Oracle BRM Training

By Nedhya Sharma

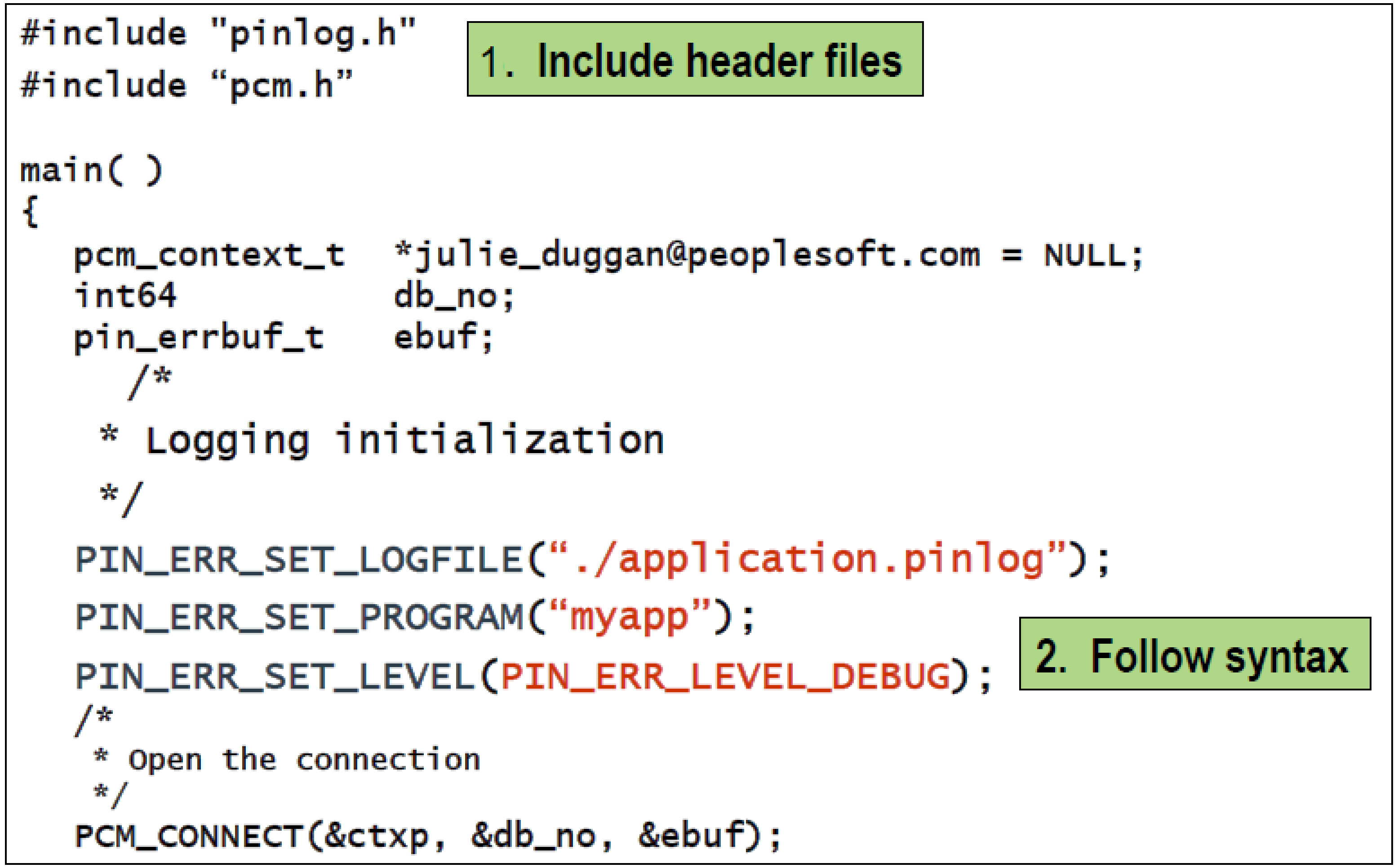
AGENDA

1. **Error Handling & Debugging**

How to Set log files & log level programmatically

- **PIN_ERR_SET_LOGFILE( ) Sets path and filename of pinlog file**
  - Often specified in pin.conf file
- **PIN_ERR_SET_PROGRAM( ) Sets program name to insert in pinlog files**
- **PIN_ERR_SET_LEVEL( ) Sets logging level**
  - PIN_ERR_LEVEL_ERROR = 1
  - PIN_ERR_LEVEL_WARNING = 2
  - PIN_ERR_LEVEL_DEBUG = 3

