET` sets field/value pair on flist:
  - Dynamically allocates more memory as needed
  - Copies value to Flist
  - Pointer to poid is still valid
    - Memory must be cleaned up later

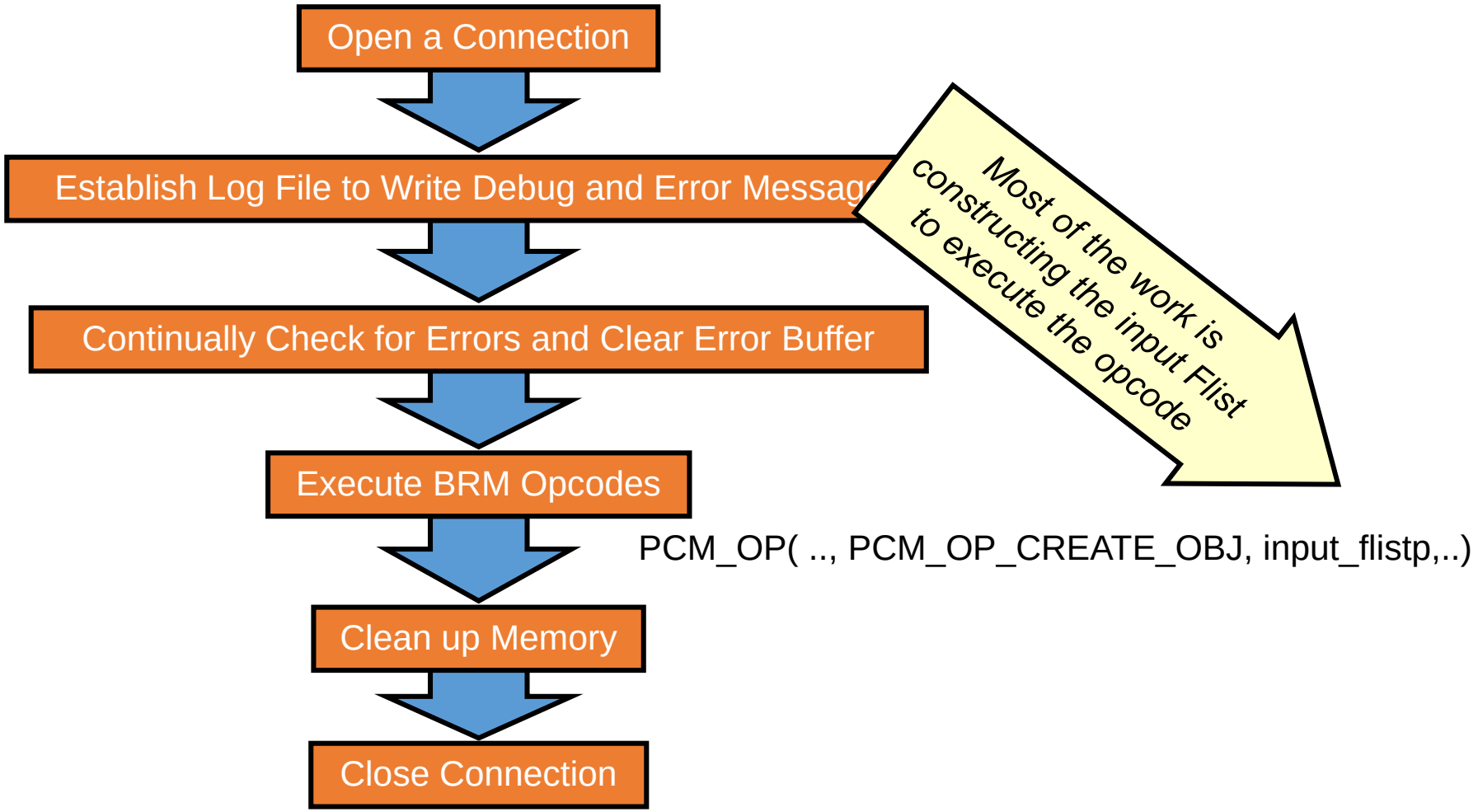
PIN MACROS – Set Base service fields on Flist

```
 /* Create the Flist
 */
input_flistp = PIN_FLIST_CREATE(&ebuf);
 /*
 *Create the poid and set on Flist
 */
service_pdp = PIN_POID_CREATE(db_no, "/service/ip", -1, &ebuf);
PIN_FLIST_FLD_SET(input_flistp, PIN_FLD_POID, service_pdp, &ebuf);
/*
* Set the name field on Flist
 */
PIN_FLIST_FLD_SET(input_flistp, PIN_FLD_NAME, "test object", &ebuf);
/*
 * Create account poid and set on Flist
 */
account_pdp = PIN_POID_CREATE(db_no, "/account", 1, &ebuf);
PIN_FLIST_FLD_SET(input_flistp, PIN_FLD_ACCOUNT_OBJ, account_pdp, &ebuf);
/*
 * Set the login and password on Flist
 */
PIN_FLIST_FLD_SET(input_flistp, PIN_FLD_LOGIN, "test", &ebuf);
PIN_FLIST_FLD_SET(input_flistp, PIN_FLD_PASSWD, "test", &ebuf);
```

