iSeries

IBM

Job Accounting

Experience Report

iSeries

Job Accounting

© Copyright International Business Machines Corporation 2004. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Job Accounting

Job accounting is a function available on every iSeries(TM) Server that can be used to track usage of system resources. The journal accounting information system value (QACGLVL) determines what type of system usage information is journaled to the system accounting journal (QACGJRN). The type of information that can be collected includes processing unit time used (CPU), transaction counts, time job was active, number of database operations, communications operations, and number of printed pages. A full list of fields found in the accounting journal entries can be found in “Tables: JB, DP, and SP journal entries” on page 4. This information can be used to track resource usage for a particular user or group of users. Queries or application programs can be written to use this data for several purposes such as charging users for use of system resources, analyzing performance, or statistical analysis. This experience report covers how to set up job accounting, special considerations for prestart or server jobs, and how to look at the data after it has been collected.

The following sections contain additional information:

Setting up Job Accounting

The job accounting function is not active by default. It requires a few initial steps to set it up:

1. Create a journal receiver by using the Create Journal Receiver (CRTJRNRCV) command.
   The journal receiver can be created with any name and library you choose, but it is recommended to give it a name with a naming convention such as ACGJRN1 so that additional receivers (such as ACGJRN2, ACGJRN3) can be created with the CHGJRN JRNRCV(*GEN) command.

   ```
   CRTJRNRCV JRNRCV(USERLIB/ACGJRN1)
   ```
2. Create the job accounting journal using the Create Journal (CRTJRN) command. The journal name must be QSYS/QACGJRN, and you need authority to add objects to the QSYS library.

   ```
   CRTJRN JRN(QSYS/QACGJRN) JRNRCV(USERLIB/ACGJRN1) AUT(*EXCLUDE)
   ```

   The journal receiver should be the same as the receiver created in step 1. The authority can be set to anything you choose, but *EXCLUDE is recommended since the data collected could be used to charge users for resource usage.
3. Change the journal accounting information system value (QACGLVL) using the Work with System Values (WRKSYSVAL) or Change System Value (CHGSYSVAL) commands.

   ```
   CHGSYSVAL SYSVAL(QACGLVL) VALUE('*JOB *PRINT')
   ```

   The system value can be set to journal job accounting information, or printer information, or both. The full list of fields collected are listed in the “Tables: JB, DP, and SP journal entries” on page 4 section. *JOB produces job (JB) journal entries, while *PRINT produces direct print (DP) or spooled print (SP) journal entries. A value of *NONE means no journaling is done for journal QACGJRN. Job accounting data will only be journaled for jobs that are started after the system value has been set to a value other than *NONE.

4. Use the Change User Profile (CHGUSRPRF) or Create User Profile (CRTUSRPRF) commands to set the accounting code parameter (ACGCDE) for each user profile. The accounting code can be set to any alphanumeric string up to 15 characters in length. If determining the current user is important in your analysis of a job accounting journal entry, it is recommended that you set the ACGCDE parameter to the user profile's name.

   ```
   CHGUSRPRF USRPRF(USERID1) ACGCDE(USERID1)
   ```

   The accounting code can also be specified for a group of users by using the Change Job Description (CHGJOBD) or Create Job Description (CRTJOBD) commands. The default accounting code for job descriptions is *USRPRF, which means it uses the accounting code from the job's user profile. If a value other than *USRPRF is specified in the job description, it will take precedence over the accounting code specified in the user profile. A simple program could be written to change all the user profiles on a system so their accounting code matches their user profile name. An "Example: CL Program to change accounting code in user profiles" on page 7 has been provided to get you started.

Displaying the data collected

After collecting data in the job accounting journal (QACGJRN), complete the following steps to write the journal entries to a file and display them:

1. Create a copy of the system supplied model outfile for the accounting journal using the Create Duplicate Object (CRTDUPOBJ) command. QAJBACG4 is the model outfile for the *TYPE4 outfile format.

