

百硕同兴

Bayshore Advisor

客 户

通 讯

冬

总第14期 (2008年12月1日)

百硕同兴科技(北京)有限公司

目录

本期主要内容

消息快速 *News Express*

- | | |
|---|---|
| DB2 Utility Best Practices
百硕资深工程师 吴笏笏 | 2 |
| DB2 V9 Index Enhancement
百硕资深工程师 贺明 | 7 |

经验分享 *Experience & Tips*

- | | |
|--|----|
| 使用 DFSORT 处理 DCOLLECT 的数据集信息
百硕资深工程师 邹杰 | 9 |
| SMF 和 PL/I: 案例学习
百硕工程师 罗兴魁 | 13 |

百硕博客 *Bayshore Blog*

- | | |
|---|----|
| 同一主机分区的多 TCPIP 环境下的网络连通性检查
百硕资深工程师 罗文军 | 20 |
| 浅析 Logstream 的用法
百硕工程师 陈银波 | 25 |



DB2 Utility Best Practice

百硕资深工程师 吴笳笳

最佳实践是近几年较为流行的一个概念，它最早是由IBM公司提出来的。所谓最佳实践，是IBM实验室收集了全球客户对其产品所使用的一些经验值，并结合产品的新功能和特性得出的，能够最充分利用其产品特性的最佳使用方法。本人有幸参加了今年在美国拉斯维加斯举办的IBM IOD(Information On Demand)会议。在这次大会上我们不但可以了解关于DB2 V9的一些新功能，而且有机会与IBM DB2实验室的专家近距离探讨DB2的使用方法技巧，更有机会得到一些来自于DB2实验室的最佳实践。在这里，我将与大家分享和DB2用户日常使用最为密切的DB2 Utility的最佳实践，希望这些方法可以帮助大家更好地使用DB2 Utility，尽可能缩短日常维护时间，减少维护成本。

■ COPY Best Practices

- PARALLEL keyword provides parallelism for lists of objects (including partitions)
- CHECKPAGE YES (always done in DB2 9, no longer sets copy pending, look for RC=8!)
- Maximize other utilities' access to objects while copying a list with SHRLEVEL CHANGE and OPTIONS EVENT(ITEMERROR,SKIP)
 - Keeps objects in the list in UTRW state *only* as each object is being copied instead of for the duration of the COPY utility
 - UTRW – utility allows read/write access by applications, but no access for exclusive utilities
- Incremental copy rule-of-thumb: Consider using incremental image copy if
 - <5% of pages are randomly updated (typically means less than 1% of rows updated)
 - <80% of pages are sequentially updated
 - Incremental image copies use list prefetch, so monitor for rid list pool full conditions

- Copy indexes on your most critical tables to speed up recovery

- MERGECOPY – Consider using it

■ RECOVER Best Practices

- PARALLEL keyword provides parallelism for lists of objects (including partitions)
- Compressed pagesets result in faster restore phase
- Enable Fast Log Apply (which can use dual-copy logs) and PAV
- <=10 jobs/member with LOGAPSTG=100MB, 20-30 objects per RECOVER
- For recovery to a prior point in time
 - Always recover related sets of objects together (same RECOVER utility statement)
- Always run REPORT RECOVERY utility on each object and determine the recovery base before running the Recover Utility
- DB2 9 for z/OS: recover to PIT with consistency
 - Backs out uncommitted changes for the objects specified on the RECOVER utility statement
 - Significantly reduces the need to run QUIESCE, which can be disruptive to applications

■ QUIESCE Best Practices

- WRITE NO is less disruptive (no quiescing of COPY=NO indexes)
- Use TABLESPACESET

■ LOAD Best Practices

- LOG NO reduces log volume; if REPLACE, then take inline copy
- KEEPDICTIONARY (track dictionary effectiveness with history statistics PAGESAVE) - small performance impact if loading lots of data

– 254 partition limit for compressed table spaces can be lifted by DBA

- PK51853 shipped new ZPARM MAX_UTIL_PARTS (watch virtual storage)

– Load Partition Parallelism (V7)

- Not individual LOAD part level jobs
- Enable Parallel Access Volume (PAV)

– Index parallelism (SORTKEYS)

- Provide value for SORTKEYS when input is tape/PDS mbr or variable length

- SORTKEYS is the sum of ALL indexes (and foreign keys) on the table

- Remove SORTWKxx / UTPRINxx, and turn on UTSORTAL=YES

– Inline COPY & Inline STATISTICS

– Use REUSE to logically reset and reuse DB2-managed data sets without deleting and redefining them (affects elapsed time)

– When using DISCARD, try to avoid having the input on tape

- Input is re-read to discard the errant records

– Avoid data conversion, use internal representation if possible

– Sort data in clustering order (unless data is randomly accessed via SQL)

– LOAD RESUME SHRLEVEL CHANGE instead of batch inserts

– “LOAD REPLACE SHRLEVEL CHANGE” can be achieved by loading into clone table and then exchanging the tables on DB2 9

– LOAD via Batchpipes to load data that is transferred via FTP from clients

- LOAD via USS Pipes will be provided via PK70269 (still open)

■ REORG Best Practices

– Use SHRLEVEL REFERENCE or SHRLEVEL CHANGE

– Inline COPY & Inline STATISTICS

– KEEPDICTIONARY (track dictionary effectiveness with history statistics PAGESAVE) – large performance impact

– 254 partition limit for compressed table spaces can be lifted by DBA for V8

- PK51853 shipped new ZPARM MAX_UTIL_PARTS (watch virtual storage)

- DB2 9 for z/OS no longer has this limit and uses virtual storage more effectively

– Index parallelism (SORTKEYS is default and ignored in V8)

- Remove SORTWKxx / UTPRINxx, and turn on UTSORTAL=YES

– Run REORG against as many partitions as possible in the same job or against the whole table space to reduce CPU overhead

– Partition parallelism in DB2 9 and NPI processing

- Parallel REORG jobs for same table space but different partitions no longer supported if NPIs defined

- After REORG PART with no BUILD2 phase, no need for REORG NPI

- Watch out for LISTDEFS on partition level with NPIs - full REORG might be more efficient

- Can be prevented by CPU/storage/DD card constraints or when REORG REBALANCE and no NPIs defined

– DIAGNOSE TYPE 66 provides more details

– SHRLEVEL NONE if constrained for disk space

- LOG NO reduces log volume; requires an image copy (inline is a good choice)

- NOSYSREC to avoid I/O (forced for SHRLEVEL CHANGE)

– Take full image copy before REORG SHRLEVEL NONE

- Use REUSE to logically reset and reuse DB2-managed data sets without deleting and redefining them (affects elapsed time)

- SORTDATA NO only if data is already in or near perfect clustering order and disk space is an issue

- Set appropriate PRIQTY/SECQTY to minimize extend processing

- PK60956 helps to improve SORTBLD elapsed time up to 20x for indexes with small SECQTY

- Affects all utilities that are (re-)building indexes

- Run MODIFY RECOVERY some time after ALTER TABLE ... ADD COLUMN

- REORG SHRLEVEL CHANGE (sometimes called online REORG)

- TIMEOUT TERM frees up the objects if timeouts occur in getting drains

- DRAIN ALL (better chance of entering SWITCH phase)

- MAXRO = IRLMRWT minus 5-10 seconds (to prevent timeouts)

- DRAIN_WAIT = IRLMRWT minus 5-10 seconds (to prevent timeouts)

- RETRY = utility lock timeout multiplier (6 by default)

- RETRY_DELAY = DRAIN_WAIT*RETRY

- Enable detection of long running readers (zparm) and activate IFCID 0313 (it's included in STATS CLASS(3))

- This will report readers that may block command and utilities from draining

- It includes “well-behaved” WITH HOLD cursors which a drain cannot break-in on

- Consider scheduling SWITCH phase in a maintenance window to avoid concurrent workloads that may prevent the utility from breaking in:

- MAXRO DEFER and LONGLOG CONTINUE will let REORG do its job except for the last log iteration and the switching

- REORG will continue applying log until MAXRO is changed with the ALTER UTILITY command

- Many log iterations might reduce the “perfect” organization of the table space, so keep the time until MAXRO is changed to allow final processing down to a minimum

■ DFSORT Best Practices

- ◇ Sort work data set allocation

- To direct data sets to storage group, use ACS (see DFSMSrmm SMS ACS Support reference on References slide)

- If assigning data class through ACS, make sure that it's defined as single volume only

- DD refresher for ACS routines: SORTWKnn, SWxxWKnn, DATAWKnn, DAxxWKnn, STATWKnn, STxxWKnn

- Turn off space constraint relief in assigned data class

- ◇ See informational APARs II14047 and II14213 for installation options

- Leave the default for SIZE set to MAX

- Don't bother with changing TMAXLIM (initial storage for each sort)

- If you have to turn a knob.... DSA (Dynamic Size Adjustment)

- You could set this to 128M, but then look to see if DFSORT ever uses this much (needs REGION > 128M!)

- Consider EXPOLD if performance of other applications suffers

- ◇ Use large volumes, >64K tracks supported since z/OS V1R7

- ◇ Background on max size for a sort work dataset

- Prior to z/OS V1R7 -> 64K tracks (on 3390 devices, this is ~3.4GB)

– z/OS V1R7 -> ~million tracks (on 3390 devices, this is ~52.7GB)

– Large volume sizes can reduce the number of sort work datasets

– Example

- Given 40GB of data to sort (so we'll need 48GB of disk space, using the 1.2 X factor)

- If the sort device is a 3390 Model 3 (whose volume capacity is 2.8GB), then at least 18 sort work data sets would be needed to satisfy the sort.

- If instead, the sort device is a 3390 Model 27 (whose volume capacity is around 27GB) and we're running z/OS V1R7, then at least 2 sort work data sets would be needed to satisfy the sort (the limiting value being the size of the 3390 Mod 27 volume).