input_flistp →

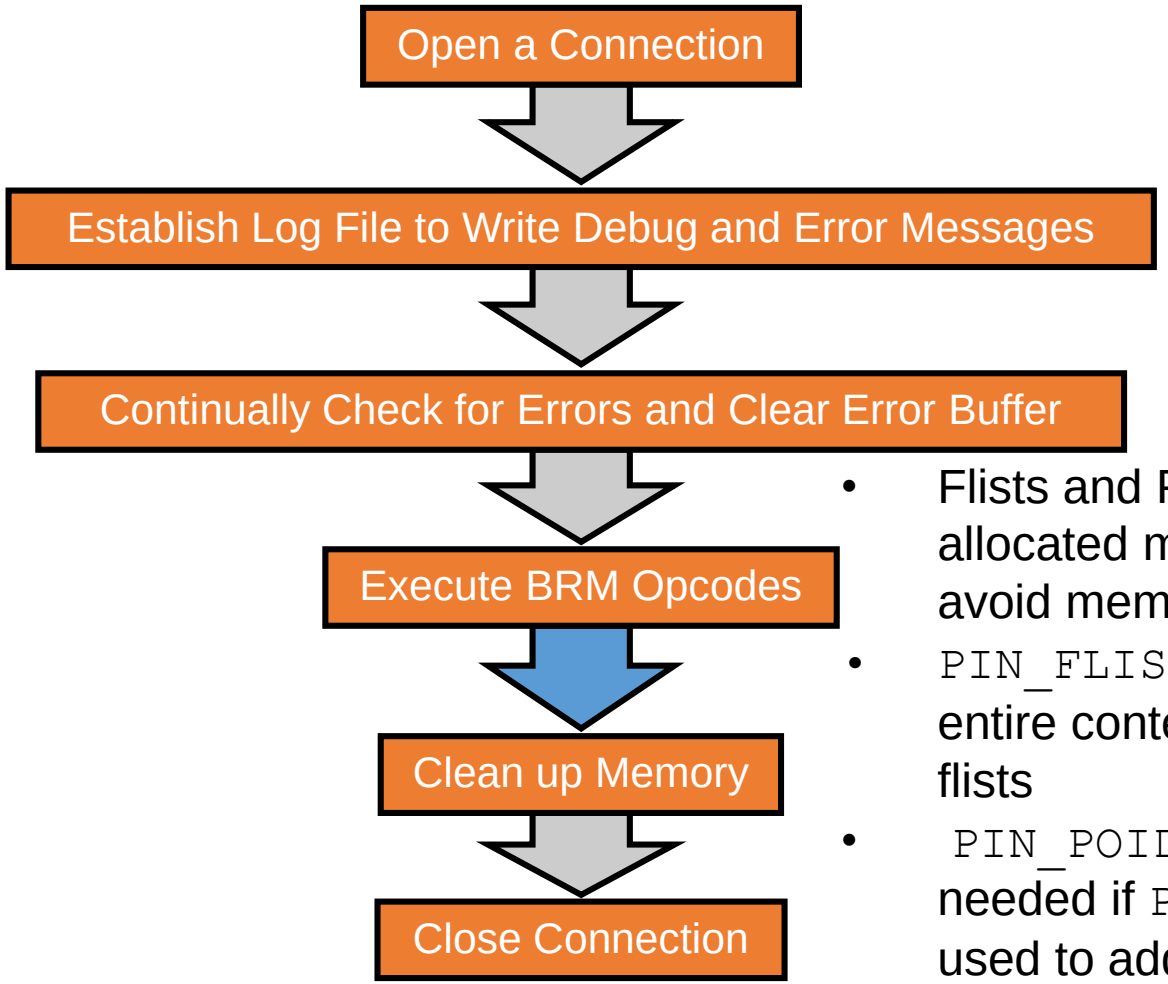
PIN_FLD_POID	0.0.0.1 /service/ip -1 0
PIN_FLD_NAME	"test object"
PIN_FLD_ACCOUNT_OBJ	0.0.0.1 /account 1 0
PIN_FLD_LOGIN	"test"
PIN_FLD_PASSWD	"test"

