

Thinking Outside the Developer's Toolbox: Application Program Recovery and Other Cool Tricks



Charles Guarino

Central Park Data Systems, Inc.

1

About The Presenter

With an IT career spanning over 30 years, Charles Guarino has been a consultant for most of them. Since 1995 he has been founder and President of Central Park Data Systems, Inc., a New York area based IBM midrange consulting company. He has published several technical and strategic articles for the IT community and continues in that role as a frequent contributor of programming and developer tips to System i Network and Power IT Pro. He is a proud member of COMMON's Speaker Excellence Hall of Fame and also Long Island Software and Technology Network's Twenty Top Techies of 2009. Charles currently serves as a COMMON Application Development Subject Matter Expert and President and monthly Q&A host of LISUG, a Long Island IBM System i User's Group www.lisug.org. Charles can be reached at cguarino@centralparkdata.com. LinkedIn - <http://www.linkedin.com/in/guarinocharles>

What We'll Cover ...

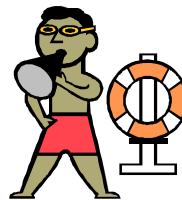
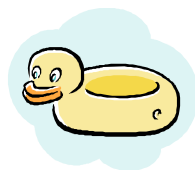
- Preventative measures to help prevent database corruption
- Techniques to prepare your database for its close-up under the microscope
- Tricks to improve your development and testing
- Resolving some everyday emergencies
- Wrap-up

Copyright Central Park Data Systems, Inc.

3

S.O.B.! *

- A sleeper program that parses XML data runs in QBATCH all day.
- Users are inadvertently ending the job.
- Once the job is ended no future XML documents are being parsed.



* Save our batch jobs!

Copyright Central Park Data Systems, Inc.

4

S.O.B.! (cont.)

- We need a way to prevent option '4' from being executed on batch job 'SAXPARSER'.
- Better yet, if the job gets ended, it will **RESTART AUTOMATICALLY!!!**
- Introducing the indestructible sleeper program!

```

CPU %:      .3      Elapsed time:  00:00:24      Active jobs:  263

Type options, press Enter.
2=Change  3=Hold  4=End  5=Work with  6=Release  7=Display message
8=Work with spooled files 13=Disconnect ...

Opt Subsystem/Job User      Type CPU % Function      Status
---
SAXPARSER CHARLIE BCH      .0 PGM-SAXPARSECL EOFW

```

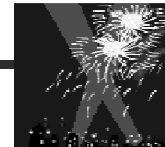


Copyright Central Park Data Systems, Inc.

5

Enter the PRESTART JOB!

- No matter how many times this job is ended it will automatically restart itself.
- The job is defined as part of a subsystem.
- As long as the subsystem is active this job will also be active unless it is explicitly ended.



```

CPU %:      3.4      Elapsed time:  00:00:01      Active jobs:  264

Type options, press Enter.
2=Change  3=Hold  4=End  5=Work with  6=Release  7=Display message
8=Work with spooled files 13=Disconnect ...

Opt Subsystem/Job User      Type CPU % Function      Status
---
SAXPARSER QUSER PJ      .0      EOFW

```

Copyright Central Park Data Systems, Inc.

6

The Prestart Job in Action

- Watch the Job Number!
- When the job is ended using option 4 it will be automatically re-submitted by the subsystem.

```

Work with Job
Job:  SAXPARSER      User:  QUSER      Number:  940356      System:  CPDS2

```

```

End Job (ENDJOB)
Type choices, press Enter:
Job name . . . . . > SAXPARSER      Name, *
User . . . . . > QUSER              Name
Number . . . . . > 940356           000000-999999
How to end . . . . . *immed         *CNTRLD, *IMMED
Controlled end delay time . . . . . 30 Seconds

```

Opt	Subsystem/Job	User	Type	CPU %	Function	Status
	QBATCH	QSYS	SBS	.0		DEQW
	SAXPARSER	QUSER	PJ	.0		EOFW

```

Work with Job
Job:  SAXPARSER      User:  QUSER      Number:  940360      System:  CPDS2

```

Copyright Central Park Data Systems, Inc.

7

How to Setup a Prestart Job

- Setup a basic job description.
- Make sure it contains all libraries required by the job.

```

Display Job Description
Job description:  XMLJOBDB      Library:  XMLLIB      System:  CPDS2
User profile . . . . . : *RQD
CL syntax check . . . . . : *NOCHK
Hold on job queue . . . . . : *NO
End severity . . . . . : 30
Job date . . . . . : *SYSVAL
Job switches . . . . . : 00000000
Inquiry message reply . . . . . : *RQD
Job priority (on job queue) . . . . . : 5
Job queue . . . . . : QBATCH
Library . . . . . : QGPL
Output priority (on output queue) . . . . . : 5
Printer device . . . . . : *USRPRF
Output queue . . . . . : *USRPRF
Library . . . . . :

```

```

Display Job Description
Job description:  XMLJOBDB      Library:  XMLLIB      System:  CPDS2
Initial library list:
Sequence number  Library
10              QGPL
20              QTEMP
30              XMLLIB

```

Copyright Central Park Data Systems, Inc.

8

How to Setup a Prestart Job (cont.)

- Adding the prestart job ensures the job will always be running, as long as the subsystem itself is active.

```

Add Prestart Job Entry (ADDPJE)
Type choices, press Enter.

Subsystem description . . . . . > QBATCH      Name
Library . . . . . > *LIBL      Name, *LIBL, *CURLIB
Program . . . . . > SAXPARSECL  Name
Library . . . . . > XMLLIB      Name, *LIBL, *CURLIB
User profile . . . . . > QUSER      Name
Start jobs . . . . . > *YES      *YES, *NO
Initial number of jobs . . . . . > 1      1-9999
Threshold . . . . . > 1      1-9999
Additional number of jobs . . . . . > 0      0-999
Maximum number of jobs . . . . . > 1      1-9999, *NOMAX

Additional Parameters

Job name . . . . . > SAXPARSER      Name, *PGM
Job description . . . . . > XMLJOB      Name, *USRPRF, *SBSD
Library . . . . . > XMLLIB      Name, *LIBL, *CURLIB

```

Copyright Central Park Data Systems, Inc.

9

How to Setup a Prestart Job (cont.)