   ```
   CRTDUPOBJ OBJ(QAJBACG4) FROMLIB(QSYS) OBJTYPE(*FILE) TOLIB(QTEMP)
     NEWOBJ(MYJBACG4)
   ```

2. Use the Display Journal (DSPJRN) command to dump the journal entries to the outfile just created in the previous step. In this example, we are only dumping the 'JB' or job type journal entries.

   ```
   DSPJRN JRN(QACGJRN) ENTTYP(JB) OUTPUT(*OUTFILE) OUTFILFMT(*TYPE4)
     OUTFILE(QTEMP/MYJBACG4)
   ```

3. Start an SQL session using the Start Structured Query Language (STRSQL) command. Then use the SELECT command from within the SQL session to choose the fields you want to display. A list of field names can be found in "Tables: JB, DP, and SP journal entries" on page 4. The same information can also be displayed interactively or to a file by creating and running a query using the Work with Queries (WRKQRY) command.

   ```
   STRSQL

   SELECT JAJOB, JAUSER, JAUSPF, JACDE, JACPU FROM QTEMP/MYJBACG4
   ```

   A sample of the SQL output for the job accounting journal is found in figure 1.

JOB NAME	USER NAME	USER PROFILE	ACCOUNTING CODE	CPU
QPADEV0001	JANEDOE	JANEDOE	JANEDOE	2519
BACKUP	QPGMR	QPGMR	QPGMR	55071
BACKUP	QPGMR	BKUP15	BKUP15	1087
QZRCSRVS	QUSER	SMITH	QUSER	1343
QZRCSRVS	QUSER	SMITH	SMITH	53
QZRCSRVS	QUSER	QUSER	SMITH	5070

```
QZRCSRVS          QUSER             QUSER               QUSER                         23
```

Figure 1. Sample journal output using SQL

Analyzing the data

When analyzing the journal entries, it is important to understand how and when journal entries are written. A JB journal entry is written to the job accounting journal for a job any time the job accounting code is changed and when the job ends. Therefore, one job may have multiple journal entries. The accounting code can be changed by one of the following methods:

- Use the Change Accounting Code (CHGACGCDE) command.
- Use the ACGCDE parameter on the Change Prestart Job (CHGPJ) command.
- Call the Change Job (QWTCHGJB) API and specify either the 1001 key (job accounting) or the 2701 key (all keys for JOBC0300 format).

If the accounting code is changed, a journal entry will be written containing information about the resources used by the job since it started, or since the accounting code was last changed. When the job ends, the journaled information will be for the time period since the accounting code was last changed until the job ends. For example, Figure 2 illustrates a job with two accounting segments:

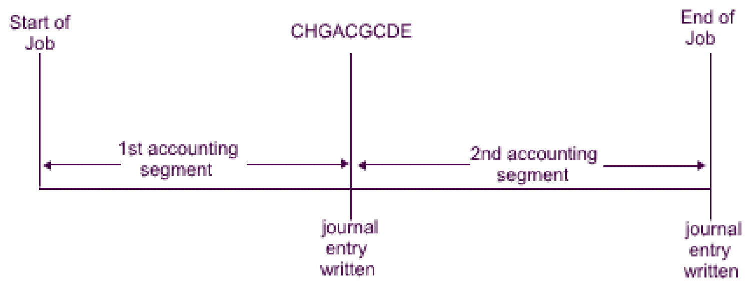

Figure 2. Job with 2 accounting segments

Prestart jobs and server jobs that use prestart jobs present a unique situation for job accounting. These types of jobs are usually configured to start with a generic user profile such as QUSER, and then they wait for a request to be handled. When a prestart job is given a request to handle, the job swaps user profiles using the Set Profile Handle (QWTSETP) API to that of the requester, services the request, and then swaps back to the initial user profile. If the prestart job is configured to be reused (MAXUSE parameter on the Add Prestart Job Entry (ADDPJE) or the Change Prestart Job Entry (CHGPJE) command is greater than 1) the job will wait for another request and repeat the above scenario. In this case, a single prestart job could potentially service many different users. If you want to be able to charge each of these users for their resources used, the accounting code needs to be updated before and after each service request. System-defined server jobs already do this for you. Figure 3 illustrates a prestart job which services one user and then ends.