How to Set log files & log level programmatically



```
#include "pinlog.h"
#include “pcm.h”

main( )
{
   pcm_context_t  *julie_duggan@peoplesoft.com = NULL;
   int64          db_no;
   pin_errbuf_t   ebuf;
      /*
    * Logging initialization
    */
   PIN_ERR_SET_LOGFILE(“./application.pinlog”);
   PIN_ERR_SET_PROGRAM(“myapp”);
   PIN_ERR_SET_LEVEL(PIN_ERR_LEVEL_DEBUG);
   /*
    * Open the connection
    */
   PCM_CONNECT(&ctxp, &db_no, &ebuf);
```

Understanding the Error Buffer

Error buffers are data structures that hold API error information.

- PCM macros use individual style error buffers.
  - Error information overwrites current effort buffer contents
  - Check the error buffer after each PCM macro opcode execution.

- PIN macros use series style error buffers.
  - Error information is appended to current error buffer contents.
  - Check the error buffer at the end of series of PIN macros execution.

```
typedef struct {
    int32 location;             // Specifies the Portal module that encountered error
    int32 pin_errclass;         // Describes the class of error that occurred
    int32 pin_err;              // Describes the exact error that was encountered
    pin_fld_num_t field;        // Field number of input parameter that caused error
    int32 rec_id;               // Element ID of array element that caused error
    int32 reserved;             // Internal system state for Portal debugging
    int32 line_no;              // Line number from source file where error detected
    char *filename;             // Name of source file where error detected
    int facility;               // Designates facility code
    int msg_id;                 // Designates unique ID for each message
    int err_time_sec;           // Output time in seconds when error occurred
    int err_time_usec;          // Output time in microseconds when error occurred
    int version;                // Designates version of arguments
    pin_flist_t *argsp;         // Used as optional arguments flist
    pin_errbuf_t *nextp;        // Used as one or more optional chained errbufs
    int reserved2               // Reserved for internal use
} pin_errbuf_t;
```

PIN Macros to Manage Error Buffer

- **PIN_ERRBUF_CLEAR( ) Initializes error buffer**
  - Always clear error buffer before first use
- **PIN_ERRBUF_IS_ERR( ) Check for errors**
  - Use immediately after PCM macro
  - Use after series of related PIN macros
- **PIN_ERRBUF_RESET( ) Resets error buffer after detecting an error**
  - Always reset error buffer before reusing
  - Frees up extra memory from series style error buffer

PIN Macros to Manage Error Buffer

Series Style Error

```
/* Construct the flist
 */
input_flistp = PIN_FLIST_CREATE(&ebuf);
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(input_flistp, PIN_FLD_NAME, “test object”, &ebuf);
account_pdp = PIN_POID_CREATE(db_no, “/account”, 1, &ebuf);
PIN_FLIST_FLD_SET(input_flistp, PIN_FLD_ACCOUNT_OBJ, account_pdp, &ebuf);
/*
 * Check for errors
 */
if (PIN_ERRBUF_IS_ERR(&ebuf)) {
      return;
}
```

INDIVIDUAL STYLE ERROR

```
pin_errbuf_t      ebuf;

/*
 * Initialize error buffer
 */
PIN_ERRBUF_CLEAR(&ebuf);
/*
 * Open the connection
 */
PCM_CONNECT(&ctxp, &db_no, &ebuf);
/*
 * Check for errors
 */
if (PIN_ERRBUF_IS_ERR(&ebuf)) {
      return;
}
```

PIN Macros to log information

- **PIN_ERR_LOG_MSG( ) Writes a string message to the pinlog file.**

  Example: PIN_ERR_LOG_MSG(PIN_ERR_LEVEL_DEBUG,“anymessagetoidentifyinlogfile");