- ✧ DFSORT in z/OS 1.10 adds support to change installation options on the fly through concatenated PARMLIB members using the ICEOPT command, e.g. to adapt to different workloads (PK59399)

- ✧ Add //SORTDIAG DD DUMMY to your JCLs to include additional diagnostic information in DFSORT output (~10 lines)

- ✧ DB2/DFSORT determine data set size, no hard coded DDs with size specification that need to be maintained

- ✧ Single JCL (template) can be used for most utility jobs

- ✧ DB2 can determine degree of parallelism according to available resources

- ✧ BUT:

–Need to specify SORTNUM, but one size does NOT fit all

- Different objects being processed by same job template

- Different sorts within same utility, e.g. REORG with data and index sorts

–DASD situation varies, SORTNUM 4 might work today, but tomorrow even SORTNUM 8 might fail

–DB2's estimate sometimes not good enough

- ✧ PTFs shipped 02/2008 to enable DB2 to dynamically allocate sort work data sets in utilities:

– DB2 for z/OS V8: PK45916 / UK33692

– DB2 9 for z/OS: PK41899 / UK33636

– Enable with UTSORTAL=YES

– Used for all sorts in utilities: LOAD, REORG, CHECK INDEX, REBUILD INDEX, CHECK DATA, RUNSTATS

– Message “DSNU3340I - UTILITY PERFORMS DYNAMIC ALLOCATION OF SORT DISK SPACE” indicates use

– New behavior ignored if hard coded DD cards are found

- ✧ No more need to specify SORTNUM. Existing SORTNUM specification can be honored or ignored (IGNSORTN=YES)

- ✧ Data sets for largest sorts are allocated first

- ✧ Attempts to allocate data sets as large as possible (starting with 2 data sets per sort task, more data sets allocated if necessary)

■ **MODIFY RECOVERY Best Practices**

- ✧ Run MODIFY RECOVERY regularly to clean up old records in SYSCOPY and SYSLGRNX

- ✧ DB2 9 has RETAIN LAST n or GDGLIMIT

- ✧ Also resets “ALTER_ADD_COLUM” flag in OBD when deleting image copies with previous row versions

–MODIFY RECOVERY DELETE AGE/DATE to delete everything before the REORG that follows the ALTER

–Will make next REORG more efficient if no more old row versions exist

- ✧ Remember that MODIFY RECOVERY works on day boundaries

- ✧ REBUILD INDEX

– Indexes are built in parallel

- Remove SORTWK_{xx} / UTPRIN_{xx}, and use UTSORTAL=YES
- Inline STATISTICS
- Use REORG INDEX SHRLEVEL CHANGE to move index data sets to different volumes
- CREATE INDEX DEFER followed by REBUILD INDEX if not unique IX
 - As of V8, dynamic SQL will not select the index until it is built
- DB2 9 allows SHRLEVEL CHANGE
 - Indexes are put in RBDP
 - Not suited for unique indexes as uniqueness can not be checked during rebuild process, so no INSERT/ UPDATE/ DELETE allowed that affects unique index
 - No parallel jobs on different indexes of the same table space -> use single job with multiple indexes specified

DB2 V9 Index Enhancement

百硕资深工程师 贺明



INDEX 设计对于 DB2 用户的应用性能至关重要, DB2 V9 对于 INDEX 做了很大改进, 主要引入了 Larger Index Page Sizes, Index Compression, Asymmetric Split, Random Index, Index on Expression 等新技术。这些新技术的应用对于减少应用响应时间, 解决 INDEX 热点, 提高直接索引访问都非常有帮助。

本人于今年 10 月 25 日至 10 月 31 日参加了在美国拉斯维加斯举办的 IBM IOD 2008 会议 (Information On Demand 2008)。在此次大会上, 来自 IBM 实验室的各位专家详细介绍了 DB2 V9 的新功能, 近距离地探讨了关于这些 DB2 新功能的使用方法和注意事项。在这里, 我将与大家分享和应用性能最为密切的 INDEX 在 DB2 V9 所提供的新技术, 希望通过介绍这些新技术, 帮助大家了解在 DB2 V9 环境下如何更好地使用 INDEX, 从而进一步提高 DB2 应用性能, 减少系统整体开销。

1. Larger Index Page Sizes

- Usage
 - ✓ Prior to DB2 9, indexes could only utilize 4K pages
 - ✓ In DB2 9 NFM, 4K, 8K, 16K, or 32K index pages are supported
 - ✓ The BUFFERPOOL specification is used to control the page size of the index.
 - ✓ ALTER BUFFERPOOL is used to change an index's page size. The index is set REBUILD PENDING.
- Benefit
 - ✓ GETPage counts for index scanners can be reduced
 - ✓ Larger page sizes can reduce the number of levels in an index - Reduces traverse GETPage costs
 - ✓ Larger page sizes can also reduce index splitting activity - Good for Ordered

Insertions

- ✓ Non-Leaf bufferpool space consumption is reduced-If you fit more non-leafs into the bufferpool, then I/Os may be reduced

◆ Concerns

- ✓ For Random Accesses, you may actually increase the amount of I/O - Reading a larger page for a single entry is more expensive
- ✓ In High Update environments, you may hit more contention on individual leaf pages - More entries per page could mean hotter pages

2. Index Compression

- Usage
 - ✓ Prior to V9, DB2 only supported compression for tablespaces
 - ✓ In V9 NFM, You enable compression by CREATing or ALTERing the index COMPRESS YES in whole index, not partition level.
 - ✓ Index compression will not take place immediately and the index will be placed in advisory REORG-PENDING state. Index compression will not be in effect until the REORG INDEX, REBUILD INDEX, or REORG TABLESPACE utility is run.
 - ✓ Index Compression is not supported in some cases:
 - ✧ For versioned indexes
 - ✧ For indexes in 4K bufferpools
- How Compress work
 - ✓ No compression dictionaries
 - ✧ Compression is done on the fly
 - ✧ Indexes will be compressed when initially populated
 - ✓ Only leaf pages are compressed
 - ✓ In the bufferpool, pages are physically expanded to the page size of the bufferpool. On disk, the pages are always 4K
 - ✓ DB2 ensures that the bufferpool pages will fit

onto the pages on disk

- ✓ The choice of bufferpool is very important
 - ✧ Too small and you limit your compression ratio
 - ✧ Too large and you waste bufferpool space
 - ✧ DSN1COMP helps you choose between compression ratio & unused bufferpool space.
- ✓ Workload pattern of the index compress
 - ✧ Pages are decompressed on read and compressed on write
 - ✧ Synchronous I/Os for Random Accesses may incur synchronous CPU penalties - Larger pages take longer to decompress
 - ✧ For in-memory index (high bp ratio), compression overhead is minimal
 - ✧ Prefetch I/O for index scanners can decompress asynchronously - Still uses CPU, but doesn't affect elapsed time. In fact, compression may improve elapsed time for I/O bound queries.

➤ Rules of Thumb

- ✓ Consider using compression for warehouse application
- ✓ Be ware of CPU overhead for OLTP applications

3. Asymmetric Split Enhancement

➤ Usage

- ✓ Prior to V9, DB2 had Symmetric Split (50/50 index split) & asymmetric split (100/0 index splits).
 - ✧ Symmetric Splits occur within insert into middle of an index
 - ✧ Asymmetric splits occur when inserting the largest/smallest key.
- ✓ Asymmetric split enhancements in DB2 9 fix many of these cases
 - ✧ Better space utilization is realized
 - ✧ Reduce need for index reorgs
 - ✧ Better performance
 - ✧ High concurrency, reduced contention, cool off hot spots
 - ✧ Algorithm was improved for better data

sharing support with PTF UK39357

4. Random Index

➤ Usage

- ✓ Prior to V9, Only ASCending & DSCending indexes were supported.
- ✓ DB2 V9 allows a new RANDOM column ordering specification
- ✓ Not Supported for NOT PADDED varying length columns
- ✓ The Random Key Columns are scrambled on insertion into the index
- ✓ Randomized index values can still be used for equality lookups
- ✓ Range scans are not supported
- ✓ Useful for avoiding some hot spots
 - ✧ Page split hotspots
 - ✧ P-Lock Contention in Data sharing

5. Index on Expression

➤ Usage

- ✓ Prior to V9, we indexed only column values
- ✓ DB2 V9, we can index expressions on the table
- ✓ Simple expressions can be used to compose the index keys
- ✓ Queries containing the same expressions can utilize these indexes
- ✓ The index stores the results of the expressions
 - ✧ The columns used in the expressions are not stored
 - ✧ If an index is created with the UNIQUE option, Uniqueness will be enforced against the expression results
- ✓ There are some restrictions for indexes with expressions
 - ✧ Indexes on expression cannot be used for clustering
 - ✧ DESC/RANDOM may not be specified
 - ✧ You may not alter add columns to indexes with expressions
 - ✧ Primary/Foreign keys with expressions are not supported

~~~~~

# 使用 DFSORT 处理 DCOLLECT 的数据集信息



百硕资深工程师 邹杰

在应用系统的运行中，数据集的使用情况需要密切监控，如发生扩展的次数、跨卷数量的多少、分配空间的大小等，否则会出现扩展次数过多导致性能下降、Volume Count 超出 Data Class 设定的限制等问题。为了避免这种情况的出现，我们可以使用 DCOLLECT 功能来收集数据集的相关信息，提前了解当前数据集的使用情况，对关注的重点数据集进行相应的调整。

DCOLLECT 是 DFSMS 的一项功能，它所收集的信息很多，包括数据集信息、VSAM 数据集信息、卷的信息以及 SMS 配置信息等等。这些信息通过提交一个 IDCAMS 作业，使用 DCOLLECT 命令就可以收集到一个顺序文件中，例如：

```
//GENDAT EXEC PGM=IDCAMS, REGION=4M
//SYSPRINT DD SYSOUT=*
//OUTDS DD DSN=userid.DCOLLECT.DATA,
// RECFM=VB, LRECL=388, DSORG=PS, UNIT=3380,
// SPACE=(1, (100, 100), RLSE), AVGREC=K,
// DISP=(, CATLG)
//SYSIN DD *
DCOLLECT +
  OFILE(OUTDS) +
  VOL(*)
/*
```

需要注意的是，输出顺序文件的格式必须为 V 或者 VB。

如前所述，DCOLLECT 搜集到信息很多，如何能挑选出我们重点关注的数据集呢？

一个简单的方法是使用 DFSORT 就可以对数据集进行筛选。当然，在进行筛选前，我们需要了解 DCOLLECT 输出文件的记录结构，特别是各字段的偏移量、所对应的字段的含义等等。

比如 DCOLLECT 搜集到的关于数据集信息的记录格式如下：

| Offset    | Type      | Length | Name     | Description                   |
|-----------|-----------|--------|----------|-------------------------------|
| 0(X'0')   | STRUCTURE | 24     | DCUOUTH  | DATA COLLECTION OUTPUT RECORD |
| 0(X'0')   | SIGNED    | 4      | DCURDW   | RECORD DESCRIPTOR WORD        |
| 0(X'0')   | SIGNED    | 2      | DCULENG  | LENGTH OF THIS RECORD         |
| 2(X'2')   | CHARACTER | 2      | *        | RESERVED                      |
| 4(X'4')   | CHARACTER | 2      | DCURCTYP | RECORD TYPE FOR THIS RECORD   |
| 6(X'6')   | SIGNED    | 2      | DCUVERS  | VERSION                       |
| .....     |           |        |          |                               |
| 24(X'18') | CHARACTER | 44     | DCDDSNAM | DATA SET NAME                 |
| .....     |           |        |          |                               |
| 77(X'4D') | UNSIGNED  | 1      | DCDNMEXT | NUMBER OF EXTENTS OBTAINED    |

其中偏移量为“4”的两个字符表示该记录的类型，如为“D”则表示为数据集记录，如为“V”则表示为卷信息；偏移量为“24”的 44 个字符表示的是数据集名称；偏移量为“77”的 1 个字节表示的是数据集扩展次数。

我们可以根据这些信息对数据集进行筛选，如按照数据集的名称只选择重点关注的数据集。这可以通过 DFSORT 中的 INCLUDE 语句来实现。例如：