- Here is the prestart job once it is successfully created.

```

Selection or command
==> wrksbsd qbatch

```

```

Display Subsystem Description
Subsystem description: QBATCH      Library: QSYS      System: CPDS2
Status: ACTIVE

Select one of the following:
1. Operational attributes
2. Pool definitions
3. Autostart job entries
4. Work station name entries
5. Work station type entries
6. Job queue entries
7. Routing entries
8. Communications entries
9. Remote location name entries
10. Prestart job entries

```

```

Display Prestart Job Entry Detail
Subsystem description: QBATCH      Status: ACTIVE      System: CPDS2

Program . . . . . : SAXPARSECL
Library . . . . . : XMLLIB
User profile . . . . . : QUSER
Job . . . . . : SAXPARSER
Job description . . . . . : XMLJOB
Library . . . . . : XMLLIB
Start jobs . . . . . : *YES
Initial number of jobs . . . . . : 1

```

Copyright Central Park Data Systems, Inc.

10

Give Your Database Built-In Auditing!

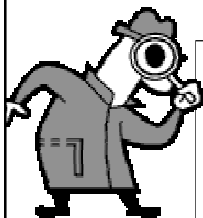
- The system automatically maintains object level auditing
- You can use command DSPOBJD to retrieve information such as date created and last date used.
 - In many cases this is not granular enough.
 - Having record level auditing is an important component when trying to re-create an update process if it ends abnormally.
 - Having *ISO date and time fields allows us to interrogate any file or group of files and write the appropriate fix program(s).

Copyright Central Park Data Systems, Inc.

11

Give Your Database Built-in Auditing (cont.)

- Provides the who, what and when of “touching” records.



A	IDTCRT	L	COLHDG('Date Created')
A			DFT('0001-01-01')
A	ITMCRT	T	COLHDG('Time Created')
A	CRTPGM	10	COLHDG('Created By Program')
A	CRTUSR	10	COLHDG('Created By User')
A	IDTCHG	L	COLHDG('Date Changed')
A			DFT('0001-01-01')
A	ITMCHG	T	COLHDG('Time Created')
A	CHGPGM	10	COLHDG('Changed By Program')
A	CHGUSR	10	COLHDG('Changed By User')

Added
To the
End of
The
Record

CPDS-DEVLB/ITEMMAST(ITEMMAST)													
ITEMNO	ITEMDS1	YIDS1	KTDS...	YIDS...	YIDS...	IDTCRT	ITMCRT	CRTPGM	CRTUSR	IDTCHG	ITMCHG	CHGPGM	CHGUSR
BOX2	CORRUGATED BOX	45,828	482	82	8,141	2010-04-21	15:24:37	DAILYBLD	CHARLIE	0001-01-01	00:00:00		
LENS8	LENS CLEANER 8-PAK	5,821	85,921	458,2...		2010-04-25	15:25:33	ITEMMNT	ALEXANDRA	2010-04-25	15:32:18	QOZXDBI	CHARLIE
NAILSR	ROOFING NAILS	482	58,458	98,482		2010-04-22	15:24:14	ITEMMNT	CHARLIE	2010-04-25	15:32:58	QOZXDBI	SAMANTHIA
PK1A	SMALL PICTURE FRAME	5,821	5,811	6,885	-18,381	2010-04-25	15:25:10	QOZXDBI	CHARLIE	2010-04-25	15:33:02	QOZXDBI	SAMANTHIA
012X06X12	1/2 INCH PLYWOOD	222	500	600		2010-04-25	15:23:00	ITEMMNT	CHARLIE	2010-04-25	15:33:05	QOZXDBI	ALEXANDRA
3032	HAMMER	4,566	3,659	94,873	43,820	2010-04-25	15:23:49	EDITTCM	ADAM	0001-01-01	00:00:00		

Copyright Central Park Data Systems, Inc.

12

Introducing the Trigger Program



SENDPGMMMSG
RECVPGMMMSG

newfile.chgpgm = user...

CHGPGM = ADAM
IDTCHG = 2010.04.25
...

Program issues an
UPDATE to the
database

Trigger program
ITEMTRG is fired,
passes both record
images and other
DB information

Trigger program
generates harmless
program message
and reads
program stack

Determines trigger event,
populates "change" fields
in buffer and returns

Record is updated
in the database
with changed fields

Copyright Central Park Data Systems, Inc.

13

Trigger Program ITEMTRG

```

ITEMTRG.RPGLE
Line 52      Column 70      Replace
.....1.....2.....3.....4.....5.....6.....7.....8.....9.....
004100      d  MsgDtaLen              101 0 const
004200      d  MsgType                10 const
004300      d  MsgQ                   10 const
004400      d  MsgQNbr              101 0 const
004500      d  MsgKey                 4 const
004600      d  Error                  like(ErrorDS)
004700
004800      ** PROTOTYPE for Receive Program Message from Calling Program.
004900      d  RecvPgmmMsg      pr          extpgm('QMRCVPM')
005000      d  MsgInfo          176 const
005100      d  MsgInfoLen       101 0 const
005200      d  FmtName           8 const
005300      d  MsgQ              10 const
005400      d  MsgQNbr          101 0 const
005500      d  MsgType           10 const
005600      d  MsgKey            4 const
005700      d  WaitTime         101 0 const
005800      d  MsgAction        10 const
005900      d  Error             like(ErrorDS)
006000
006100      d  Main              pr          extpgm('ITEMTRG')
006200      d  TrgBuffer         likeds(trgvparm)
006300      d  TrgBufferLen      101 0
006400
006500      d  Main              pi
006600      d  TrqBuffer         likeds(trqparm)
006700      d  TrqBufferLen      101 0
006800
006900      ** Pointer based data structures for the input buffer.
007000      d  NewPoint          s          *
007100      d  OldPoint          s          *
007200
007300      D  ErrorDS           ds          16
007400      D  BytesProv         101 0 inz(16)
007500      D  BytesAvail         101 0
007600      D  ExceptionID        7
007900

```

Copyright Central Park Data Systems, Inc.

