Performing DRD Functions Within an IMS Transaction

John Ganci
Texas Instruments

Thursday, August 11, 2011
Session 9402
Presentation Outline

• Prologue
• Single Point of Control Overview
• Dynamic Resource Definition Overview
• Type-2 Commands Used for DRD
• The Common Service Layer Application Programming Interface
• Implementing Type-2 Commands Within a Transaction
• Sample Transaction Invocations
• Sources of Documentation
• Summary
Prologue

- Texas Instruments allows application developers to request IMS sysgen changes
  - Requests are entered using an IMS transaction
  - Add, change, and delete programs, transactions, and databases

- Sysgens were originally done once per week

- Real-time IMS resource definitions were mandated

- A user modification was written for IMS 1.3 to manage IMS resource definitions in real time
  - Real-time definition (RTD) is the TI name of the provided functionality
  - Accessed via a user transaction (IRSREQ) and a “system programmer” transaction (RTDTRAN)
  - Add, change, delete; PDIR, DDIR, SMB
  - IRSREQ message switches to RTDTRAN
  - System programmers can run RTDTRAN directly

- IMS V10 introduced Dynamic Resource Definition (DRD)
  - New type-2 IMS commands CREATE, UPDATE, DELETE
  - Use SPOC via TSO or batch; no transaction interface provided
  - Provides a CSL API to perform IMS commands from a program

- Challenge
  - Use the CSL API to replace all the user modified code
  - Leave IRSREQ unchanged
Single Point of Control (SPOC) Overview

- Common Service Layer (CSL)
- CSL components; each is a separate address space
  - Operations Manager (OM)
    - Command entry
  - Resource Manager (RM)
    - Manage global and IMSplex-wide resources
  - Structured Call Interface (SCI)
    - Communication between address spaces
- SPOC
  - Interfaces between a user and the OM
  - Batch, TSO, REXX
    - Only the TSO interface is discussed here
- Session 9811 “Using IMS Dynamic Resource Definition (DRD)” is a hands-on lab where you can practice using the DRD type-2 commands
SPOC Overview

- TSO access is via ISPF
- Requires IMS datasets
  - SDFSCLST, SDFSDATA, SDFSEEXEC, SDFSMLIB, SDFSPLIB, SDFSSLIB, and SDFSTLIB
- Invoke from ISPF Option 6
  - `ex 'your.names.SDFSEEXEC(DFSSPSRT) 'HLQ(your.hlq) ALTRESL("your.ims.reslib")'
  - Note the double single quotes
  - Easier way
    - Create your own CLIST/REXX EXEC that has all the above data
    - Invoke it: `ex 'daccjjg.clist(spoc)'
- First-time setup required
  - Next few slides show what to do
SPOC Overview

- User CLIST DACCJJG.CLIST(SPOC)
- EXEC 'AACC.IMSADMIN.R11.SDFSEXEC(DFSSPSRT)'
  'HLQ(AACC.IMSADMIN.R11) ALTRESL("SYS1.IMSVS.RESLIB")'
SPOC Overview

- First time only: Options; Preferences ...
SPOC Overview

- First time only: set default IMSplex
SPOC Overview

- Display information about a database ...
SPOC Overview

- Output; then display information about a program ...
SPOC Overview

• Output; then display information about a transaction …
SPOC Overview

- Output; then display some invalid output …
SPOC Overview

- Output
  - Next two slides show more details about the error
SPOC Overview

- Display command entry and log …
SPOC Overview

- Note the return and reason codes; two sets of them
  - First set is the OM return code and reason code
  - Second set is the QUERY return code and reason code
DRD Overview