- **PIN_ERR_LOG_FLIST( ) Writes a string message and dumps the contents of an flist to the pinlog file.**

  – Nested flists automatically included
  – Typically used for debugging

  Example: PIN_ERR_LOG_FLIST(PIN_ERR_LEVEL_DEBUG, “input flist”, input_flistp);

PIN Macros to log information

- **PIN_ERR_LOG_EBUF( ) Writes a string message and dumps the contents of an error buffer to the pinlog file**
  - – Typically used when an error is detected

  Example: **PIN_ERR_LOG_EBUF(PIN_ERR_LEVEL_ERROR**, **“error constructing input flist”,** &ebuf);

- **PIN_ERR_LOG_POID( ) Writes a string message and dumps the contents of a poid to the pinlog file**
  - – Typically used for debugging

  Example: **PIN_ERR_LOG_POID(PIN_ERR_LEVEL_DEBUG**, “cloudshine-acctpdp", acct_pdp);

Example: Clearing a Populated Error Buffer

```
/* Clean the Error Buffer              */
     PIN_ERRBUF_RESET(&ebuf);
/*   PCM operations                     */
/*   Call the COMMIT CUSTOMER opcode    */
     PCM_OP(ctxp, PCM_OP_CUST_COMMIT_CUSTOMER, 0, flistp, &r_flistp, &ebuf);
/*   Check for errors                   */
         if (PIN_ERRBUFF_IS_ERR(&ebuf)) {
                PIN_ERR_LOG_EBUF(PIN_ERR_LEVEL_ERROR,
                        "create_customer error"", &ebuf);
                        PIN_ERRBUF_RESET(&ebuf);
                return;
            }
```

To avoid memory leaks and to reuse the error buffer, use the PIN_ERRBUF_RESET macro