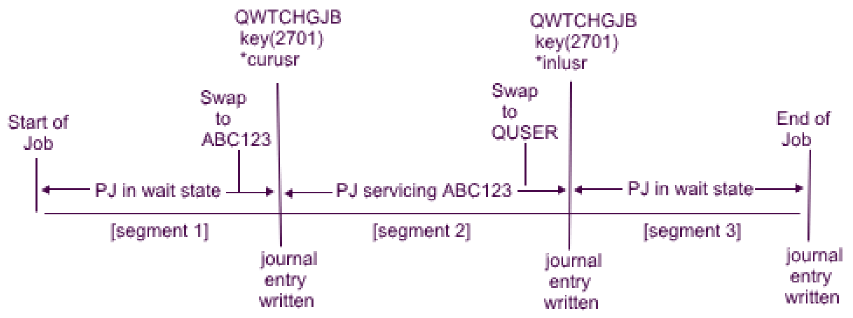

Figure 3. Prestart job with 3 accounting segments

If we look back at the three journal entries written in the example in Figure 3, they might look something like this when using SQL or query to format:

```
Journal  Job       Job     Job      User      Accounting
Entry #  Name      User    Number   Profile   Code           CPU  Transactions
  1      QSVREX1   QUSER   123456   ABC123    QUSER           50          1
  2      QSVREX1   QUSER   123456   QUSER     ABC123        3729        120
  3      QSVREX1   QUSER   123456   QUSER     QUSER           73          2
```

The resources used, such as CPU and transactions, can be charged back to the accounting code, but not necessarily to the user listed under the User Profile field (JAUSPF). The user profile is the current user at the time the journal entry is written, but it is not necessarily the user profile that was active during the entire accounting segment. In this example, the user profile has been swapped once in each of the first two segments. Since the journal entry is written after the swap, the current user profile logged in the entry is not the user who used the resources during the prior accounting segment.

Likewise, the Job User cannot reliably be used to charge for resources used, because this is the user that the job started with, and as part of the qualified job name, it does not change, even when servicing a different user. The accounting code is the only reliable field that can be used for charging resource usage. The accounting code differs from the other user fields because the accounting code is saved with the job until it is changed. At the time of the change, the job's current accounting code is written to the journal entry first, and then the new accounting code is stored in the job.

Tables: JB, DP, and SP journal entries

Table 1. Fields Found in JB Journal Entry

Field Name	Description	Field Attributes
JAJOB	Job name	Character (10)
JAUSER	Job user	Character (10)
JANBR	Job number	Zoned (6,0)
JACDE	Accounting code	Character (15)
JACPU	Processing unit time used (in milliseconds)	Packed decimal (11,0)
JARTGS	Number of routing steps	Packed decimal (5,0)
JAEDTE	Date job entered system (mmddyy format)	Character (6)

Field Name	Description	Field Attributes
JAETIM	Time job entered system (hhmmss format)	Character (6)
JASDTE	Date job started (mmddyy format)	Character (6)
JASTIM	Time job started (hhmmss format)	Character (6)
JATRNT	Total transaction time (in seconds)	Packed decimal (11,0)
JATRNS	Number of transactions	Packed decimal (11,0)
JAAUX	Synchronous auxiliary I/O operations and database operations (including page faults for any reason)	Packed decimal (11,0)
JATYPE	Job type	Character (1)
JCCDE	Job completion code	Packed decimal (3,0)
JALINE	Number of print lines	Packed decimal (11,0)
JAPAGE	Number of printed pages	Packed decimal (11,0)
JAPRTF	Number of print files	Packed decimal (11,0)
JADBPT	Number of database write operations	Packed decimal (11,0)
JADBGT	Number of database read operations	Packed decimal (11,0)
JADBUP	Number of database update, delete, FEOD, release, commit, and rollback operations	Packed decimal (11,0)
JACMPT	Number of communications write operations	Packed decimal (11,0)
JACMGT	Number of communications read operations	Packed decimal (11,0)
JAACT	Time job was active (in milliseconds)	Packed decimal (11,0)
JASPN	Time job was suspended (in milliseconds)	Packed decimal (11,0)
JAEDTL	Timestamp job entered system (mmddyyyyhhmmss)	Character (14)
JAESTL	Timestamp job started (mmddyyyyhhmmss)	Character (14)
JAAIO	Asynchronous I/O for database and non-database operations.	Packed decimal (11,0)
JAXCPU	Expanded CPU time used	Packed decimal (29,0)
JAXSIO	Expanded synchronous auxiliary I/O operations	Packed decimal (29,0)
JAXAIO	Expanded asynchronous auxiliary I/O operations	Packed decimal (29,0)
JAXDBP	Expanded number of database puts	Packed decimal (29,0)
JAXDBG	Expanded number of database gets	Packed decimal (29,0)
JAXDBU	Expanded number of database updates and deletes	Packed decimal (29,0)