14

Adding a trigger to your database file

- Repeat this process for EVENT *INSERT
- AN EXCLUSIVE NO-READ LOCK IS REQUIRED
- Use command RMVPFTRG to remove any or *ALL triggers!
- Up to 300 triggers can be added to one file

```

Add Physical File Trigger (ADDPFTRG)

Type choices, press Enter.

Physical file . . . . . > ITEMMAST      Name
Library . . . . . > DEVLIB           Name, *LIBL, *CURLIB
Trigger time . . . . . > *BEFORE       *BEFORE, *AFTER
Trigger event . . . . . > *UPDATE      *INSERT, *DELETE, *UPDATE...
Program . . . . . > ITEMTRG          Name
Library . . . . . > DEVLIB           Name, *LIBL, *CURLIB
Replace trigger . . . . . > *NO        *NO, *YES
Trigger . . . . . > *GEN

Trigger library . . . . . > *FILE      Name, *FILE, *CURLIB
Allow Repeated Change . . . . . > *YES  *NO, *YES

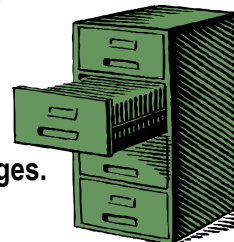
```

Copyright Central Park Data Systems, Inc.

15

Tracking Changes to an Entire Database

- Journaling 101:
- With the ability to capture both BEFORE and AFTER images of each record along with program information, a JOURNAL is one of the most powerful tools in your programming arsenal.
 - You'll need both images to compare so you can track changes.
- Consider the journal a big filing cabinet with multiple drawers.
 - Each "drawer" is called a journal receiver.
 - ▶ The receiver must be created before the actual journal.
 - Each receiver can contain multiple files.
- You'll need an *EXCLRD lock to *start and *end journaling a file.
- A journal can capture more than just database changes.
 - Data areas and data queues too!



Copyright Central Park Data Systems, Inc.

16

Creating the receiver and the journal itself

Create Journal Receiver (CRTJRNRCV)

Label:			
Journal receiver:	> JRNRCV	Name	
Library:	> DEVLIB	Name	
ASP number:	> *LIBASP	1-32	
Journal receiver threshold:	> 1500000	1-100000000	
Text 'description':	> *BLANK	Character value	
Advanced Parameters			

CRTJRNRCV JRNRCV(DEVLIB/JRNRCV)

What the command looks like

Create Journal (CRTJRN)

Label:			
Journal:	> JRN	Name	
Library:	> DEVLIB	Name	
Journal receiver:	> JRNRCV	Name	
Library:	> DEVLIB	Add	Name
ASP number:	> *LIBASP	1-32	
Journal message queue:	> QSYSOPR	Name	
Library:	> *ITR	Name	

CRTJRN JRN(DEVLIB/JRN) JRNRCV(DEVLIB/JRNRCV)

What the command looks like

Copyright Central Park Data Systems, Inc.

17

Issuing the Start Journaling Command STRJRNPF

Start Journal Physical File (STRJRNPF)

Label:			
Physical file to be journaled:	> CUSTMAST	Name, generic	
Library:	> DEVLIB	Add	Name
Journal:	> JRN	Name	
Library:	> DEVLIB	Name	
Record images:	> *BOTH		
Journal entries to be omitted:	> *OPNCLO		
Logging level:	> *ERRORS		
Comment:			

```

/*      Start journaling CUSTMAST                                     */
STRJRNPF FILE(DEVLIB/CUSTMAST) JRN(DEVLIB/JRN) +
        IMAGES(*BOTH) OMITRE(*OPNCLO)

/*      Update customer master file CUSTMAST                         */
CALL      PGM(UPDCUS)

CALL      PGM(CUSTMAINT)

/*      End journaling CUSTMAST                                     */
ENDJRNPF FILE(DEVLIB/CUSTMAST) JRN(DEVLIB/JRN)

```

Copyright Central Park Data Systems, Inc.

18

We will perform three different database actions

```

CUSTMAINT.RPGLE X
Line 1      Column 1      Replace
.....1.....2.....3.....4.....5.....6.
000100      // Customer Master File
000200      Ccustmast  uf a e          k disk
000300
000400      /free
000500
000600      // Delete customer number 150
000700      delete ( 01 : 150) custmast; ← DL
000800
000900      // Write a new customer record
001000      clear custmast;
001100      ccomp = 01;
001200      ccusno = 7777777;
001300      ccusnam = 'Jones Super Pizza';
001400      caddr1 = '411 Gold Street';
001500      caddr2 = 'San Diego, CA';
001600      cytdsl = 1292.02;
001700      cytdsl = 9907.09;
001800      write custmast; ← PT
001900
002000      // Change the name of customer 12345
002100      chain ( 01 : 12345) custmast;
002200      if %found(custmast);
002300      ccusnam = 'Smith's Our Customer';
002400      cytdsl = 309.90;
002500      update custmast; ← UB/UP
002600      endif;
002700
002800      *inlr = *on;
          return;

```

Copyright Central Park Data Systems, Inc.

19

Displaying the journal using the DSPJRN command

Display Command String

DSPJRN JRN(JRN)

Display Journal Entries

Journal : JRN Library : DEVLIB
 Largest sequence number on this screen : 0000000000000000017
 Type options, press Enter.
 5=Display entire entry

Opt	Sequence	Code	Type	Object	Library	Job	Time
=	1	J	PR			JRNDEMO	21:38:56
-	2	D	JF	CUSTMAST	DEVLIB	JRNDEMO	21:39:53
-	3	F	JM	CUSTMAST	DEVLIB	JRNDEMO	21:39:53
-	5	R	UB	CUSTMAST	DEVLIB	JRNDEMO	21:39:53
-	6	R	UP	CUSTMAST	DEVLIB	JRNDEMO	21:39:53
-	9	R	DL	CUSTMAST	DEVLIB	JRNDEMO	21:39:53
-	13	R	PT	CUSTMAST	DEVLIB	JRNDEMO	21:39:53
-	14	R	UB	CUSTMAST	DEVLIB	JRNDEMO	21:39:53
-	15	R	UP	CUSTMAST	DEVLIB	JRNDEMO	21:39:53
-	16	F	EJ	CUSTMAST	DEVLIB	JRNDEMO	21:39:53
-	17	D	EF	CUSTMAST	DEVLIB	JRNDEMO	21:39:53

Copyright Central Park Data Systems, Inc.