- Enabled via IMS PROCLIB member DFSCGxxx
  - `MODBLKS=DYN`
- Implemented using type-2 commands
- Eliminates the need for a MODBLKS sysgen
- MODBLKS “enhanced” by Resource Definition Data Sets
  - Data is exported/imported to/from the RDDS
  - Exports: sysgen resources and dynamic create/update resources
  - Imports: done at cold start or via the IMPORT command
  - Analogous to a checkpoint dataset for the type-2 commands
- Eventually could remove MODBLKS; just use RDDS
  - We’ve done this
- There are utilities that manage the RDDS
  - See the *IMS V11 System Utilities* manual
Type-2 Commands Used for DRD

- Quick look at the format of type-2 commands and output
- Understand how to construct commands and interpret output
  - Output is in XML
- The *IMS V11 Commands* manuals document
  - Command syntax
  - Command output
  - Return codes, reason codes, and completion codes
- The XML output is also documented in the *IMS V11 System Programming APIs* manual
  - See Chapter 9
Type-2 Commands Used for DRD

- Documented in *IMS V11 Commands, Volumes 1, 2*
- CREATE, DELETE, UPDATE
  - DB, PGM, TRAN
  - Also DBDESC, PGMDESC, RTC, RTCDESC, and TRANDESC
    - Not discussed here
- QUERY
  - Although not needed for DRD, useful for our implementation
  - Syntax: `<cmd> <type> NAME(<resource>) [<attributes>]`
    - DELETE DB NAME(DI21PART)
    - UPDATE TRAN NAME(PART) SET(CLASS(1))
    - QUERY TRAN NAME(PART) SHOW(CLASS,MAXRGN)
- Command invocation
  - Output is in XML
  - May have a return code and reason code
  - May have a completion code
- Examples are shown following the CSL API macro slides
Common Service Layer Application Programming Interface

- Documented in *IMS V11 System Programming APIs*
  - Chapter 3. Writing a CSL client
  - Chapter 4. CSL automated operator program requests
  - Chapter 6. Writing a CSL OM client
  - Chapter 8. Writing a CSL SCI client
  - Chapter 9. CSL Operations Manager XML output
- Describes assembler macros used to perform the client functions
- Documents the SCI, OM, and RM return and reason codes
- Describes output returned for each request
  - Output is in XML
CSL API

- We wrote a CSL SCI client that issues AOP requests
  - Called by RTDTRAN
  - More about our client appears in later slides
- Protocol requires that the client
  - SCI: Connect to SCI
    - Issue a command registration request to register as an AOP
    - Issue a ready request
  - OM: Issue the command and receive command output
  - User: Process command output
  - SCI: Release command output buffer
  - SCI: Disconnect from SCI
    - Quiesce
    - Deregister
- All of the above except the User bullet are done using CSL macros
CSL API – the macros

- CSLSCREG
  - Register to SCI (connect with SCI)
- CSLSCRDY
  - Ready the member (IMSplex member) to SCI
- CSLSCBFR
  - Release storage allocated by SCI
- CSLSCQSC
  - Quiesce the member to SCI
- CSLSCDRG
  - Deregister the member from SCI (terminate the SCI connection)
- There are other SCI macros; we did not need them
- CSLOMCMCMD
  - Requests that an IMS command be issued (type-1 or type-2)
CSL API – the macros

• Each macro invocation returns
  • A return code and a reason code
• Each macro has a DSECT request that generates equates needed by the macro invocation
• *System Programming APIs* documents the values
• Return code high order byte indicates the component that set the return code and reason code
  • X'00' IMS set the return and reason code
  • X'01' SCI set the return and reason code
  • X'02' OM set the return and reason code
  • X'03' RM set the return and reason code
• Examples shown later
CSL API – the macros

- The DSECTs
CSL API – the macros

• DFSCMDRR
  • While not needed, the DFSCMDRR macro in SDFSMAC contains equates for the return codes, reason codes, and completion codes for all IMS commands that are routed from OM
CSL API – the macros: CSLSCREG

- Excerpt from *System Programming APIs* manual

- Type we used
- Needed for all subsequent calls
- Return and reason codes
CSL API - register