Input Flist to Opcode



PCM_OP(.., PCM_OP_CREATE_OBJ, input_flistp,..)

PIN Macros for Cleaning up Memory



- Free the flist and POID memory when program is finished

- Flists and POIDS use dynamically allocated memory which must be freed to avoid memory leaks
- `PIN_FLIST_DESTROY( )` Destroys the entire contents of an flist, including nested flists
- `PIN_POID_DESTROY( )` Typically needed if `PIN_FLIST_FLD_SET` was used to add a POID to the flist
  - Destroys the entire POID, including the type string

PIN MACROS

EXAMPLES

```
database = PIN_POID_GET_DB(act_obj);
vp = PIN_POID_CREATE(database,"/search", -1, ebufp);

s_flistp = PIN_FLIST_CREATE(ebufp);
PIN_FLIST_FLD_PUT(s_flistp, PIN_FLD_POID, vp, ebufp);
PIN_FLIST_FLD_SET(s_flistp, PIN_FLD_FLAGS, &srch_flags, ebufp);
PIN_FLIST_FLD_SET(s_flistp, PIN_FLD_TEMPLATE, template, ebufp);
```

```
PIN_ERR_LOG_FLIST(PIN_ERR_LEVEL_DEBUG,
"op_ee_corp_pymt_pol_collect: PCM_OP_EE_CORP_PYMT_POL_GET_CHILDREN i/p flist", i_flistp);

    PCM_OP(ctxp, PCM_OP_EE_CORP_PYMT_POL_GET_CHILDREN, 0, i_flistp, &mem_flistp, ebufp);

    if (PIN_ERR_IS_ERR(ebufp)) {
        PIN_ERR_LOG_EBUF(PIN_ERR_LEVEL_ERROR,
                         "Error in calling PCM_OP_EE_CORP_PYMT_POL_GET_CHILDREN", ebufp);
        fm_ee_corp_pymt_pol_collect_set_error(ctxp, i_flistp, r_flistpp, ret_code, ebufp);
        PIN_FLIST_DESTROY_EX(&mem_flistp, NULL);
        PIN_ERR_CLEAR_ERR(ebufp);
        return;
```

Flist Field Management Macros

PIN_FLIST_FLD_GET – This macro gets the value of a field from an flist.

- returns a copy of the pointer to value on Flist
- Flist retains ownership of memory
- No additional memory is allocated
- Value must be treated as read-only

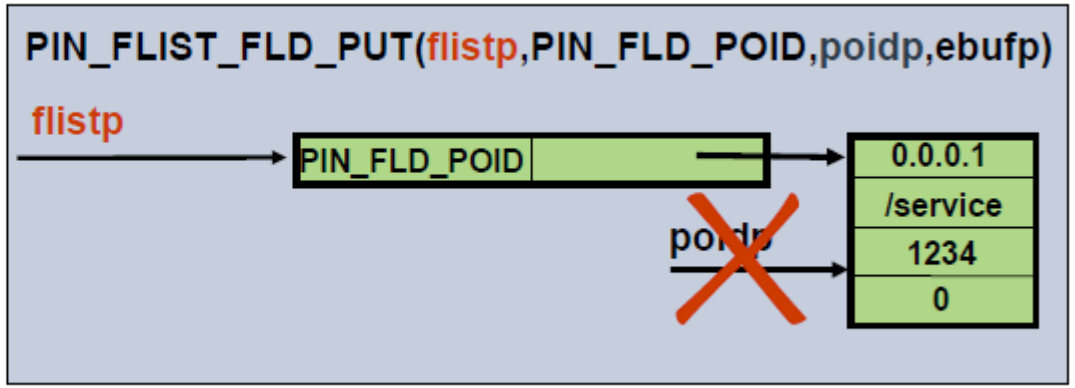
Example:

```
poid_t *poidp = NULL;
poidp=PIN_FLIST_FLD_GET(flistp,PIN_FLD_POID,0,ebufp);
```