Table 2. Fields Found in the DP Journal Entry

Field Name	Description	Field Attributes
JAJOB	Job name	Character (10)

Field Name	Description	Field Attributes
JAUSER	Job user	Character (10)
JANBR	Job number	Zoned (6,0)
JACDE	Accounting code	Character (15)
JADFN	Device file name	Character (10)
JADFNL	Library in which device file is stored	Character (10)
JADEVN	Device name	Character (10)
JADEVT	Device type	Character (4)
JADEVM	Device model	Character (4)
JATPAG	Total number of print pages produced	Packed decimal (11,0)
JATLIN	Total number of print lines produced	Packed decimal (11,0)
JASPFN	Always blank	Character (10)
JASPNB	Always blank	Character (4)
JAOPTY	Always blank	Character (1)
JAFMTP	Always blank	Character (10)
JABYTE	Always zero	Packed decimal (15,0)
JAUSRD	User data	Character (10)
JALSPN	Always blank	Character (6)
JASPSY	Always blank	Character (8)
JASPDT	Always blank	Character (7)
JASPTM	Always blank	Character (6)
JADFASP	Always blank	Character (10)

Table 3. Fields Found in SP Journal Entry

Field Name	Description	Field Attributes
JAJOB	Job name	Character (10)
JAUSER	Job user	Character (10)
JANBR	Job number	Zoned (6,0)
JACDE	Accounting code	Character (15)
JADFN	Device file name	Character (10)
JADFNL	Library in which device file is stored	Character (10)
JADEVN	Device name	Character (10)
JADEVT	Device type	Character (4)
JADEVM	Device model	Character (4)
JATPAG	Total number of print pages produced	Packed decimal (11,0)
JATLIN	Total number of print lines produced	Packed decimal (11,0)
JASPFN	Spooled file name	Character (10)
JASPNB	Spooled file number (blank if too long)	Character (4)
JAOPTY	Output priority	Character (1)
JAFMTP	Form type	Character (10)
JABYTE	Total number of bytes sent to the printer	Packed decimal (15,0)

Field Name	Description	Field Attributes
JAUSRD	User Data	Character (10)
JALSPN	Spooled file number	Character (6)
JASPSY	Spooled file job system name	Character (8)
JASPDT	Spooled file create date (cyymmdd format)	Character (7)
JASPTM	Spooled file create time (hhmmss format)	Character (6)
JADFASP	ASP name for device file library	Character (10)

Example: CL Program to change accounting code in user profiles

This is an example of a program that retrieves a list of all the user profiles on a system, and then changes the accounting code in each user profile to match the user profile name. It is written in iSeries Command Language (CL). This program example does not check for damaged user profiles, system user profiles, etc. Additional exception handling or status checking may be necessary. The program also needs to be modified if there are more than 1000 user profiles on the system. The program could be customized to append up to 5 additional characters to the user profile name in the accounting code, if needed.