Troubleshooting Errors in BRM

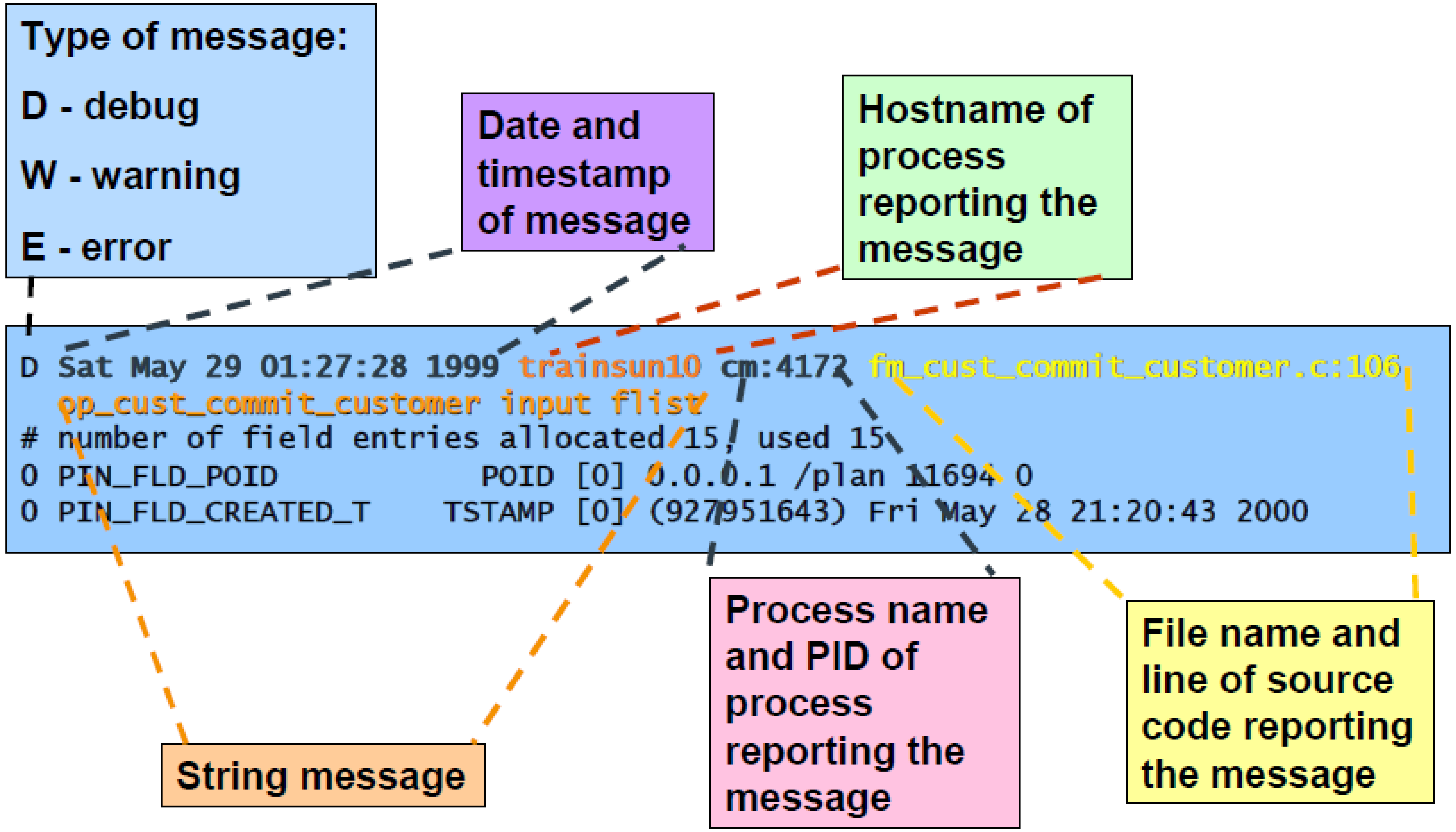
- **Debugging Process Overview:**

  – Step 1: Open the appropriate message file
  – Step 2: Go to the most recent messages at the end of the file
  – Step 3: Search for the first instance of an error message
  – Step 4: LOCATION – note the location first
  – Step 5: Search the message file specific to the location of the error
  – Step 6: Retrieve the context from the string message, the Opcode being executed, and the data on the flist.

Common Causes of Errors Based on Location

- PIN_ERRLOC_APP Problem is generated by application passing bad or invalid data. Check data on input flist to make sure any referenced poids exist in database, or values will pass any validation policies.
- PIN_ERRLOC_FLIST Problem constructing an flist. Value doesn't match data type or memory allocation problem.
- PIN_ERRLOC_POID Problem constructing a poid. Value doesn't match data type or memory allocation problem.
- PIN_ERRLOC_PCM or PIN_ERRLOC_PCP The most common cause is that one tier is trying to connect to another that is not running.
  - If the client app cannot connect, check that CM is running. If CM cannot connect, check that DM is running. Another possible cause is inconsistency between pin.conf files. Check that correct hostname and port numbers are specified in pin.conf files.
  - Check that network is functional between hosts.
- PIN_ERRLOC_CM CM didn't know how to handle a PCM_OP request. Check that opcode is valid and defined in fm_*.so. Check that PIN_FLD_POID exists on flist.
- PIN_ERRLOC_FM Problem in an fm attempting to implement an opcode. most common cause is data doesn't match flist spec for the opcode. check that input flist data matches spec - mandatory fields, data types, etc.
- PIN_ERRLOC_DM Most common problem is in creating an object. check that data on input flist matches object spec for the object you want to create.