- CSLSCREG invocation
CSL API - ready

- CSLSCRDY invocation; note the SCITOKEN value on line 835
CSL API – issue command

- CSLOMCMD invocation; note output values on lines 1098-99
CSL API – release buffer

- CSLSCBFR invocation; release buffer
CSL API - quiesce

- CSLSCQSC invocation
CSL API - deregister

- CSLSCDRG invocation
CSL API – CSLOMCMCD sample output 1

- Actual buffer from successful create database
CSL API – CSLOMOMCMD sample output 1 (edited)

• Edited printable part of the returned buffer
Unsuccessful delete database; edited returned buffer
CSL API - summary

- Token returned by register request
- All subsequent invocations use the token
- All invocations receive a return code and reason code
- All macros, return and reason codes are documented in the System Programming APIs manual
- All XML output is documented in
  - Command Reference manuals
  - System Programming APIs manual
- Command security is required for non-authorized programs
  - Unless CSL OM runs with no security
  - Need something like DFSCCMD0 command authorization checking
Implementation – describe environment

- TI’s Real-Time Definition manages IMS … with IMS!
  - Extension of using “/” commands
  - Implementation is via a transaction
- IMS V9 RTD transaction ran as a normal transaction
  - GU
  - Process input
    - Determine request (Add, Change, Delete; SMB, PDIR, DDIR)
    - Process the request
      - Some parts inline
      - Some parts in Control Region
      - Some parts in another TI specific address space
  - Insert output (to user transaction or input lterm)
- IMS V10 goal
  - Replace “Process the request” (user modifications galore)
    - Replace with CSL API calls that issue type-2 commands
Implementation – describe environment …

• CSL API is easy to use but is complex
• We decided to isolate all CSL API calls in a separate module
• Transaction calls the separate module
  • Analogous to an ASMTDLI call
  • Transaction knows nothing about the CSL API
    • Well, almost nothing
• There are new concepts the transaction needs to know
  • The type-2 commands
  • XML
Implementation – describe environment …

- The CSL API module
  - Provides an interface between the transaction and OM
  - Receives as input a type-2 command
  - Does all OM interaction using CSL macros
    - Initialize CSL OM (Register, Ready)
    - Issue the command
    - Copy returned output from CSL buffer to an obtained user buffer
    - Release CSL output buffer
    - Terminate CSL OM (Quiesce, Deregister)
    - Copy CSL return, reason codes, error message(s) to user area

- Application does not know about CSL or OM
  - Maybe a little: return and reason codes
  - Must “learn” a new language: XML
Implementation – describe environment (Ends)

• Summary of IMS V10 (and beyond) transaction processing
  • GU
  • Decode input; determine function
  • Construct appropriate type-2 command
  • Call the “CSL API interface” with the request
  • Process the returned buffer and return code data
  • Release the returned buffer
  • Insert response to transaction or lterm
    • The response to the IRSREQ transaction is unchanged!
    • We will see examples of the lterm responses real soon
Sample transaction interactions

- First compare IMS V9 screens to IMS V10/V11 screens
  - IMS V9
    - Formatted screens
      - But uses a local modification!
  - IMS V10/V11
    - Unformatted screens
Sample transaction interactions: IMS V9 vs V11

- Initial screen; displayed by entering RTDTRAN

```plaintext
FUNCTION:  = (A = ADD, C = CHANGE, D = DELETE)  
           (B = DYNAMIC ACE REQUEST)

TYPE: ---- (SMB, PDIR, DDIR)
```

Usage (from a clear screen):

- `RTDTRAN <func> <type>`
- `<func>` = CREATE, UPDATE, DELETE, or QUERY
- `<type>` = DB, PCH, or TRAN
- `RTDTRAN <func>`
- `<func>` = DYNACB
Sample transaction interactions: IMS V9 vs V11

- “Create transaction” input screen
Sample transaction interactions: query database

- RTDTRAN QUERY DB; fill in database name; Enter
Sample transaction interactions: query program