```
INCLUDE COND=(9, 1, CH, EQ, C' D' , AND, 29, 4, CH, EQ, C' ABCD' )
SORT  FIELDS=(29, 44, CH, A, 82, 1, BI, D)
END
```

以上语句表示：仅从 DCOLLECT 记录中挑选出数据集信息的记录，而且数据集名称的前 4 个字符为“ABCD”，然后按照数据集名称和扩展次数进行排序，这样就实现了对数据集的筛选。当然我们也可以根据其它条件做更进一步的筛选。

但 DFSORT 的 INCLUDE 语句中所使用的偏移量与 DCOLLECT 的记录结构并不匹配，这是为什么呢？这与数据集的 VB 格式有关，VB 格式数据集的前 4 个字节为 RDW，再加上 DCOLLECT 的记录结构的偏移量是从“0”开始计算，而 DFSORT 的偏移量应从“1”开始计算，因此在进行排序时，一条记录中相应的偏移量应以“5”为开始位。按照上面的示例，原偏移量为“24”的数据集名称，应从第 29 位开始计算。

现在我们已经获得了相关数据集的信息，如何来处理它们呢？当然我们可以从 DCOLLECT 数据中直接获得，编写一个 REXX 或者其它高级语言的程序都可达到此目的，但 DFSMS 已经为我们提供了一个产生报告的工具，这就是 NaviQuest 的 CLIST 和 REXX，其中 SYS1.SACBCNTL 中的示例作业 ACBJBARD 就可以直接使用 DCOLLECT 数据来产生一个数据集的报告。示例作业如下：

```
//MYLIB JCLLIB ORDER=SYS1.SACBCNTL
//GENREP EXEC ACBJBAOB, PLIB1=SYS1.DGTPLIB, TABL2=IBMSE09.TEST.ISPTABL
//DCOLIN DD DSN=userid.DCOLLECT.DATA, DISP=SHR
//ISPFIL DD DSN=userid.DATASET.REPORT, DISP=OLD
//SYSTSIN DD *
PROFILE PREFIX(IBMSE09)
ISPSTART CMD(ACBQBAR7) +
BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)
/*
//SYSIN DD *
DSN
DSORG
VOLSER
EXTNUM
TITLE=DATA SET REPORT FROM DCOLLECT DATA - xx/xx/xx
SG
LASTREF
ALLOCSP
/*
.....
```

这样我们就可以得到一个数据集信息的报告，仅包含我们选择的数据集。但由于收集的数据是对所有 Volume 进行扫描而得到的，有些数据集是跨卷数据集，因此在报告中同一个数据集可能对应多行报表记录，我们需要进一步进行处理。

下面提供一个示例 REXX 程序，可以把之前产生的数据集信息报告按照数据集名进行合并，对数据集的扩展次数以及分配的空间数量进行计算，使每条记录对应一个数据集的完整信息。

```

/*REXX*/
"EXECIO * DISKR DCIN (STEM INDD. FINIS"
QUEUE "DSNAME, DSORG, NUMVOL, NUMEXT, STORGRP, LASTREF, SIZEALC"

DSNAME = ""
NUMVOL = 0
NUMEXT = 0
SIZEALC = 0

DO A = 1 TO INDD.0

  IF POS("ABCD", INDD. A) > 0 THEN DO

    TMP_DSNAME = WORD(INDD. A, 1)
    TMP_DSORG = WORD(INDD. A, 2)
    TMP_NUMEXT = WORD(INDD. A, 4)
    TMP_STORGRP = WORD(INDD. A, 5)
    TMP_LASTREF = WORD(INDD. A, 6)
    TMP_SIZEALC = WORD(INDD. A, 7)

    IF DSNAME = "" THEN DO
      DSNAME = TMP_DSNAME
      DSORG = TMP_DSORG
      NUMVOL = NUMVOL + 1
      NUMEXT = NUMEXT + TMP_NUMEXT
      STORGRP = TMP_STORGRP
      LASTREF = TMP_LASTREF
      SIZEALC = SIZEALC + TMP_SIZEALC
    END

    IF TMP_DSNAME = DSNAME THEN DO
      DSORG = TMP_DSORG
      NUMVOL = NUMVOL + 1
      NUMEXT = NUMEXT + TMP_NUMEXT
      STORGRP = TMP_STORGRP
      LASTREF = TMP_LASTREF
      SIZEALC = SIZEALC + TMP_SIZEALC
    END

    IF TMP_DSNAME <> DSNAME THEN DO

      IF NUMVOL >= 5 THEN;

        QUEUE DSNAME      || ", " ||,
          DSORG           || ", " ||,
          NUMVOL          || ", " ||,
          NUMEXT          || ", " ||,
          STORGRP         || ", " ||,
          LASTREF         || ", " ||,
          SIZEALC

        NUMVOL = 1
        NUMEXT = TMP_NUMEXT
        SIZEALC = TMP_SIZEALC
        DSNAME = TMP_DSNAME
        DSORG = TMP_DSORG
        STORGRP = TMP_STORGRP
        LASTREF = TMP_LASTREF
        SIZEALC = TMP_SIZEALC
    END
  END

```

```

END

END /* END OF IF WORD INDD */

END /* END OF DO */

/* BEGAIN WRITE LAST RECORD */

IF NUMVOL >= 5 THEN;

QUEUE DSNAME      || ", " ||,
  DSORG           || ", " ||,
  NUMVOL          || ", " ||,
  NUMEXT          || ", " ||,
  STORGRP         || ", " ||,
  LASTREF         || ", " ||,
  SIZEALC

/* END WRITE LAST RECORD */

"EXECIO "      ||,
  QUEUED() ||,
  " DISKW DCOU (FINIS"

EXIT

```

通过这个简单 REXX 的处理，就可以检查出哪些数据集跨卷数量较多，它们的扩展次数是多少，以及分配了多少空间。我们可以周期性地对重点数据集进行检查，及时了解这些数据集的使用情况，提前进行数据集的调整，避免应用运行中出现异常情况。

# SMF 和 PL/I：案例学习

百硕工程师 罗兴魁



## 1. SMF 数据处理概述

IBM 提供了 IFASMFDP 工具程序将 SMF 数据从 MANx 数据集 DUMP 到顺序文件中，该顺序文件具有以下两个特点：

- 首先，记录格式 (RECFM) 为 VBS，这要求处理它的语言需具备读取 SPANNED 记录的能力。

这不难做到，以 QSAM 为例，可在 DCB 中使用 BFTEK=A 参数要求 QSAM 做 SPANNED 记录的 SEGMENT 的拼装。汇编语言及大部分高级语言都可以直接读取 VBS 文件，一个例外是 REXX。

- 其次，逻辑记录的结构较复杂。

SMF 逻辑记录由两部分组成：

- HEADER 部分，包括 RDW
- 不同类型的 SECTION

HEADER 在逻辑记录中的开始位置是固定的，即从 RDW 开始，对某一特定类型的 SMF 记录而言，HEADER 的长度为定长。

各类型的 SECTION 在逻辑记录中的起始位置是变动的，每一类型的 SECTION 又可能出现重复 (Repeat)，即数量为 0-N，SECTION 中可能存在变长的域，因此每个 SECTION 的长度也可能为变长。

SMF 在逻辑记录的 HEADER 中使用三个域来描述 (Triplet) 其它不同类型的 SECTION，内容分别为：

- 该类型 SECTION 在逻辑记录中重复的数量 (0-N)
- 该类型 SECTION 相对逻辑记录起始地址的 OFFSET (RDW 计算在内)
- 该类型 SECTION 在逻辑记录中的总长度

理论上讲，任何一种支持 SPANNED 记录和地址运算的语言都可以用来读取 SMF 数据，但在实践中，如果没有购买专用的 SMF 数据处理工具（如基于 SAS 的 MXG），读取 SMF 数据的语言首选仍然是汇编。

汇编语言的编写与维护 and 高级语言相比要耗费更多的时间与精力，但由于 IBM 只提供 SMF 的 COPYBOOK 汇编版本可以使用，因此除非可以找到一种快捷可靠的获得其它语言版本的 COPYBOOK 的方法，否则汇编语言在 SMF 数据处理上很难被替代。

## 2. PL/X

针对 z/OS 的 DATA AREA（包括 SMF 逻辑记录）的 COPYBOOK，IBM 基本上都提供了两种语言的版本：汇编与 PL/X，因为 IBM 编写这些 COPYBOOK 的方式是使其中一种语言的语句在另外一种语言中被解释为注释，因此两种语言的版本是放在同一个 MEMBER 中。

PL/X 由 PL/S (Programming Language/Systems)，PL/AS (Programming Language/Advanced Systems) 逐渐升级演变而来。PL/S 基于 PL/I，由 IBM 在六十年代研发，其初衷是在主机操作系统的开发中用其替代汇编语言，IBM 于七十年代成功地用 PL/S 改写了 OS360 之后的主机操作系统。

PL/S 的语法与 PL/I 相似，但更加贴近系统底层，比如可以直接操作寄存器，在代码中嵌入汇编语句等。但 PL/S 最大的特点仍在于其编译器无法直接产生目标代码 (Object Code)，PL/S 编译器的输出是汇编源程序！

IBM 一直将 PL/S 系列语言的编译器及相关文档作为商业机密，很少对外界公开其技术细节。但这些语言始终是基于 PL/I，因此在数据定义语句上相差不大，这使得将 PL/X 的 COPYBOOK 按照某些固定的规则转换为 PL/I 版本成为可能。

### 3. 为什么使用 PL/I?

PL/I (Programming Language One) 由 IBM Hursley 实验室在六十年代与 System/360 项目同期开发，除了应用于商业数据处理与科学计算，PL/I 也是较早地被成功用来开发操作系统的高级语言。1964 年麻省理工学院与贝尔实验室用 PL/I 开发了操作系统 Multics，一直到 2000 年 10 月 Multics 才彻底退出历史舞台。

与 COBOL 等高级语言相比，PL/I 有如下更适合用来处理 SMF 数据的特点：

1. PL/I 拥有丰富的数据类型，SMF 记录中的每一种数据类型在 PL/I 中都有对应的数据定义；
2. PL/I 支持指针以及 AREA、OFFSET 和 BASED 类型变量，可以大大简化寻址 SMF 记录各个 SECTION 时的地址运算；
3. PL/I 支持位操作 (BIT OPERATION)；
4. PL/I 有强大的错误处理机制：ON-UNIT，并且使用简单。

此外 PL/I 还有丰富的内置函数 (BUILT-IN FUNCTION)。

因此，即便不考虑 SMF 记录存在 PL/X 版本的 COPYBOOK 因素，那么 PL/I 也较其它语言更加适合做 SMF 数据处理。

### 4. 案例

在这个例子中，我们将读取 SMF 类型 30 子类型 5 的记录，取出每个作业(JOB)的 EXCP 数，并且遍历所有的 EXCP SECTION，从而了解作业的 EXCP 数在每个 DDNAME/DEVICE 上的分布，输出文件各域以逗号分隔，方便导入 EXCEL 做进一步分析。所使用的编译器为 ‘IBM(R) Enterprise PL/I for z/OS V3.R3.M0’。

#### 4.1 展开 PL/X 宏

SMF TYPE 30 记录的 COPYBOOK 存在 SYS1.MACLIB 的 IFASMFR3 中，以宏的形式存在，因此要先用 PL/I 的 PREPROCESSOR 将其展开。