20

Displaying a “before” image, code UB

Display Journal Entry

Object : CUSTMAST Library : DEVLIB
 Member : CUSTMAST
 Incomplete data . . . : No Minimized entry data : *NONE
 Sequence : 14
 Code : R - Operation on specific record
 Type : UB - Update, before-image

Entry specific data

Column	*...+...1...+...2...+...3...+...4...+...5
00001	'One Great Customer 529 Bestcust D'
00051	'rive
00101	' b ê 0001'
00151	'-01-010001-01-01

Copyright Central Park Data Systems, Inc.

21

The same record, displayed here with the “after” image

Display Journal Entry

Object : CUSTMAST Library : DEVLIB
 Member : CUSTMAST
 Incomplete data . . . : No Minimized entry data : *NONE
 Sequence : 15
 Code : R - Operation on specific record
 Type : UP - Update, after-image

Entry specific data

Column	*...+...1...+...2...+...3...+...4...+...5
00001	'Still Our Customer 529 Bestcust D'
00051	'rive
00101	' b x 0001'
00151	'-01-010001-01-01

Bottom

Copyright Central Park Data Systems, Inc.

22

RANDOM WRKSPLF FUN FACT!!!



- New parameters were added at V5R4.
- Prompt the command and use them!

Work with Spooled Files (WRKSPLF)

Type choices, press Enter.

Select files for:

User	*CURRENT	Name, generic*, *CURRENT...
Print device	*ALL	Name, *ALL, *OUTQ
Form type	*ALL	Form type, *ALL, *STD
User data	*ALL	User data, generic*, *ALL
ASP	*ALL	1-32, *ALL, *ASPDEV
Spooled file	*ALL	Name, generic*, *ALL
Job name	*ALL	Name, generic*, *ALL
User		Name, generic*, *ALL
Number		000000-999999, *ALL

Time period:

Start time and date:

Beginning time	*AVAIL	Time, *AVAIL
Beginning date	*BEGIN	Date, *BEGIN

End time and date:

Ending time	*AVAIL	Time, *AVAIL
Ending date	*END	Date, *END

ASP device

	*	Name, *, *SYSBAS, *CURASPGRP
--	---	------------------------------

Output

	*	*, *PRINT
--	---	-----------

Copyright Central Park Data Systems, Inc.

25

Learning from Hansel and Gretel

- First gather digital “bread crumbs” to prepare for your journey

```
CRTDTAARA DTAARA(DEVLIB/JOBSTEP) TYPE(*CHAR) LEN(1)
```

- As you start your journey, begin leaving bread crumbs:

```
CHGDTAARA DTAARA(JOBSTEP (1 1)) VALUE('A')
```

- For each step or process in the job stream, leave more bread crumbs:

```
CHGDTAARA DTAARA(JOBSTEP (1 1)) VALUE('C')
```

- Don't allow this job to run again if the data area is not at a certain value, in this case it must be blank.

```
RTVDTAARA DTAARA(JOBSTEP (1 1)) RTNVAR(%JOBSTEP)
IF COND(%JOBSTEP NE ' ') THEN(DO)
  SNDPOMMSG MSG('Previous AR Month End did not end !
              actually!')
  RETURN
ENDIF
/* When here, begin AR Month End Close */
```



Copyright Central Park Data Systems, Inc.

26

Building auto-recovery into the application

```

*UPDATEPGM1.RPGLE
Line 23      Column 50      Insert  30 changes
.....1.....2.....3.....4.....5.....6.....7.....
000100
000101      // Only records with a blank in field rodets should be processed
000102      if rodets = *blank;
000103
000104      // Blah blah blah record processing occurs here
000105
000106      // Mark each record as processed
000107      rodets = '1';
000108      update customer;
000109
000110
000111

```

Subsequent updates become dependent on the success of the previous steps:

```

*UPDATEPGM2.RPGLE
Line 10      Column 80      Replace  4 changes
.....1.....2.....3.....4.....5.....6.....7.....8.....9.....
000100
000101      // Only records with a '1' in field rodets should be processed.
000102      // This means that the record has gone through the first step of 12 month end.
000103      if rodets = '1';
000104
000105      // Blah blah blah more record processing occurs here
000106
000107      // Mark each record as processed
000108      rodets = '2';
000109      update customer;
000110
000111

```

Copyright Central Park Data Systems, Inc.

27

Using command SNDMSG as your breadcrumb trail

- When a message is sent using this command a message ID is not associated with it.
- The sender is the user profile.
- The recipient can be QSYSOPR, *ALLACT or *REQUESTOR

```

From . . . . . : CGUARINO      Severity . . . . . : 80
Message type . . . . . : Information
Date sent . . . . . : 08/31/11    Time sent . . . . . : 15:43:21

Message . . . . . : This is a test message sent using SNDMSG

```

Copyright Central Park Data Systems, Inc.

28

Using command SNDPGMMSG as your breadcrumb trail

- This command can only be used in a CL program
- The recipient can be QSYSOPR, a program message queue, a specific user ID, *ALLACT or the history log *HSTLOG
- Helpful for sending information message from an interactive program, such as “Program xxxxxx submitted to batch.”
- You can specify the actual text or use the existing text from a message file.
- Very helpful when your message description has variables that can create specific informational text.
- When the message is prompted the actual message ID will display.

```

Additional Message Information
Message ID . . . . . : MTH0001      Severity . . . . . : 00
Message type . . . . . : Information
Date sent . . . . . : 08/31/11      Time sent . . . . . : 15:31:22
Message . . . . . : Job ARMONTHEND Completed Job Step 2 On 083111 At 153122

```

Copyright Central Park Data Systems, Inc.

29

How the system operator messages screen can be helpful

- When a batch job is being actively being monitored by a user sending a message to the system operator screen implies an urgency.
- Many shops have system monitoring software in place that will intercept these messages and perform a designated action.
- The SNDMSG command can request a user response and designate where the response will be delivered.
- There is a greater chance of a message being seen here to help prevent a backlog of jobs in the queues.
- To see VERY SERIOUS error system messages in a concentrated form create message queue QSYS/QSYSMSG
 - The system will automatically decide which serious message will be reported to this message queue

Copyright Central Park Data Systems, Inc.