- RTDTRAN QUERY PGM; fill in program name; Enter
Sample transaction interactions: query transaction

- RTDTRAN QUERY TRAN; fill in transaction name; Enter
Sample interactions: successful create database

- RTDTRAN CREATE DB; fill in fields; Enter
Sample interactions: checkpoint and export

- Take checkpoint; note z/OS console messages
Sample interactions: verify create database

- Issue another query database to verify create
  - Note resident values
Sample interactions: unsuccessful delete #1

- RTDTRAN DELETE DB; fill in name; Enter
Sample interactions: unsuccessful delete #1

• The DELETE DB failed
• The CSLOMCMCMD invocation has a return code and reason code
• The DELETE DB invocation has a return code and reason code
  • It also has a completion code
• CSLOMCMCMD
  • *System Programming APIs* documents values
  • Return/reason code X'0200000C/00003008'
    • Return code high byte X'02' means OM set the values
• DELETE DB
  • *Command Reference Volume 1* documents values
  • Return/reason code X'0000000C/00003004'
  • Completion code X'E1'
Sample interactions: unsuccessful delete #2

- RTDTRAN DELETE PGM; fill in name; Enter
Sample interactions: unsuccessful delete #2

- Returned buffer (hard to read!)
Sample interactions: unsuccessful delete #2

- Edited printable part of the returned buffer
Sample interactions: successful create program

- RTDTRAN CREATE PGM; fill in data; Enter
Sample interactions: successful create transaction

- RTDTRAN CREATE TRAN; fill in data; Enter
Sample interactions: WARNING about MSC
See PK89475: update SIDR without setting REMOTE(Y)

- RTDTRAN UPDATE TRAN; fill one data value; Enter
Sample interactions: RTDTRAN vs SPOC

- RTDTRAN update on left; SPOC update on right
Sample transaction interactions: summary

- All RTDTRAN interactions are similar
  - Enter rtdtran <function> <resource>
  - Fill in values in the “fields”
    - The “fields” are not really fields since the screen is unformatted
    - Not all values have to be filled in
  - Transaction does some editing of the input
    - May abort the request early and show the error
    - If no early error, constructs the type-2 command and calls RCSLOM
    - Extracts data from values returned by RCSLOM (data and output buffer)
- Screen response echoes input
  - Shows the built command for CREATE, DELETE, UPDATE
  - Shows the CSL register, ready, and command return/reason codes
  - Shows the command completion code(s)
  - Shows additional data if an error occurs
- Output to IRSREQ is completely different (but unchanged from IMS V9!)
  - It is the IRSREQ input buffer, with an added IRSREQ return code
Sources of Documentation

- **IMS V11 Command Reference, Volume 1**
  - CREATE, DELETE commands
- **IMS V11 Command Reference, Volume 2**
  - QUERY, UPDATE commands
- **IMS V11 Operations and Automation**
  - SPOC introduction
- **IMS V11 System Definition**
  - System definition macros for IMS resources
- **IMS V11 System Programming APIs**
  - Chapters 3, 4, 6: Writing CSL client, AOP requests, CSL OM client
  - Chapter 8: Writing CSL SCI client
  - Chapter 9: OM XML output
- **IMS V11 System Utilities**
  - Chapters 26, 42-46: Batch SPOC, RDDS utilities
Summary

- IMS V10 added Dynamic Resource Definition
- New type-2 commands to manage resources
- Requires Common Service Layer
  - At least SCI and OM
- Type-2 commands entered from non-IMS environment
  - TSO SPOC
  - Batch
- IBM provides an API to access the CSL via a program
  - Very well documented
- TI rewrote it Real-Time Definition transaction to use DRD
  - Used the CSL API; isolated the CSL code in a separate module
    - Some error checking done by TI code, some by IMS code
    - Other programs also use the RCSLOM module
- Command security is still an issue
- Now you, too, can manage IMS resources using IMS!