```
//SET1 SET COPYLIB=CCBAP10.LXK.PLIMAC
//      SET CMLPLIB=CCBAP10.IBMZ.ZOS160.SIBMZCMP
//*-----
//PLI EXEC PGM=IBMZPLI, PARM=' +DD:OPTIONS'
//STEPLIB DD DISP=SHR, DSN=&CMLPLIB
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSUT1 DD SPACE=(CYL, (20, 10)), DCB=BLKSIZE=1024
//SYSLIB DD DISP=SHR, DSN=SYS1.MACLIB
//* OUTPUT MDECK
//SYSPUNCH DD DISP=SHR, DSN=&COPYLIB(SMF30PLI)
//* COMPILER OPTIONS
//OPTIONS DD *
NOBJECT, SOURCE, MACRO, INSOURCE, MDECK, NOCOMPILE, NOSYNTAX
//SYSIN DD *
%IFAR30 = 'YES' ;
```

```
DCL SMF30PTR PTR;
%INCLUDE SYSLIB(IFASMFR3);
```

编译器的选项 ‘NOBJECT,NOCOMPILE,NOSYNTAX’ 是为了抑制编译器在展开宏之后做其它工作，如语法检查、编译等，在这一步我们只需要编译器的预处理功能。

选项 ‘MDECK’ 要求将预处理后的结果写入到 SYSPUNCH 中。

SYSIN 中的内容为标准的使用 PL/X 版本 SMF COPYBOOK 的格式，假设我们要生成 TYPE 42 的 COPYBOOK，则应使用如下语句：

```
%IFAR42 = 'YES';
DCL SMF42PTR PTR;
%INCLUDE SYSLIB(IFASMFR4);
```

以%开始的语句为 PL/X（或 PL/I）的 PREPROCESSOR 语句，设置相应的%IFARXX 变量值为‘YES’即可生成类型 XX 的 COPYBOOK。

对 SMF 记录的 HEADER，PL/X 版本的 COPYBOOK 都是用一个单独的 01 层来定义，并 BASED 一个 SMFXXPTR 的指针：

```
DCL 1 SMFRCD30 BASED(SMF30PTR),
```

BASED 变量是 PL/I 的一个重要特色：

```
DCL 1 RECORD BASED(RECPTR),
    2 A CHAR(13),
    2 B CHAR(17);
```

PL/I 不会为 RECORD 这个 STRUCTURE 分配内存，因为它是 BASED 一个地址，随着 RECPTR 所指向地址的变化，A 和 B 也分别指向不同的区域。这和汇编语言中的 DUMMY SECTION 完全对应。

PL/X 的 COPYBOOK 中没有声明(DCL, declare) SMFXXPTR 变量，因此在 SYSIN 中我们声明了 SMF30PTR 这个指针（POINTER）。

由于 PL/X 与 PL/I 在预处理上没有差异，这一步无需对 SYS1.MACLIB 中的 PL/X 宏做任何修改即可通过（返回码为 4）。

## 4.2 转换 PL/X COPYBOOK 到 PL/I

上一步中我们在 SYSPUNCH 中得到了 SMF 30 记录的 COPYBOOK：SMF30PLI。接下来将要对该 MEMBER 做必要的修改，使其成为合法的 PL/I COPYBOOK。

### 4.2.1 定点二进制数（Fixed Binary）

PL/X 中定点二进制数的定义方式如下：

```
5 SMF30SOF FIXED (31),
```

PL/X 没有定点压缩十进制(Fixed Packed Decimal)这种数据类型，因此用 FIXED 关键字即可唯一标示定点二进制数；

但在 PL/I 中存在两种类型的定点数：

```
5 A FIXED BIN(31), /* FIXED BINARY */
5 B FIXED DEC(7), /* FIXED PACKED DECIMAL */
```

对只声明 **FIXED** 属性的变量，PL/I 默认将其解释为压缩十进制（**FIXED DEC**），因此对于 PL/X 中的 **FIXED** 变量，必须要明确加上 **BIN** 属性才能在 PL/I 中得到正确结果。

使用如下 ISPF EDIT 命令即可完成转换：

**C ALL FIXED WORD 'FIXED BIN'**

PL/I 是自由格式的语言，因此以下格式均合法：

```
5 A FIXED BIN(31),
5 A FIXED(31) BIN,
5 A (31) FIXED BIN,
5 A (31)FIXED BIN,
```

关于定点二进制数，另外一个问题牵涉到有符号数与无符号数。在 PL/X 中有这样的数据定义：

```
5 SMF30AFL FIXED BIN(8) ,
5 SMF30RSB FIXED BIN(16) ,
5 SMF30DSV FIXED BIN(32) ,
```

上面三个变量在 PL/X 中分别是 1 字节、2 字节和 4 字节的无符号二进制数，但在 PL/I 中，给这三个变量分配的 **STORAGE** 长度首先取决于其定义是有符号二进制数还是无符号二进制数：

| <b>FIXED BIN SIGNED</b> | <b>FIXED BIN UNSIGNED</b> | <b>Storage Occupied(byte)</b> |
|-------------------------|---------------------------|-------------------------------|
| precision <= 7          | precision <= 8            | 1                             |
| 7 < precision <= 15     | 8 < precision <= 16       | 2                             |
| 15 < precision <= 31    | 16 < precision <= 32      | 4                             |
| 31 < precision <= 63    | 32 < precision <= 64      | 8                             |

从上表中可知，在 PL/I 中对于精度/位数定义是 8、16 和 32 的定点二进制数，作为有符号数和无符号数时分配的 **STORAGE** 是不同的；而对于精度为 64 的定点二进制数，必须定义为无符号数。

在 PL/I 中定义有符号数与无符号数的方法是使用 **SIGNED/UNSIGNED** 属性：

```
5 SMF30AFL FIXED BIN(8) SIGNED ,
5 SMF30RSB FIXED BIN(16) UNSIGNED ,
```

对没有明确指定 **SIGNED/UNSIGNED** 的定点二进制数，PL/I 给的默认值是 **SIGNED**。因此如下变量定义：

```
5 SMF30AFL FIXED BIN(8) ,
```

**SMF30AFL** 在 PL/X 中会分配 1 个字节，在 PL/I 中会分配 2 个字节。为消除这种不一致，需要对 PL/X 中的定点二进制数定义做第二个修改：

对精度定义为 8、16、32 和 64 的定点二进制数，要加上 **UNSIGNED** 属性。即：

```
5 SMF30AFL FIXED BIN(8) UNSIGNED ,
5 SMF30RSB FIXED BIN(16) UNSIGNED ,
5 SMF30DSV FIXED BIN(32) UNSIGNED ,
```

#### 4.2.2 BASED 变量

PL/X 在定义 SMF 记录的 **COPYBOOK** 时，针对 **HEADER** 和每一个 **SECTION** 都定义了一个单独的 01 层，并使用 **BASED** 属性来简化寻址的过程：

```
DCL 1 SMFRCD30 BASED(SMF30PTR) ,
    1 SMF30CMP BASED(ADDR(SMFRCD30)+SMF30TOF) ,
```

只需给指针 SMF30PTR 赋以正确的值，即可访问 HEADER 和 COMPLETION SECTION 中的域，无需再根据 OFFSET 人工做地址运算。

但在 PL/I 中，类似 SMF30CMP 的定义无法通过编译，原因是 PL/I 要求 BASED 变量必须是基于一个 LOCATOR，‘ADDR(SMFRC30)+SMF30TOF’ 这样的地址运算在 PL/I 中是合法的，但得到的结果不是 LOCATOR，不能用于 BASED 变量的基址。

解决方法是用内部函数 PTRADD 来做地址运算：

```
DCL 1 SMFRC30 BASED(SMF30PTR),
    1 SMF30CMP BASED(PTRADD(ADDR(SMFRC30),SMF30TOF)),
```

PTRADD 函数的返回是一个 LOCATOR，可用在 BASED 变量中。

#### 4.2.3 非元素 (ELEMENT) 项

PL/X 版本的 COPYBOOK 中，存在如下的数据定义：

```
3 SMF30DCF BIT(32),
4 SMF30MFL BIT(1),
4 SMF30IIN BIT(1),
4 SMF30TEF BIT(1),
```

该结构 (STRUCTURE) 中，SMF30DCF 属于非元素 (ELEMENT) 项，因其有隶属的更低 LEVEL 的变量。在 COBOL 和 PL/I 中都不允许非 ELEMENT 项存在数据属性 (DATA ATTRIBUTE)，因为两种语言都将结构中的非元素项视作字符定义。

仅仅删除掉非元素项的数据属性定义是不够的，以上面的结构为例，SMF30DCF 长度为 4 字节，由于只有第一个字节高位的三个 BIT 需要涉及，在第四层只定义了这三位，如果仅拿掉 03 层的数据定义，整个 STRUCTURE 的长度就由 4 字节变成了 3 个 BIT，因此需要用类似 COBOL 中的 FILLER 来补齐余下区域：