30

Using message files to improve your breadcrumb trail

- Each process in this program identifies itself to the system.
- It uses a general purpose message sending subroutine

```

MONTHENDCLCLE
Line 43      Column 108      Replog
+ 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8
000600      DCL      VAR(&JOBSTEP) TYPE(*CHAR) LEN(1)
000700      DCL      VAR(&MSGTEXT) TYPE(*CHAR) LEN(100)
000800      DCL      VAR(&STATUS) TYPE(*CHAR) LEN(30)
000900      DCL      VAR(&COMPLETED) TYPE(*CHAR) LEN(9) +
001000      VALUE('Completed')
001100      DCL      VAR(&ERRORMSG) TYPE(*CHAR) LEN(20) +
001200      VALUE('Ended with error at ')
001300
001400      /* Global error handler */
001500      MSGID(CPF0000) EXEC(GOTO CMDLBL(ERRRRTN))
001600
001700      /* Housekeeping */
001800      RTVJOBA JOB(&JOBNAME)
001900      RTVSYSVAL SYSVAL(QDATE) RTNVAR(&QDATE)
002000
002100
002200      /* Step 1 */
002300      /* Copy company 01 customers before any processing begins */
002400      CHGVAR      VAR(&JOBSTEP) VALUE('1 Copy files')
002500
002600      DLTF      FILE(CUSTMAST01)
002700      MSGID(CPF0000)
002800
002900      CPYF      FROMFILE(CUSTMAST) TOFILE(DEVLIB/CUSTMAST01) +
003000      MSGOPT(*ADD) CRTFILE(*YES) +
003100      FROMKEY(*BLDKEY ('01')) TOKEY(*BLDKEY ('01'))
003200
003300      CHGVAR      VAR(&STATUS) VALUE(&COMPLETED)
003400      CALLSUBR      SUBR(SNDMSGSUBR)
003500
003600
003700      /* Step 2 */
003800      /* Re-age Open A/R Balances */
003900      CHGVAR      VAR(&JOBSTEP) VALUE('2 Re-age A/R')
004000
004100      CALL      PGM(REAGEAR)
004200
004300      CHGVAR      VAR(&STATUS) VALUE(&COMPLETED)

```

Copyright Central Park Data Systems, Inc.

31

The road to recovery

- Create your message file first
- Add a generic message ID, similar to CPF9898, with a length to accommodate your longest message

```

Create Message File (CRTMSGF)

Type choices, press Enter.

Message file . . . . . apprec      Name
Library . . . . . devlib      Name, *CURLIB
Text 'description' . . . . . Message file for my company's applications

```

```

Add Message Description (ADDMSGD)

Type choices, press Enter.

Message identifier . . . . . > MTH0001      Name
Message file . . . . . > APPREC      Name
Library . . . . . > DEVLIB      Name, *LIBL, *CURLIB
First-level message text . . . . . > '&1'

Message data fields formats:
Data type . . . . . > *CHAR      *NONE, *QDCHAR, *CHAR...
Length . . . . . > 100      Number, *VARY

```

Copyright Central Park Data Systems, Inc.

32

When the messages are sent to QSYSOPR

```

                                Display Messages
                                System:      CPDS
Queue . . . . . : QSYSOPR           Program . . . . . : *DSPMSG
Library . . . . . : QSYS             Library . . . . . :
Severity . . . . . : 70              Delivery . . . . . : *HOLD

Type reply (if required), press Enter.
Cleanup of system journals and system logs successfully completed.
Cleanup has completed.
Service Agent is analyzing your system product activity log entries.
Service Agent is analyzing your system product activity log entries.
Job ARMONTHEND Completed Job Step 1 On 083111 At 153122
Job ARMONTHEND Completed Job Step 2 On 083111 At 153122
Attempt to divide by zero (C G D F).
Reply . . . : C
Job ARMONTHEND Ended with error at Job Step 3 On 083111 At 153126

                                Bottom

```

Copyright Central Park Data Systems, Inc.

33

When the messages are sent to the history log

- Use the message ID to extract the message from the log
- Specifying the exact error message you can easily filter which errors you are trying to resolve.
 - Using a generic message ID ensures you retrieve every message

```

Selection or command
==> DSPLOG MSGID(MTH0001)

```

```

                                Display History Log Contents

Job ARMONTHEND Completed Job Step 1 On 083111 At 153122
Job ARMONTHEND Completed Job Step 2 On 083111 At 153122
Job ARMONTHEND Ended with error at Job Step 3 On 083111 At 153126

```

Copyright Central Park Data Systems, Inc.

34

Using the DSPLY statement – no WORKSTN files

- Quickly create an interactive program to change runtime variables
- Ideal for testing new functions or sections of new code.
- When this program runs the “Value of x” is displayed.
- The program will wait for a response.
 - The returned value will be stored in variable x.
- After 5 invalid attempts the program displays an error message.

```

DSPVAR.RPGLE
Line 1      Column 1      Replace
.....1.....+.....2.....+.....3.....+.....4.....+.....5.....+.....6.....+.....7.
nnnn1nn
000200      D x              S              S 0 inz(100)
uuu303
000304
nnn305      /free
000401
000500      dsply 'Value of x is ' ' ' x;
000502      dsply 'Now the value of x is' ' ' x;
000503      dsply 'Finally the value of x is ' 'QSYSOPR' x;
000505
uuu600      finish = *on;
uuu700      return;
000701
000702      /end-free
  
```

Copyright Central Park Data Systems, Inc.

35

Coding the DSPLY statement – a closer look

- DSPLY 'message text' 'message queue' response
- Message text and message queue can be literals, field names or message IDs from the message file.
 - QSYSOPR can also be specified as message queue.
- At least message text or response must be coded.
 - Both may be specified.
- If message queue is not specified, default values are:
 - Interactive = *EXT (your display)
 - Batch = QSYSOPR

Copyright Central Park Data Systems, Inc.

36

The D\$PLY statement in action:

- After typing 125 the value is redisplayed. Next, 145 is keyed.
- Final request is displayed in QSYSOPR.

```

Display Program Messages
Job 958889/CHARLIE/QPADEV0002 started on 03/22/09 at 22:47:16 in subsystem Q
D$PLY Value of x is 100
125
D$PLY Now the value of x is 125

Type reply, press Enter.
Reply . . . 145

F3=Exit F12=Cancel
Waiting for reply to message on message queue QSYSOPR.