Parsing a Pinlog Message

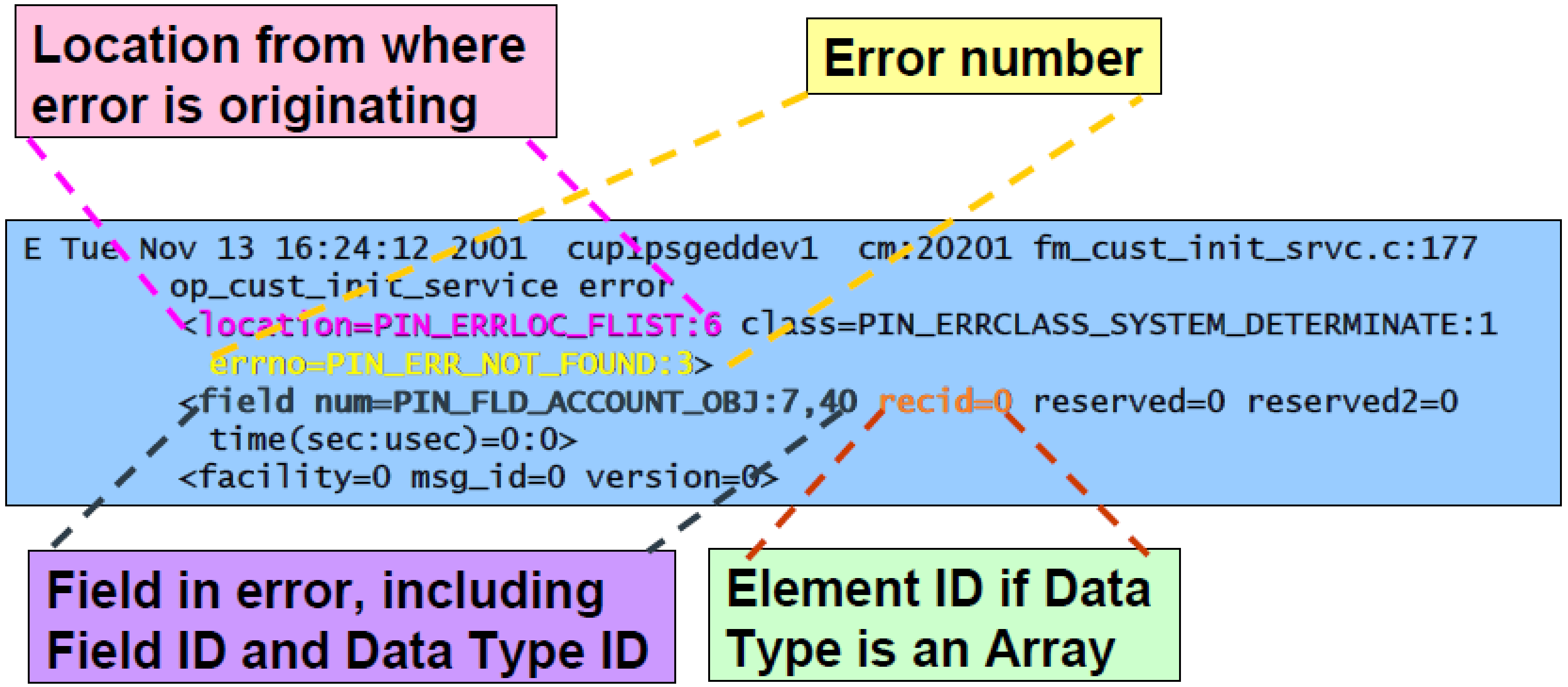


```
D Sat May 29 01:27:28 1999 trainsun10 cm:4172 fm_cust_commit_customer.c:106
  op_cust_commit_customer input flist
# number of field entries allocated 15, used 15
0 PIN_FLD_POID           POID [0] 0.0.0.1 /plan 11694 0
0 PIN_FLD_CREATED_T    TSTAMP [0] (927951643) Fri May 28 21:20:43 2000
```

Type of message:
D - debug
W - warning
E - error
Date and timestamp of message
Hostname of process reporting the message
String message
Process name and PID of process reporting the message
File name and line of source code reporting the message

PUBLIC

Parsing the Error Buffer


Location from where error is originating
Error number
Field in error, including Field ID and Data Type ID
Element ID if Data Type is an Array

```
E Tue Nov 13 16:24:12 2001   cup1psgeddev1  cm:20201 fm_cust_init_srvc.c:177
        op_cust_init_service error
        <location=PIN_ERRLOC_FLIST:6 class=PIN_ERRCLASS_SYSTEM_DETERMINATE:1
          errno=PIN_ERR_NOT_FOUND:3>
        <field num=PIN_FLD_ACCOUNT_OBJ:7,40 recid=0 reserved=0 reserved2=0
          time(sec:usec)=0:0>
        <facility=0 msg_id=0 version=0>
```

“Thank You”