```
3 SMF30DCF,
4 SMF30MFL BIT(1),
4 SMF30IIN BIT(1),
4 SMF30TEF BIT(1),
4 *          BIT(29),
```

#### 4.2.4 边界对齐 (Boundary Alignment)

汇编、PL/X 与 PL/I 都存在边界对齐，以汇编语言为例，如下定义：

```
A DS 0H
B DS CL1
C DS HL2
```

C 与 B 相邻，如果将 C 的定义略做修改，如下：

```
A DS 0H
B DS CL1
C DS H
```

变量 C 仍然是 2 个字节长，但变量 C 与变量 B 之间会多出一个字节，这是编译器自动插入的用来满足变量 C 的边界对齐要求的字节 (SLACK BYTE)。变量 C 这两次定义的不同就在于第二次的定义方式要求半字对齐 (HALFWORD BOUNDARY)。

因此要做到转换过来的 PL/I 的 COPYBOOK 和 PL/X 版本完全一致，还必须考虑边界对齐的影响，即在某个位置如果 PL/X 的编译器插入了 SLACK BYTES，PL/I 的 COPYBOOK 也必须有这些 BYTES，反过来如果在某个位置 PL/X 的编译器没有插入 SLACK BYTES，PL/I 也不能插入。

可采用如下步骤来确保 PL/I COPYBOOK 与 PL/X COPYBOOK 的精确一致：

- 首先删除掉所有 BDY 关键字。

PL/X 中可以用 BDY 关键字来控制变量的边界对齐：

```
5 SMF30TME FIXED(31) BIN BDY(HWORD),
```

BDY(HWORD)意为要求半字对齐，如要求全字对齐则是 BDY(WORD)。PL/I 不支持 BDY 关键字，因此要将 BDY(...)删除。

- 然后在编译时使用选项 ‘DEFAULT(UNALIGNED)’，要求 PL/I 编译器不对变量做边界对齐。
- 最后要确认 SMF 记录中是否有虚设字节 (SLACK BYTE)。

在编译时使用 UNALIGNED 选项确保了编译器不会在 PL/I COPYBOOK 中插入 SLACK BYTES，但 SMF 记录中是否存在虚设字节 (SLACK BYTES) 仍然未知。假如有，则我们的 COPYBOOK 的 LAYOUT 也会因为缺少虚设字节 (SLACK BYTES) 而出现错误。

虽然 PL/X 的编译器无从获取，但是我们可以通过查看汇编版本的 COPYBOOK 的编译 LIST 来了解 SMF 记录中是否存在编译器自动插入的虚设字节 (SLACK BYTES)。

#### 4.2.5 RDW (Record Descriptor Word)

PL/I 与 COBOL 都无法直接读到变长文件的 RDW，因此要去掉 COPYBOOK 中 RDW 的定义：

```
5 SMF30LEN FIXED BIN(15), /*RECORD LENGTH */
5 SMF30SEG FIXED BIN(15), /*SEGMENT DESCRIPTOR */
```

至此我们已经得到了一个可以通过 PL/I 编译器语法检查的 COPYBOOK。

#### 4.3 样例程序片段

```
DCL SMFDUMP          INPUT    RECORD FILE;
DCL REPORT           OUTPUT   RECORD FILE;
DCL SYSPRINT         PRINT    FILE;

ON ENDFILE (SMFDUMP) BEGIN;
  EOF_SMF='1'B;
  PUT SKIP LIST('* * * * *');
  PUT SKIP LIST('EOF SMFDUMP');
  STOP;
  END;

DCL EOF_SMF BIT(1) INIT('0'B);
```

SMFDUMP 文件 EOF 时程序流程会自动跳转到 ON ENDFILE 段；

```
OPEN FILE(SMFDUMP) INPUT;
OPEN FILE(REPORT ) OUTPUT;

READ FILE(SMFDUMP) SET(SMF30PTR);
```

文件 OPEN 之后做一次 INITIAL READ，使用 LOCATE MODE，将读到的逻辑记录在 PL/I BUFFER 中的地址赋给指针 SMF30PTR；

```
DO WHILE(!EOF_SMF);
  IF (SMF30RTY!=30) | ((SMF30RTY=30)&(SMF30STP!=5)) THEN
  DO;
  READ FILE(SMFDUMP) SET(SMF30PTR); /* READ NEXT */
  ITERATE;
  END;
```

顺序读取，只处理 SMF TYPE 30 SUBTYPE 5 记录；

```
IF (SMF30EON=0) THEN ITERATE;

DO INDX1=1 TO SMF30EON BY 1 ;

RPT_EXCP = SMF30BLK;      /* EXCP COUNT */
RPT_DD   = SMF30DDN;      /* DD NAME   */
RPT_DEVN = HEX(SMF30CUA); /* DEVICE NUM */

WRITE FILE(REPORT) FROM(RPTREC);

SMF30EOF += 30;           /* TO NEXT SEG*/
END;
```

遍历 EXCP SECTION，每处理完一个 SECTION 之后都将 OFFSET 自增 30（一个 EXCP SECTION 的长度为 30），即自动获得下一个 EXCP SECTION 的地址。

## 总结

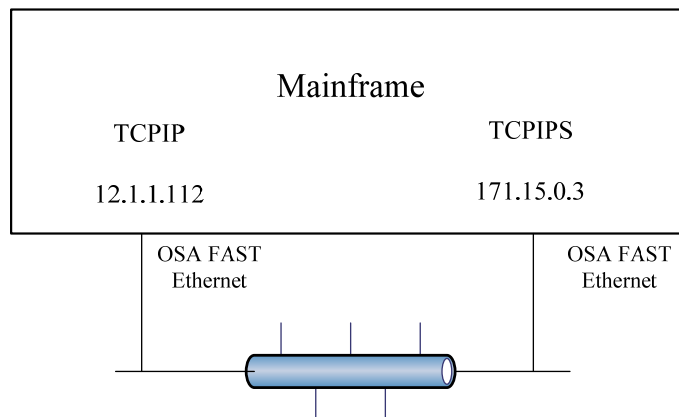
PL/I 读取 SMF 数据的主要障碍是 COPYBOOK 的转换，PL/X 与 PL/I 在数据声明语法上的相似使得这一转换过程并不十分复杂。COPYBOOK 转换就绪后，使用 PL/I 来编写程序进行 SMF 数据处理要比使用汇编语言简单省时得多，因此性能分析者也有机会将更多的精力放在对提取出来的 SMF 数据进行分析上。

# 同一主机分区的多 TCPIP 环境下的网 络连通性检查



百硕资深工程师 罗文军

目前，国内大型银行的主机环境，基本上都是SYSPLEX。在SYSPLEX环境上，可以将每个LPAR分区上的TCP/IP组成组来使用，从而提高TCP/IP应用的可用性。通常每个分区上只有一个TCP/IP Stack。与此有所不同的是，笔者曾经历了同个主机分区上的多个TCP/IP环境，并遇到了在此环境下的网络检查案例。为了同个OS/390主机系统上不同的TCP/IP各自使用独立的TCP/IP Stack，我们在同一主机OS/390系统上启动了两个TCP/IP Stack，启动过程名分别为TCPIP和TCPIPS。其中，TCPIP供TN3270等应用使用，而TCPIPS供某个第三方产品软件专用。为了使两套TCP/IP应用互不影响，我们将两个TCP/IP Stack各自通过单独的OSA卡与外部局域网连接。TCPIP和TCPIPS所用子网分别是12.1.1.x、171.15.0.x，两个TCP/IP Stack的HOME IP地址分别是12.1.1.112、171.15.0.3。如下图所示：



而后在第三方软件的配置过程中，却出现了TCP/IP的连通性问题。虽然这两个TCP/IP Stack的启动正常，从主机外部的TCP/IP网络的子网终端上可以分别使用“PING”命令连通主机的这两个TCP/IP Stack的HOME IP地址，但是从主机端使用TCPIPS Stack的第三方产品软件却始终不能连通外部对应的子网设备，从主机端使用“PING”命令也不能连通TCPIPS Stack所在的子网设备，以至我们总是反复怀疑TCPIPS的TCP/IP设置是否得当。

通过检查分析，我们判断，之所以出现通讯建立不起来的情况，是因为在第三方软件启动过程中，我们设置了与TCPIP Stack而不是与TCPIPS Stack相对应的TCPIP.DATA数据集。在第三方软件安装设置中，通常在安装说明中有要求在第三方软件启动过程中设置SYSTCPD的DD语句，其中指定了相关TCP/IP Stack的TCPIP.DATA，这样软件的启动过程可以找到所需要的TCP/IP配置。在本案例中，TCPIP Stack配置针对的是12.1.1.x子网，TCPIPS Stack配置针对的是171.15.0.x子网。因此，如果在第三方软件启动过程设置了对应于TCPIP Stack所用的12.1.1.x子网配置信息的TCPIP.DATA，那么启动之后，第三方软件启动过程就无法获得TCPIPS Stack所用的171.15.0.x子网的配置信息，导致无法正常通讯。我们在检查并修改了第三方软件所使用的TCPIP.DATA以后，第三方软件的对外TCP/IP通讯恢复正常。

以下是对本案例检查过程中所涉及TCP/IP技术内容的简单小结。

## 一、关于TCPIP.DATA数据集

TCPIP.DATA最初来源是系统数据集hlq.SEZAINST中的TCPDATA，它既可以以分区数据集的Member形式出现，也可以以顺序数据集的形式出现。TCPIP.DATA是指定TCP/IP Stack和运行于TCP/IP Stack上的TCP/IP Server和Client所采用的TCP/IP配置信息的数据集。在TCPIP.DATA中包含了TCP/IP启动过程名（TCPIPJOBNAME）、TCP/IP HOST名称（HOSTNAME)和TCP/IP动态分配数据集的前缀（DATASETPREFIX）等重要的参数。TCPIP.DATA在所有TCP/IP Server、Client功能初始化时被读取。

以下是启动过程TCPIP所用的TCPDATA定义的主要内容：