```

```

Display Messages
Queue . . . . : QSYSOPR          System: CPDS2
Library . . . : QSYS            Program . . . : *DSPMSG
Severity . . . : 10             Library . . . :
Delivery . . . : *HOLD          Cleanup of system journals and system logs successfully completed.
Cleanup has completed.
D$PLY Finally the value of x is 145
Reply . . .

```

Copyright Central Park Data Systems, Inc.

37

Built-in error handling

- On this screen 5 invalid values were keyed.
 - Program was expecting numeric, received character.
 - Program ends with D\$PLY operation failed.

```

Display Program Messages
Job 958924/CHARLIE/QPADEV0002 started on 03/22/09 at 23:13:20 in subsystem Q
D$PLY Value of x is 100
a
D$PLY Value of x is 100
b
D$PLY Value of x is 100
c
D$PLY Value of x is 100
d
D$PLY Value of x is 100
e
The D$PLY operation failed (C G D F).

```

Type reply, press Enter.
Reply . . .

Copyright Central Park Data Systems, Inc.

38

Detailed information on why the error message was issued:

```

Additional Message Information
Message ID . . . . . : RNQ0333      Severity . . . . . : 99
Message type . . . . . : Inquiry      Time sent . . . . . : 23:13:29

Message . . . . . : The DSPLY operation failed (C G D F).
Cause . . . . . : RPG procedure DSPVAR in program DEVLIB/DSPVAR received an
                  error at statement 8 while performing a DSPLY operation. The error was
                  caused by one of the following conditions:
                  -- The message queue is not valid.
                  -- The message identifier could not be found in message file QUSERMSG.
                  -- The message file QUSERMSG is not in the library list.
                  -- More than five responses have been entered that are not valid.
Recovery . . . . . : Check the job log for more information on the cause of the
                  error and contact the person responsible for program maintenance.
Possible choices for replying to message . . . . . :
                  D -- Obtain RPG formatted dump.

Press Enter to continue.
More...

```

Copyright Central Park Data Systems, Inc.

39

Next tip: Changing CL command parameters at runtime

- The question marks invoke the runtime command prompter

```

000100 PGM
000200
000300 /* Retrieve current audit number from data area, */
000400 /* allow the user to change before running the program. */ Global MONMSG
000500
000600 DCL VAR(&NUMBER) TYPE(*DEC) LEN(5 0)
000700
000800 MONMSG MSGID(CPF6601) EXEC(COTO CMDLBL(ERRHANDLER))
000900
001000 RLVDTAARA DTAARA(AUDIT) RLVVAR(&NUMBER)
001100
001200
001300 /* Allow the user to send a start message to any user. */
001400 ? SNDMSG ?*MSG('Starting program UPDAVG') ??TOUSR()
001500
001600 /* Change the data area with either the returned value. */
001700 ? CHGDTAARA ?*DTAARA(DEVLIB/AUDIT) ??VALUE(&NUMBER)
001800
001900 /* Run the program to update the average year to date sales */
002000 CALL PRM(UPDAVG)
002100
002200 /* Run the query to display the selected updated records */
002300 /* Display records with payment days > 120 or average balance < $500 */
002400 RUNQRY QRY(CREDCHK) OUTTYPE(*DISPLAY)
002500
002600 ? SNDMSG ?*MSG('End of program UPDAVG') ??TOUSR()
002700 ?-TOMSGQ() ?-MSGTYPE() ?-RFXMSGQ() ?-CCSID()
002800
002900 RETURN
003000
003100 ERRHANDLER: SNDPGMMSG MSG('The user has ended this program +
003200 abnormality.')
003300 RETURN
003400
003500 ENDPGM

```

Prompt w/
parms, allows
F10

Prompt w/
values

Prompt w/o
addl parms

Copyright Central Park Data Systems, Inc.

40

How do the question marks work?

- Can only be used in an interactive environment
- The system will issue message CPD0037 if run in batch
- ? = Instructs the system to prompt the command with all parameters
- ?* = Protects the parameter, allows it to be prefilled
- ?? = Allows input to the parameter
- ?- = Suppresses the parameters from being displayed

Copyright Central Park Data Systems, Inc.

41

Automatic command prompting in action

- Since ?* was specified the SNDMSG MSG parameter is protected
- The ?? on the TOUSR parameter opens it for input
- Not all parameters have been blocked so F10 is still available

Send Message (SNDMSG)

Type choices, press Enter.

Message text > **'Starting program UPDAVC'**

To user profile _____ Name, *SYSOPR, *ALLACT...

Bottom

F3=Exit F4=Prompt F5=Refresh F10=Additional parameters F12=Cancel

F13=How to use this display F24=More keys

```

^ Allow the user to send a start message to any user. ^/
?      SNDMSG ?*MSG('Starting program UPDAVC') ??TOUSR()

```

Copyright Central Park Data Systems, Inc.

42

Prompting with pre-filled value:

```

Change Data Area (CHGDTAARA)
Type choices, press Enter.

Data area specification:
Data area . . . . . > AUDIT      Name, *LDA, *GDA, *PDA
Library . . . . . > DEVLIB      Name, *LIBL, *CURLIB
Substring specifications:
Substring starting position . *ALL      1-2000, *ALL
Substring length . . . . .      1-2000
New value . . . . . > 255

```

- “Data area” is protected, “New value” can be overridden.

Copyright Central Park Data Systems, Inc.

43

SNDMSG again- Note: No additional parameters (F10) are available.

```

Send Message (SNDMSG)
Type choices, press Enter.

Message text . . . . . > 'End of program UPDAVG'
To user profile . . . . .      Name, *SYSOPR, *ALLACT...

Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys

```

Global MONMSG
Appears when F3
is pressed by user



```

Selection or command
===>
F3=Exit  F4=Prompt  F9=Retrieve  F12=Cancel
F23=Set initial menu
The user has ended this program abnormally.

```

Copyright Central Park Data Systems, Inc.