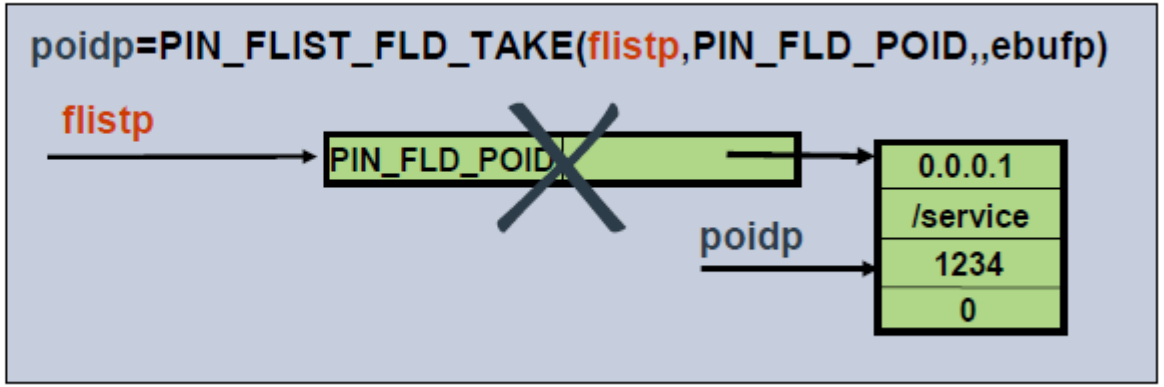
The arguments to the macro are:
flistp pointer from which value to be read.
The field : PIN_FLD_POID.
Flag value: If this flag is set (by passing in a non-0 value) and the element is not found, no error condition is set. If this flag is not set and the element is not found, an error condition is set.
Error buffer pointer: ebufp

Flist Field Management Macros

- PIN_FLIST_FLD_PUT puts memory containing value on Flist
  - Transfers ownership of poid memory to Flist
  - POID pointer no longer valid
  - Memory will be cleaned up when Flist is destroyed



- PIN_FLIST_FLD_TAKE returns pointer to value on Flist
  - Transfers ownership of Flist memory to pointer
  - Field/value pair no longer exists on Flist
  - Memory must be cleaned up later



NOTE: PIN_FLIST_FLD_PUT & PIN_FLIST_FLD_TAKE uses Original pointer variables just like call by reference in C language.

Flist Field Management Macros

Set vs Put vs Add Macros

	Simple field	Substruct	Array
Copy pointer to value that exists	PIN_FLIST_FLD_SET( )	PIN_FLIST_SUBSTR_SET( )	PIN_FLIST_ELEM_SET( )
Transfer existing pointer to value	PIN_FLIST_FLD_PUT( )	PIN_FLIST_SUBSTR_PUT( )	PIN_FLIST_ELEM_PUT( )
Nested Flist doesn't exist	-	PIN_FLIST_SUBSTR_ADD( )	PIN_FLIST_ELEM_ADD( )

- Set: may need to explicitly destroy memory later
- Put: eliminates need to destroy memory later

Flist Field Management Macros

Get vs Take Macros

	Simple field	Substruct	Array
Returns pointer to value on Flist	PIN_FLIST_FLD_GET( )	PIN_FLIST_SUBSTR_GET( )	PIN_FLIST_ELEM_GET( )
Returns pointer to value removed from Flist	PIN_FLIST_FLD_TAKE( )	PIN_FLIST_SUBSTR_TAKE( )	PIN_FLIST_ELEM_TAKE( )

- Get: never modify value or destroy memory
- Take: may need to destroy memory later

Exercise 4

1. What is the use of poid_type field?
2. On the flist all the parent table fields will be at level?
3. Child tables with rec_id fields represents?
4. What is use of obj_jd0 field & in which table it exists?
5. What is the use of /account storable class?
6. What is the use of /balance_group storable class?
7. What is the use of /billinfo storable class?
8. Which context management macro is used to open a connection between an Portal client application and the CM?
9. Which context management macro opens a connection between an FM and a DM?
10. If PCM_CONNECT is used to open a connection, which context management macro is used to close the connection?
11. What six parameters are needed to execute any Portal Opcode?

“Thank You”