```
TCPIPJOBNAME TCPIP
HOSTNAME XYZ145
DATASETPREFIX TCPIP
LOADDBCSTABLES SCHINESE
.....
```

TCP/IP应用根据TCPIP.DATA的信息找寻到相关联的TCP/IP Stack以及所使用的TCPIP PROFILE等，在PROFILE里包含了HOME IP地址、路由等信息。以下是许多主机TCP/IP应用所使用的一些共同的搜索路径：

- 1) //SYSTCPD DD DSN=TCP.TCPPARMS(TCPDATAS)
- 2) Userid.TCPIP.DATA for TSO users or jobname.TCPIP.DATA for batch jobs
- 3) SYS1.TCPPARMS(TCPDATAS)
- 4) TCPIP.TCPIP.DATA (Implicit Allocation) or DEFAULTTCPIPDATA

在许多涉及TCP/IP应用的第三方软件的安装过程中，在安装说明中会有此项要求：设置SYSTCPD的DD语句，指定需要的TCPIP.DATA等。在本案例的多TCP/IP的系统环境中，TCP/IP应用的TCPIP.DATA的正确设置显得更为突出。

## 二、IP系统命令检查

在主机系统网络环境的使用过程中，有多种网络检查手段、各种网络检查命令、不同的检查方法等。在本案例中网络检查命令的作用也尤为突出。

- 1) 在SYSLOG下直接输入IP系统命令“Display TCPIP,,NETSTAT,ROUTE”来查看主机路由。如下所示：

```
D TCPIP, TCPIP, N, ROUTE
EZZ2500I NETSTAT CS V2R10 TCPIP 633
DESTINATION      GATEWAY          FLAGS  REFCNT  INTERFACE
DEFAULTNET       10. 1. 1. 1      UG     000018  ETH1
12. 1. 1. 0       0. 0. 0. 0       U      000000  ETH1
12. 1. 1. 112     0. 0. 0. 0       UH     000000  ETH1
127. 0. 0. 1      0. 0. 0. 0       UH     000002  LOOPBACK
192. 168. 252. 1  0. 0. 0. 0       UH     000000  TCPIPS
192. 168. 252. 2  0. 0. 0. 0       UH     000000  TCPIPS
6 OF 6 RECORDS DISPLAYED
```

在命令中第二个TCPIP（兰色字体）指的是TCP/IP的启动过程名。

另一个TCP/IP的启动过程名是TCPIPS，则检查命令如下所示：

```
D TCPIP, TCPIPS, N, ROUTE
EZZ2500I NETSTAT CS V2R10 TCPIPS 635
DESTINATION      GATEWAY      FLAGS  REFCNT  INTERFACE
127. 0. 0. 1     0. 0. 0. 0   UH     000004  LOOPBACK
171. 15. 0. 0    0. 0. 0. 0   U      000001  ETH2
171. 15. 0. 3    0. 0. 0. 0   UH     000000  ETH2
192. 168. 252. 1 0. 0. 0. 0   UH     000000  TCPIP
192. 168. 252. 2 0. 0. 0. 0   UH     000000  TCPIP
5 OF 5 RECORDS DISPLAYED
```

以上的红色字体显示的是两个不同TCP/IP Stack的HOME IP地址。

## 2) 执行TSO NETSTAT ROUTE命令

在ISPF P.6下执行NETSTAT ROUTE命令，因为缺省主机的TCP/IP启动过程名称是TCPIP，所以如果在TSO命令中不指定TCP/IP启动过程名的话，缺省显示的是启动过程名为TCPIP的对外路由。如下所示：

```
NETSTAT ROUTE
EZZ2350I MVS TCP/IP NETSTAT CS V2R10      TCPIP NAME: TCPIP      16:16:24
EZZ2755I Destination      Gateway      Flags  Refcnt  Interface
EZZ2756I -----
EZZ2757I Defaultnet      12. 1. 1. 1  UG     000022  ETH1
EZZ2757I 12. 1. 1. 0      0. 0. 0. 0   U      000000  ETH1
EZZ2757I 12. 1. 1. 112    0. 0. 0. 0   UH     000000  ETH1
EZZ2757I 127. 0. 0. 1     0. 0. 0. 0   UH     000002  LOOPBACK
EZZ2757I 192. 168. 252. 1   0. 0. 0. 0   UH     000000  TCPIPS
EZZ2757I 192. 168. 252. 2   0. 0. 0. 0   UH     000000  TCPIPS
```

而如果要查看另一个TCP/IP Stack：TCPIPS的路由，则需要是在ISPF P.6下输入以下命令，指定特定的TCP/IP启动过程名：TCPIPS。

```
NETSTAT ROUTE TCP TCPIPS
EZZ2350I MVS TCP/IP NETSTAT CS V2R10      TCPIP NAME: TCPIPS      16:17:52
EZZ2755I Destination      Gateway      Flags  Refcnt  Interface
EZZ2756I -----
EZZ2757I 127. 0. 0. 1     0. 0. 0. 0   UH     000003  LOOPBACK
EZZ2757I 171. 15. 0. 0    0. 0. 0. 0   U      000001  ETH2
EZZ2757I 171. 15. 0. 3    0. 0. 0. 0   UH     000000  ETH2
EZZ2757I 192. 168. 252. 1   0. 0. 0. 0   UH     000000  TCPIP
EZZ2757I 192. 168. 252. 2   0. 0. 0. 0   UH     000000  TCPIP
```

以上的IP系统命令，一般情况下系统人员都会使用，只不过当存在多TCP/IP Stack的情况下，输入命令时要注意针对不同的TCP/IP启动过程，加上指定的TCP/IP启动过程名参数。

## 3) TSO PING命令

从主机系统发出的PING命令如果不被主机网络的防火墙所限制的话，可以用来检查主机和子网终端的网络路径是否连通。不过在本案多TCP/IP的OS/390系统环境下，从OS/390主机系统上直接PING TCPIPS Stack所用的子网171.15.0.x的外部终端IP地址，甚至TCPIPS的HOME IP地址都PING不通。这是因为，对于OS/390系统来说，PING命令只可用于名为TCPIP的启动过程，如果对于名称不为TCPIP的启动过程，这需要主机系统做相应的前期TSO定义设置才可以使用，见“三、通过TSO下多TCP/IP间的切换进行IP系统命令检查”。

注：因为本案例中的操作系统版本是OS/390 V2R10，所以如果没有主机系统的前期TSO定义设置，TSO PING命令无法针对不同的TCP/IP启动过程名使用。但在z/OS 1.4之后的多TCP/IP Stack的系统

环境下，PING命令中可以直接指定TCP/IP的启动过程名。ISPF P.6中的TSO PING命令示例如下：(兰色字体指的是TCP/IP的启动过程名)

```
PING 171.15.0.1 (TCP TCPIPS)
```

这大大方便了多TCP/IP Stack环境下的系统网络检查。

### 三、通过TSO下多TCP/IP间的切换进行IP系统命令检查

我们还可以通过TSO下多TCP/IP间的切换进行多TCP/IP的网络检查。这里的“多TCP/IP间的切换”是指通过TSO分配不同的TCP/IP Stack所用的TCPIP.DATA数据集。我们可通过TSO分配TCPIP.DATA，进行文件名为SYSTCPD的定义，然后再用不带TCP/IP启动过程名的IP系统命令进行主机网络检查。TCPIP.DATA数据集的TSO分配有以下几种途径：

1) 使用TSO ALLOC命令来分配TCPDATA。

通过TSO临时分配一个TCPDATA，然后再在TSO中执行NETSTAT ROUTE命令来检查TSO用户所使用的主机TCP/IP网络。首先执行TSO ALLOC命令分配一个名为SYSTCPD的TCPIP.DATA数据集。如果要在不同的TCP/IP Stack之间切换，用户只需简单地DEALLOCATE当前使用的SYSTCPD，重新分配另一个TCP/IP Stack相应的TCPIP.DATA数据集。以下是ISPF P.6上执行的命令示例：

```
FREE FI (SYSTCPD)
ALLOC FI (SYSTCPD) DA ('TCP.TCPPARMS(TCPDATAS)') SHR
```

我们也可以直接做一个REXX程序来执行以上的动作。

```
/* REXX */
Say 'Switching to the TCPIPS stack'
msgstat = msg()
z = msg("OFF")
"FREE FI (SYSTCPD)"
"ALLOC FI (SYSTCPD) DA ('SYS1.TCPPARMS(TCPDATAS)') SHR"
z = msg(msgstat)
```

2) 使用增加了SYSTCPD语句的TSO Logon Procedure。

在TSO Logon Procedure中增加SYSTCPD DD语句，指向不同的TCPIP.DATA数据集，让使用不同的TCP/IP Stack的TSO用户使用不同的TSO Logon Procedure进行TSO登录。当使用不同的TCP/IP Stack进行切换的时候，需先从当前的TSO Logoff再用另一个的Logon Procedure进行TSO登录。

在通过上述途径分配了特定的对应于某个TCP/IP Stack的TCPIP.DATA之后，我们可以从主机ISPF P.6上发IP系统命令，检查该TCP/IP Stack的主机路由情况、子网外接终端的网络路径，此时的IP系统命令中无须再指定TCP/IP启动过程名称。

以下是执行示例，假设我们通过TSO分配了与TCPIPS Stack相关的TCPIP.DATA。在ISPF P.6下先输入以下命令：

```
FREE FI (SYSTCPD)
ALLOC FI (SYSTCPD) DA ('SYS1.TCPPARMS(TCPDATAS)') SHR
```

这时从ISPF P.6下发NETSTAT ROUTE命令显示如下：

```
NETSTAT ROUTE
EZZ2350I MVS TCP/IP NETSTAT CS V2R10          TCPIP NAME: TCPIPS          16:17:52
EZZ2755I Destination      Gateway          Flags  Refcnt  Interface
EZZ2756I -----
EZZ2757I 127.0.0.1          0.0.0.0         UH     000003  LOOPBACK
EZZ2757I 171.15.0.0         0.0.0.0         U      000001  ETH2
```

|          |               |         |    |        |       |
|----------|---------------|---------|----|--------|-------|
| EZZ2757I | 171.15.0.3    | 0.0.0.0 | UH | 000000 | ETH2  |
| EZZ2757I | 192.168.252.1 | 0.0.0.0 | UH | 000000 | TCPIP |
| EZZ2757I | 192.168.252.2 | 0.0.0.0 | UH | 000000 | TCPIP |

这时从ISPF P.6下发PING命令显示如下：