44

Resolving decimal data issues with level check *NO

- “Some” programmer adds a numeric field to the end of the record
- An older program adds records to the file
- The new record(s) now have blanks in these new numeric fields

```

Display Physical File Member
File . . . . . : CUSTMAST      Library . . . . . : DEVLIB
Member . . . . . : CUSTMAST    Record . . . . . : 4
Control . . . . . :           Column . . . . . : 160
Find . . . . . :

6 . . . . . + . . . . . 7 . . . . . + . . . . . 6 . . . . . + . . . . . 7 . . . . . + . . . . .
F860F0F3 60F0F300 120F0025 5F000000 0F          *8-03-03          *
F860F0F7 60F0F200 068F0025 5F000000 0F          *8-07-02          *
F160F0F1 60F0F100 000F0000 0F404040 40          *1-01-01          *
***** END OF DATA *****

```

```

FIXCUS.RPGLE
Line 1      Column 1      Replace
.....Keywords+++++
UUU1UU      H fixnbr(*inputpacked:*zoned)
000200
000300      Feustmast  up  e          disk
UUU4UU
UUU5UU      C          update  custmast
UUU6UU

```

Copyright Central Park Data Systems, Inc.

45

We execute FIXCUS and examine the job log:

```

dpppfm custmast
call fixcus
MCH1202 Decimal data error.

```

A decimal data error message (MCH1202) will appear for each record encountered with invalid packed data, which will automatically be repaired!

```

Display Physical File Member
File . . . . . : CUSTMAST      Library . . . . . : DEVLIB
Member . . . . . : CUSTMAST    Record . . . . . : 4
Control . . . . . :           Column . . . . . : 160
Find . . . . . :

6 . . . . . + . . . . . 7 . . . . . + . . . . . 6 . . . . . + . . . . . 7 . . . . . + . . . . .
F860F0F3 60F0F300 120F0025 5F000000 0F          *8-03-03          *
F860F0F7 60F0F200 068F0025 5F000000 0F          *8-07-02          *
F160F0F1 60F0F100 000F0000 0F000000 0F          *1-01-01          *
***** END OF DATA *****

```

Copyright Central Park Data Systems, Inc.

46

H Spec FIXNBR option

- This keyword will find and repair any declared numeric field that contains non-numeric data
- The value will ALWAYS be set to *zero
- It does not matter where the field resides on the record
- FIXNBR can also be specified on the CRTBNDRPG command
 - H spec keyword will override compile command

User profile	<u>*USER</u>	*USER, *OWNER
Authority	<u>*LIBCRTAUT</u>	Name, *LIBCRTAUT, *ALL...
Truncate numeric	<u>*YES</u>	*YES, *NO
Fix numeric	<u>*NONE</u>	*NONE, *ZONED, *INPUTPACKED

Copyright Central Park Data Systems, Inc.

47

Record format identifier fun facts!



- Every record format in a file has a unique format level identifier.
- The system assigned identifier is determined based on the following values:
 - Length of the record format
 - Record format name
 - Number and order of fields defined
 - The data types and size of the fields
 - Field names and number of decimal positions
 - Whether or not the fields allow null values

```

Message . . . . : Error message CPF4131 appeared during OPEN for file
                  CUSTMAST (C S D F).
Cause . . . . : RPG procedure WRTCUS in program DEVLIB/WRTCUS received the
                  message CPF4131 while performing an implicit OPEN operation on file
                  CUSTMAST. The actual file is CUSTMAST.
  
```

Copyright Central Park Data Systems, Inc.

48

Still more record format identifier fun facts!



- Not every CHGPF will cause the format identifier to be changed.
- These DDS keywords can be added or deleted without any consequence of a runtime level check error occurring:
 - COLHDG, CHECK
 - EDTCDE, EDTWRD
 - REF, REFFLD
 - CMD, VALUES
 - RANGE, DFT
 - TEXT, CCSID
 - Key fields, access path keywords, select/omit fields too!

Copyright Central Park Data Systems, Inc.

49

Some Emergency File Lock Resolution Tricks



- Let's say you need to CLEAR a file but another job has a lock.

1

```
Selection or command
==> clrpfm devlib/reportwrk

F3=Exit F4=Prompt F9=Retrieve F12=Cancel F13=Info
F23=Set initial menu
Member REPORTWRK already in use.
```

2

```
Enter SQL Statements

Type SQL statement, press Enter.
Current connection is to relational database S108E6BE.
==> delete from devlib/reportwrk
```

3

```
Confirm Statement

You are about to alter (DELETE or UPDATE) all of the records in your file(s).
Press Enter to confirm your statement to alter the entire file.
Press F12=Cancel to return and cancel your statement.
```

4

```
Enter SQL Statements

Type SQL statement, press Enter.
> delete from devlib/reportwrk
5 rows deleted from REPORTWRK in DEVLIB.
==>
```

Copyright Central Park Data Systems, Inc.

50

Object Lock Emergency!!!



- Made changes to display file CUSINQFM and need to install **NOW!**

Save DDS - Create Display File

Type choices, press Enter.

Save DDS source	Y	Y=Yes
Source file	QDDSSRC	F4 for list
Library	XMLLIB	Name, *LIBL ...
Member	CUSINQFM	F4 for list
Text	Sample customer inquiry	

Create display file	Y	Y=Yes
Prompt for parameters		Y=Yes
Display file	CUSINQFM	F4 for list
Library	XMLLIB	Name, *CURLIB
Replace existing file	Y	Y=Yes

Submit create job in batch Y Y=Yes

Specify additional

Copyright Central Park Data Systems, Inc.

51

How do you get around this situation?

- The display file is already in use, cannot replace it

Type options, press Enter.

2=Change 3=Hold 4=End 5=Work with 6=Release 7=Display message

8=Work with spooled files 10=Disconnect ...

Opt	Subsystem/Job	User	Type	CPU %	Function	Status
—	QBATCH	QSYS	SBS	.0		DEQW
—	CUSINQFM	CHARLIE	BCH	.0	CMD-CRTDSPF	LCKW
—	SAXPARSER	QUSER	PJ	.0		EOFW
—	QINTER	QSYS	SBS	.0		DEQW
—	QPADEV0002	CHARLIE	INT	.0	CMD-WRKACTJOB	RUN
—	QPADEV0003	ALEXANDRA	INT	.0	PGM-CUSINQ	DSPW
—	QPADEV0005	ADAM	INT	.0	PGM-CUSINQ	DSPW

Message : Cannot allocate object CUSINQFM in XMLLIB type *FILE.