```
PING 171.15.0.1
EZA0458I Ping CS V2R10:Pinging host 171.15.0.1. Use ATTN to interrupt.
EZA0462I PING:Ping #1 response took 0.003 seconds.Successes so far 1.
***
```

以上命令信息显示，可以从主机系统上经由“PING”命令连通TCPIPS Stack子网的外部终端IP地址171.15.0.1。在前文中提到的“从系统上直接PING TCPIPS子网171.15.0.x的外部终端地址甚至TCPIPS的HOME IP地址都PING不通”的情况此时将不会出现。这是因为，我们已经在TSO中将当前使用的TCPIP.DATA切换到了TCPIPS Stack上了。

通过前面的IP主机系统命令检查，我们看到两个TCP/IP Stack在系统上的路由信息不同，且不同TCP/IP Stack所对应的外部子网路径都是连通的。前文所提到的第三方软件TCP/IP不通的原因是因为从第三方软件发出去的IP数据包没有经过特定的TCPIPS Stack的IP路由而到达目的地。按照此判断，我们检查了第三方软件的相关TCP/IP配置，使问题得到了解决。

# 浅析 LOGSTREAM 的用法

百硕工程师 陈银波



## 概述

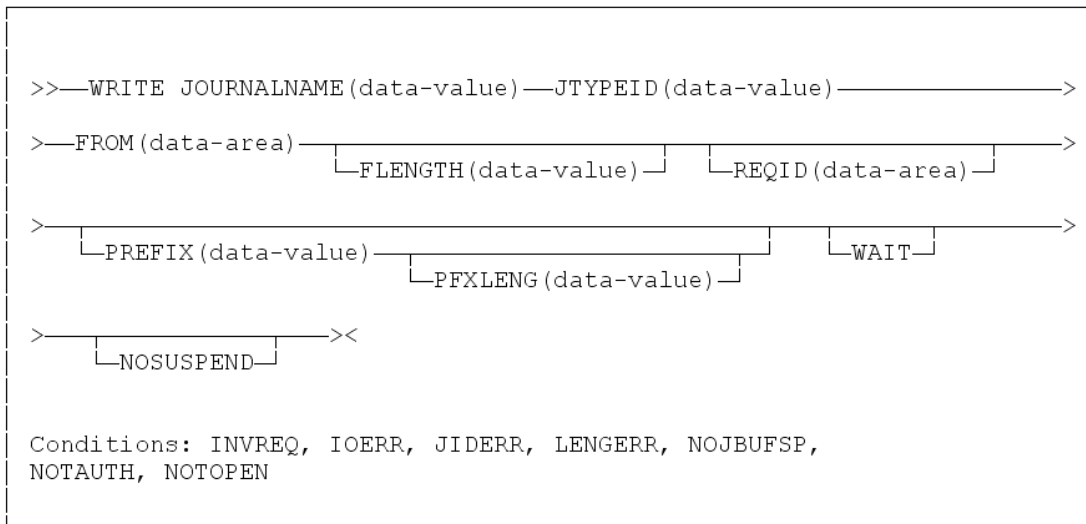
在主机z/OS操作系统中，IBM提供了很多非常有用的系统服务，其中之一就是SYSTEM LOGGER服务。主机的许多子系统都在使用这个服务，完成一些特殊的功能。比如CICS，它的DFHLOG和DFHSHUNT 使用的就是LOGGER服务。当CICS异常宕机时，就是通过读取LOGSTREAM中的信息进行恢复启动的。这个服务，在我们平时主机工具开发或者是某些应用的开发中都会使用到。下面，我们将介绍如何在CICS中将数据写入LOGSTREAM，以及如何通过批量方式将数据从LOGSTREAM中读取和写入。

## 1、程序如何在 CICS 中使用 LOGSTREAM

### 1.1 CICS 定义 LOGSTREAM

在CICS应用程序能够使用LOGSTREAM之前，首先需要对其进行定义，然后才能经由CICS API向LOGSTREAM中写入数据。通过Journal定义，我们可以在CICS中定义一个LOGSTREAM资源。那么，以后CICS中的应用程序只要将数据写入到对应的Journal资源中，数据就会被写入到Journal资源中指定的LOGSTREAM中。

### 1.2 WRITE JOURNALNAME 接口



WRITE JOURNALNAME是CICS提供给用户专门用来写入日志的接口。只要CICS定义了合适的Journal资源，那么用户在应用程序中根据该接口要求输入各个参数进行调用，就能将数据写入LOGSTREAM中。

下表列出了该接口所使用的参数及注释。

| 参数          | 注释                                               |
|-------------|--------------------------------------------------|
| JOURNALNAME | 指定 1 到 8 位的 JOURNAL 名字，该名字需要与所定义的 journalname 匹配 |
| JTYPEID     | 2 位的标识符，来标识该记录来自于哪个应用                            |

| 参数        | 注释                                         |
|-----------|--------------------------------------------|
| FROM      | 指定将写入日志的用户数据                               |
| FLENGTH   | 指定用户数据的长度                                  |
| REQID     | CICS 用来同步的参数，当指定 WAIT 参数后就不需要此参数           |
| PREFIX    | 指定用户 PREFIX 数据，将写入日志中                      |
| PFXLENG   | 指定用户 PREFIX 数据的长度                          |
| WAIT      | 指定同步日志输出。调用该接口的任务将等待，直到日志数据已经被写入 LOGSTREAM |
| NOSUSPEND | 指定该参数后，应用程序不会被暂停，因此用户记录将被忽略                |

调用举例：

以下是调用该接口的例程（例程中没有写入 PREFIX 数据）

COBOL 语言形式：

```
exec cics write journalname(jname)
  JTYPEID('JT')
  from(data-area) wait
end-exec.
```

C 语言形式：

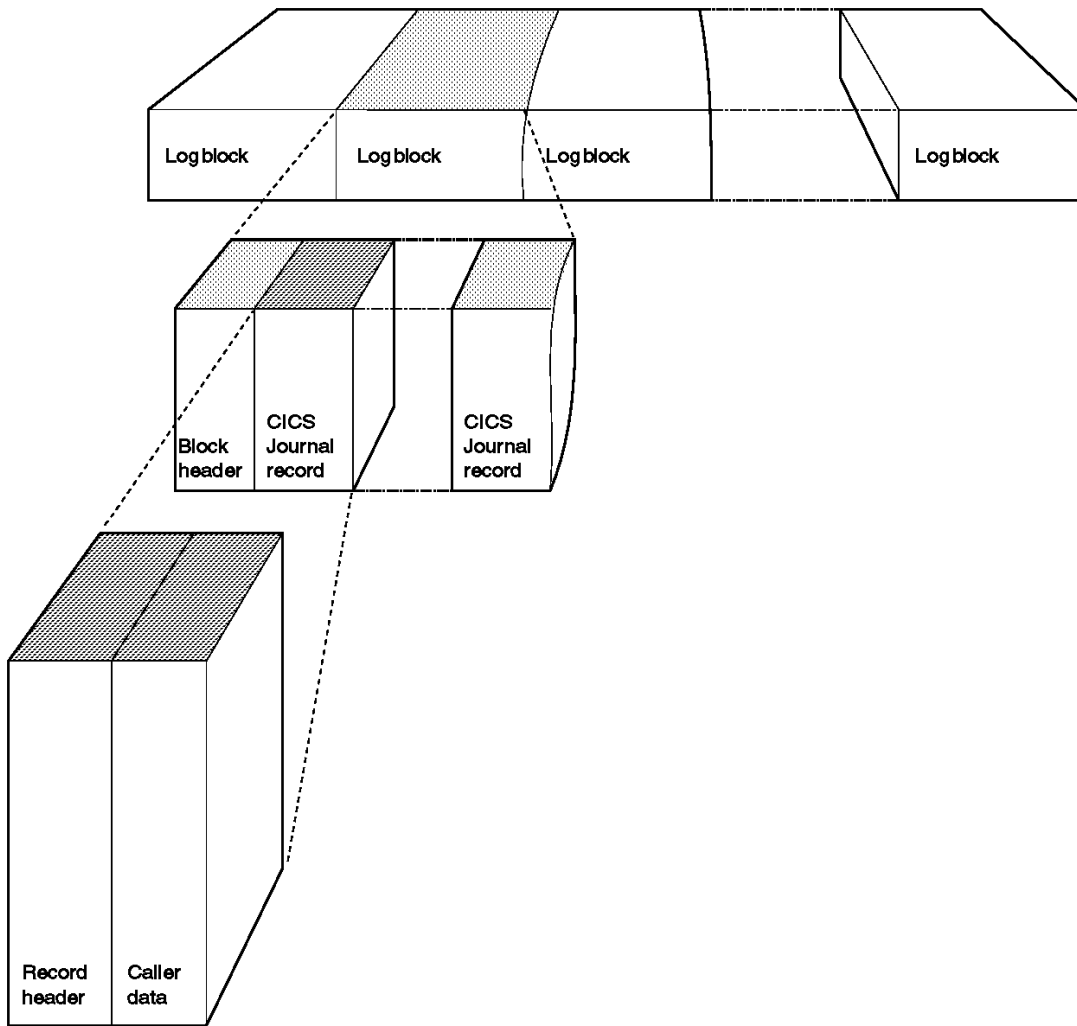
```
EXEC CICS
WRITE JOURNALNAME("TRANLOG") JTYPEID("TL")
FROM(buffer) FLENGTH(len) WAIT;
```

### 1.3 CICS LOG 结构

CICS 没有提供读取 LOGSTREAM 数据的接口，我们通过 WRITE JOURNALNAME 写入的日志数据，只能通过编写外部程序进行读取。如何编写外部 LOGSTREAM 访问程序我们将在下章进行介绍。这里，我们首先需要了解 CICS 写入 LOGSTREAM 的日志数据结构。只有了解了这些数据的结构，我们才能从 LOGSTREAM 中正确地读取到用户日志数据。

CICS LOG 数据是由连续的 LOG BLOCK 组成，每个 LOG BLOCK 包含一个 BLOCK HEADER 和一组 CICS 日志记录（CICS Journal record），记录可能是一个或多个。每个 CICS 日志记录包含一个 RECORD HEADER 和 Caller Data 或者是一个 RECORD HEADER 和 Start-of-run。用户的数据包含在 Caller Data 中，Start-of-run 是每次 CICS 连接 LOGSTREAM 时，第一个写入 LOGSTREAM 的记录。

下图是整个 CICS LOG 的层次结构图：



Note: 更为详细的CICS LOG数据结构介绍，请查看《CICS Customization Guide》中5.1.4章节。

## 2、批量程序如何访问 LOGSTREAM

z/OS 操作系统提供了访问 LOGSTREAM 的各种 macro 接口。通过调用这些接口我们就可以对 LOGSTREAM 实现 DISCONNECT、CONNECT、READ、WRITE、DELETE 操作。

### 2.1 LOGSTREAM macro 接口介绍

对 LOGSTREAM 进行访问操作的宏接口主要有以下几个：

- IXGBRWSE -- Browse/Read a Log Stream
- IXGCONN -- Connect/Disconnect to Log Stream
- IXGDELET -- Deleting Log Data from a Log Stream
- IXGIMPRT -- Import Log Blocks
- IXGWRITE -- Write Log Data to a Log Stream

#### 2.1.1 IXGBRWSE

通过 IXGBRWSE 宏，我们可以浏览和读取 LOGSTREAM 数据。我们既可以从 LOGSTREAM 中连续读取 LOG BLOCK，也可以指定读取某个 LOG BLOCK。执行 IXGBRWSE 时，有以下 5 种请求：

**REQUEST=START** 开启一个BROWSE SESSION，同时创建并返回一个BROWSE TOKEN，这个TOKEN指定了这个SESSION。当程序去读取LOGSTREAM数据时，必须指定TOKEN。BROWSE SESSION一旦开启，只有当执行“REQUEST=END”请求，或者LOGSTREAM DISCONNECT的时候才会关闭。开启一个BROWSE SESSION时还有几个重要参数，它们决定了访问LOGSTREAM的顺序和方式。OLDEST, YOUNGEST参数表示从最老的还是从最新的记录开始BROWSE SESSION；“VIEW=ACTIVE”或者“VIEW=ALL”参数表示访问的是ACTIVE的数据还是整个LOGSTREAM中的数据。

**REQUEST=READCURSOR** 该参数表示读取下一个LOG BLOCK。当参数中指定MULTIBLOCK时，则可以连续读取多个LOG BLOCK。

**REQUEST=READBLOCK** 该参数表示从LOGSTREAM中读取指定的LOG BLOCK。

**REQUEST=RESET** 该参数表示将读取LOGSTREAM的CURSOR重置到LOGSTREAM开始或者结尾。

**REQUEST=END** 该参数表示结束BROWSE SESSION。

### 2.1.2 IXGCONN

IXGCONN用来将程序连接到某个LOGSTREAM，或者将程序与某个LOGSTREAM断开连接。当使用IXGCONN的“REQUEST=CONNECT”请求连接到某个LOGSTREAM时，IXGCONN会返回一个STREAM TOKEN，其他LOGGER 服务宏需要通过这个STREAM TOKEN来访问该LOGSTREAM。如果多个应用程序连接到同一个LOGSTREAM，那么它们写入的LOG BLOCK将会被合并到一起。

### 2.1.3 IXGDELET

顾名思义，IXGDELET宏用来删除LOGSTREAM中的LOG BLOCK。

### 2.1.4 IXGIMPRT

IXGIMPRT宏可以将一个LOGSTREAM中的LOG BLOCK拷贝到另外一个LOGSTREAM中。

### 2.1.5 IXGWRITE

IXGWRITE用于将LOG BLOCK写入到一个LOGSTREAM中。

参考书目：如果读者希望深入了解以上各接口参数更详细的信息，请参考以下书目：

《z/OS: MVS Programming: Authorized Assembler Services Reference, Volume 2 (ENFREQ-IXGWRITE)》中有接口参数的更详细介绍和例程。

《z/OS: MVS Programming: Assembler Services Guide》中的27章，详细介绍了如何使用LOGGER服务。

## 2.2 编写汇编接口

虽然z/OS已经提供了各种访问LOGSTREAM的接口，但是这些接口都是提供给汇编代码使用的。而一般批量应用程序都是使用高级语言开发的，比如C、COBOL等，它们是无法直接使用这些接口的。因此，为了让高级语言也能使用LOGGER服务，我们首先必须将这些接口整合到一起，编译成可供高级语言调用的模块，然后C、COBOL等语言调用这些模块，从而达到使用LOGGER系统服务的目的。文章中将介绍两个供C和COBOL调用的汇编模块，它们都是使用汇编编写，将系统提供的LOGSTREAM访问接口整合到自己的模块中，最终供其他语言调用。

## 2.2.1 汇编模块介绍

### 模块名称：CON2LS

功能：用来 CONNECT、DISCONNECT 一个 LOGSTREAM

描述：CON2LS 函数是将 IXGCONN 整合起来，根据 action 参数的不同，进行 LOGSTREAM 的 CONNECT 或者 DISCONNECT 操作。

该模块的 C 调用形式是：