Message : File CUSINQFM not created in library XMLLIB.

* * * * * E N D O F C O M P I L A T I O N * * * * *

Copyright Central Park Data Systems, Inc.

52

Don't go through the "wall" just go around it!

```

Work with Object Locks
System: CPDS2
Object . . . . : CUSINOFM      Type . . . . . : *FILE-DSP
Library . . . . : XMLLIB       ASP device . . . : *SYSBAS

Type options, press Enter.
4=End job      5=Work with job      8=Work with job locks

Opt  Job          User      Lock      Status      Scope      Thread
--  -
   QPADEV0002    ADAM      *SHRNUP   HELD        *JOB
   QPADEV0003    ALEXANDRA  *SHRNUP   HELD        *JOB

```

Put the newer version (display file and recompiled program) higher up in everyone's library list!

```

Display Library List
Job: QPADEV0003 User: ALEXANDRA Number: 945791 System: CPD52

Type options, press Enter.
5=Display objects in library

Opt Library Type ASP Device Text
- QSYS SYS - System Library
- QSYS2 SYS - System Library for CPI's
- QHLPSYS SYS
- QUSASYS SYS System Library for Users
- TEMP USR For temporarily moving objects to prod
- XMLLIB USR
- QGPL USR General Purpose Library
- QTEMP USR

```

Copyright Central Park Data Systems, Inc.

53

Locked display file problem solved

- Having library 'TEMP' as a permanent part of each user's library list allows for these emergency situations.
- Make sure you clear out this library after you create the new objects in the real production library.
 - Issue a CLRLIB every night as a scheduled job.

```
Selection or command
==> CHGSYSVAL SYSVAL(QUSRLIBL) VALUE('QGPL QTEMP TEMP')
```

```

Display System Value

System value . . . . . : QUSRLIBL
Description . . . . . : User part of the library list

Sequence          Sequence
number           number   Library
  0
 10      QGPL
 20      QTEMP
 30      TEMP

```

Copyright Central Park Data Systems, Inc.

54

When each user signs on, library TEMP is automatically added to their job:

Selection or command
==> dsplibl

Display Library List					System: CPDS2
Type options, press Enter.					
b=Display objects in library					
Opt	Library	Type	ASP Device	Text	
-	QSYS	SYS		System Library	
-	QSYS2	SYS		System Library for CPI's	
-	QHLPSYS	SYS			
-	QUSRSYS	SYS		System Library for Users	
-	QGPL	USR		General Purpose Library	
-	QTEMP	USR			
-	TEMP	USR		For temporarily moving objects to produ	

Copyright Central Park Data Systems, Inc.

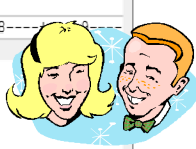
55

Plan "B" to install production changes during the day

- Never have a display file lock again
- Every use has their own copy of the display file in QTEMP

```

WEATHQTEMP.CIP 01
Line 1      Column 1      Replace
-----1-----2-----3-----4-----5-----6-----7-----8-----
000100      PGM
000200
000201/** This program first copies the display file to the user's QTEMP library **/
000202/** If we need to back out changes or simply apply new changes          **/
000203/** during the middle of the day we simply change the object in XMLLIB   **/
000204/** and have the user sign off and then sign on again.                  **/
000206
000207/** If the display file is already in QTEMP then don't recreate it.      **/
000300      CHKOBJ      OBJ(QTEMP/WEATHERFM) OBJTYPE(*FILE)
000400      MONMSG      MSGID(CPF9001) EXEC(CRTDUPOBJ OBJ(WEATHERFM) +
000500                      FROMLIB(XMLLIB) OBJTYPE(*FILE) TOLIB(QTEMP))
000600
000601      ** OVRDSPF forces the system to use the QTEMP version                **/
000700      OVRDSPF      FILE(WEATHERFM) TOFILE(QTEMP/WEATHERFM)
000800
000801      CALL WEATHERSCN
000900
001000      ENDPGM
  
```



Copyright Central Park Data Systems, Inc.

56

The QTEMP solution in action

- Simply replace your “base” display file
- Every time a user signs on they’ll have the latest version

Display Open Files

Job . . . : QPADEV000B User . . . : CHARLIE Number . . . : 095100
 Number of open data paths : 5

File	Library	Member/ Device	Record Format	File Type	I/O Count	----Open--- Opt Shr-Nbr	Relative Record
QDUI80	QSYS	QPADEV000B	USRRCD	DSP	255	IO NO	
QDDSPEXT	QSYS	QPADEV000B	OUTFMT	DSP	8	IO NO	
WEATHERFM	QTEMP	QPADEV000B	SCREEN1	DSP	1	IO NO	
QDUI80	QSYS	QPADEV000B	USRRCD	DSP	2	IO NO	
QDDSPDF	QSYS	QPADEV000B	DETAIL2	DSP	14	IO NO	

Copyright Central Park Data Systems, Inc.

57

Thinking Outside the Developer's Toolbox: Application Program Recovery and Other Cool Tricks



Charles Guarino

THANK YOU !!!

Copyright Central Park Data Systems, Inc.

58

Sessions by Charles Guarino

Sunday 9:30AM	REST Web Services for Everyday RPG Consumption Using REST to extend your applications beyond your four walls
Sunday 12:30PM	Raising the Bar: A Jumpstart to Using SOAP Web Services in RPG A step-by-step discussion on implementing IBM's WSDL2RPG
Monday 11:00AM	From STRPDM to RDP An introduction to RDP and source code editing
Monday 2:00PM	From STRPDM to RDP – The Journey Continues Exploring advanced features in RDP and making it your own
Monday 3:30PM	From STRSDA and STRRLU to RDP A step by step demonstration on using the RDP visual designer
Tuesday 9:30 AM	From STRDBG to the RDP Debugger A walk through on using the debugging perspective
Wednesday 9:30AM	Introduction to Processing XML in RPG A beginner's guide to processing XML in RPG
Wednesday 2:00PM	There's Gold in Them Thar Hills! Making the most of an RPG formatted dump
Wednesday 5:00PM	Program Application Recovery and Other Tricks Proven techniques to keep your cool before and after an IT crisis

Copyright Central Park Data Systems, Inc.

59