```
void CON2LS(char *streamname,char *token,int *retcode,int *rsncode,char *action)
```

参数介绍：

- streamname 必须是 26 字节的 logstream 名字，不够长的以空格补齐
- token 是函数执行正确后返回的 16 字节的 stream token
- retcode、rsncode 是函数执行时的 return code 和 reason code
- action 是指函数执行的操作，是“CONNECT”，还是“DISCONNECT”

COBOL 调用形式：

```
CALL 'CON2LS' using streamname token retcode rsncode action
```

例程：

比如，我们需要连接到 LOGSTREAM MY.LOGSTR，那么 C 程序可以编写如下：

```
#pragma linkage (CON2LS, 0S)
...
char token[16];
char streamname[26];
int retcode,rsncode;
char action[10];
.....
strcpy(streamname, " MY.LOGSTR          ");
strcpy(action, " CONNECT" );
CON2LS(streamname, token, &retcode, &rsncode, action);
...

```

COBOL 程序可以编写如下：

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ACTION          PIC X(11)  VALUE 'CONNECT '.
01 TOKEN           PIC X(16).
01 STREAMNAME      PIC X(26) value space.
01 retcode         pic s9(8) binary value zero.
01 rsncode         pic s9(8) binary value zero.
.....
move 'CICSLOG.TRANSLOG' to streamname.
call 'CON2LS' using streamname token retcode rsncode action.
...

```

CON2LS 汇编源程序可以到我的 MSN 共享空间下载：

<http://cinbo.spaces.live.com/blog/cns!3F94EB317090B230!144.entry>

**模块名称：OPERLS**

功能：对 LOGSTREAM 提供 START、READ、DELETE、WRITE 功能

描述：OPERLS 是将 IXGBRWSE、IXGDELET、IXGWRITE 三个函数整合起来，根据 action 参数的不同，调用不同的 LOGSTREAM 接口函数，从而实现对 LOGSTREAM 的读，写，删除日志数据的操作。

OPERLS 的 C 调用形式：

```
void OPERLS(char *token,int &retcode,int &rsncode,char *action,struct *operdata)
```

- token 是 16 字节的 streamtoken，是 CON2LS 产生的
- retcode、rsncode 是 return code 和 reason code
- action 包含 'start','read','delete','write','end' 等操作
- operdata 是一个结构，C 的定义形式：

```
struct {
    char brstoken[4];
    char retblkid[4];
    char timestmp[16];
    int blksize;
    char buffer[60000];
}
```

operdata 在 COBOL 中的定义形式：

```
01 operdata.
    05 brstoken          pic x(4).
    05 retblkid          pic x(8).
    05 timestmp         pic x(16).
    05 blksize          pic s9(8) binary value zero.
    05 buffer           pic x(60000).
```

COBOL 调用形式：

```
CALL 'OPERLS' using token retcode rsncode action operdata
```

Action 操作介绍：

- ✓ 当 action 是 “START” 时，函数会将 BROWSE TOKEN 放入结构的 brstoken。
- ✓ 当 action 是 “READ” 时，brstoken 需要填入 BROWSE TOKEN，同时其他 4 个参数都会被填上返回值。retblkid 是 LOG BLOCK 的 BLOCKID，timestmp 是 LOG BLOCK 的 time stamp，blksize 是 LOG BLOCK 的数据大小，buffer 是 LOG BLOCK 的数据存放位置。
- ✓ 当 action 是 “delete” 时，结构中只有 retblkid 有效，该操作将删除 retblkid 指定的 LOG BLOCK。
- ✓ 当 action 是 “WRITE” 时，结构中只有 retblkid、blksize、buffer 有效。buffer 中存放用户数据，blksize 指定数据的大小，retblkid 是写入 LOG 数据时返回的 BLOCKID。

例程：

OPERLS 使用举例，比如，我们 START 一个 BROWSE SESSION，那么程序可以编写如下：

```
#pragma linkage(OPERLS, OS)
struct read_data{
    char brstoken[4];
    char retblkid[8];
    char timestmp[16];
    int blksize;
    char buffer[62500];
} operdata;
...
char token[16];
int retcode, rsnocode;
...
strcpy(action, "START");
OPERLS(token, &retcode, &rsnocode, action, &operdata);
...

```

COBOL 程序可以编写如下：

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ACTION          PIC X(11)  VALUE 'CONNECT '.
01 TOKEN           PIC X(16).
01 STREAMNAME     PIC X(26) value space.
01 retcode        pic s9(8) binary value zero.
01 rsnocode       pic s9(8) binary value zero.
01 operdata.
    05 brstoken    pic x(4).
    05 retblkid    pic x(8).
    05 timestmp    pic x(16).
    05 blksize     pic s9(8) binary value zero.
    05 buffer      pic x(60000).
.....
move 'START' to action.
call 'OPERLS' using token retcode rsnocode action operdata.
.....

```

OPERLS 汇编源代码可以到我的 MSN 共享空间下载：

<http://cinbo.spaces.live.com/blog/cns!3F94EB317090B230!145.entry>

## 2.3 编程实例

了解了 CICS LOG 的结构，以及拥有了访问 LOGSTREAM 的高级语言接口之后，我们就可以编写批量程序，读取在 CICS 中写入 LOGSTREAM 的日志数据。

我们以主机中最为常用的 COBOL 语言为例，通过使用以上两个汇编模块，编写一个读取 CICS LOG 的程序。该程序将连接到系统的 LOGSTREAM CICSLOG.TRANSLOG，然后从 LOGSTREAM 读取从 CICS WRITE Journalname API 写入的日志数据。

源程序代码可以到我的 MSN 共享空间下载：

<http://cinbo.spaces.live.com/blog/cns!3F94EB317090B230!146.entry>

## 总结

通过实例编程实现应用程序使用系统提供的SYSTEM LOGGER服务，不仅可以使我们了解如何使用系统提供的LOGSTREAM访问函数，同时也能使我们加深对SYSTEM LOGGER这一系统部件的理解。

**百硕客户通讯 BAYSHORE ADVISOR**

中国主机用户专享的资讯季刊

2008 年 12 月 1 日出版（总第 14 期）

主办：百硕同兴科技（北京）有限公司

出版：百硕客户通讯编委会

吕宁

郭庆雪

李琰

马彤雷

王晓兵

刘京平

吴筋筋

张凤华

陈银波

陈建

邹杰

罗文军

贺明

徐卫华

高春霞

高大川

高玉超

郑霞

康会影

Darryn Salt

Martha Hall

地址：北京市朝阳区望京科技园利泽中二路 1 号中辰大厦 209 室

电话：010 64391733 传真：010 64391582

电子邮箱：Bayshore\_advisor@bayss.com

如果您对百硕客户通讯有任何意见和建议，欢迎您随时与我们交流！



百硕同兴

百硕客户通讯总第 14 期（2008 年 12 月 1 日）

百硕同兴科技（北京）有限公司

Bayshore Consulting & Services Co., Ltd.