Note: Read the code example disclaimer for important legal information.

```
PGM
        DCL        VAR(&USRSPC) TYPE(*CHAR) LEN(20)          +
                       VALUE('PROFILESUSQTEMP     ')
        DCL        VAR(&USLEN) TYPE(*CHAR) LEN(4)            +
                     VALUE(X'00500000')  /* Size of user      +
                     space */
        DCL        VAR(&USAUT) TYPE(*CHAR) LEN(10)           +
                      VALUE('*EXCLUDE  ')  /* authority */
        DCL        VAR(&USRPL) TYPE(*CHAR) LEN(10)           +
                      VALUE('*YES      ')  /* replace         +
                      existing user space  */
        DCL        VAR(&NUMENT) TYPE(*CHAR) LEN(4)  /* Number +
                      of entries from list object */
        DCL        VAR(&NUMENTB) TYPE(*DEC) LEN(4 0)/* Number +
                       of user profiles          */
        DCL        VAR(&COUNT) TYPE(*DEC) LEN(4 0) VALUE(0)
        DCL        VAR(&ENTRYLEN) TYPE(*CHAR) LEN(4)
                      /* Entry length                          */
        DCL        VAR(&ENTRYLENB) TYPE(*DEC) LEN(5 0)
                      /* Entry length in decimal format       */
        DCL        VAR(&OFFSETUS) TYPE(*CHAR) LEN(4)
                      /* Offset to profile entries            */
        DCL        VAR(&OFFSETUSB) TYPE(*DEC) LEN(6 0)
                      /* Offset to profile entries            */
        DCL        VAR(&START) TYPE(*CHAR) LEN(4)            +
                     VALUE(X'00000001') /* Starting position  +
                     in user space that is to be retrieved */
        DCL        VAR(&SIZE) TYPE(*CHAR) LEN(4)             +
                     VALUE(X'0000008C') /* Number of bytes to +
                     be retrieved from user space         */
        DCL        VAR(&GENHDR) TYPE(*CHAR) LEN(300) /* Generic +
                       header information from the user space  */
        DCL        VAR(&RECVAR0100) TYPE(*CHAR) LEN(32000)
        DCL        VAR(&FORMAT) TYPE(*CHAR) LEN(8)   +
                     VALUE('OBJL0100')
        DCL        VAR(&OBJNAME) TYPE(*CHAR) LEN(20) +
                     VALUE('*ALL      QSYS      ')
        DCL        VAR(&UPNAME) TYPE(*CHAR) LEN(10)  +
                     VALUE('          ')
```

```
      DCL        VAR(&OBJTYPE) TYPE(*CHAR) LEN(10) +
                   VALUE('*USRPRF   ')
      DCL        VAR(&ERRCODE) TYPE(*CHAR) LEN(8) +
                   VALUE(X'0000000000000000')
/*******************************************************************/
/* Create a user space                                             */
/*******************************************************************/
      CALL PGM(QSYS/QUSCRTUS) PARM(&USRSPC ' ' +
      &USLEN ' ' &USAUT ' ' &USRPL &ERRCODE )
/*******************************************************************/
/*  Call the QUSLOBJ API to get a list of user profiles            */
/*******************************************************************/
       CALL       PGM(QSYS/QUSLOBJ) PARM( +
                   &USRSPC &FORMAT &OBJNAME   +
                   &OBJTYPE &ERRCODE )
       MONMSG     MSGID(CPF0000) +
                    EXEC(GOTO CMDLBL(ABORT))

/*******************************************************************/
/*   READ THE GENERIC HEADER FROM THE USER SPACE                   */
/*   to find the number of user profiles returned and the length of */
/*   each entry.                                                   */
/*******************************************************************/
      CALL PGM(QSYS/QUSRTVUS) PARM(&USRSPC &START      +
                       &SIZE &GENHDR &ERRCODE)
     /*     number of profile entries in list           */
      CHGVAR   VAR(&NUMENT)  VALUE(%SST(&GENHDR 133 4))
      CHGVAR   VAR(&NUMENTB)  VALUE(%BIN(&NUMENT))
     /*     length of each profile entry                */
      CHGVAR   VAR(&ENTRYLEN) VALUE(%SST(&GENHDR 137 4))
      CHGVAR   VAR(&ENTRYLENB) VALUE(%BIN(&ENTRYLEN))
     /*     offset to first profile entry               */
      CHGVAR   VAR(&OFFSETUS) VALUE(%SST(&GENHDR 125 4))
      CHGVAR   VAR(&OFFSETUSB) VALUE(%BIN(&OFFSETUS) + 1)
      CHGVAR   VAR(%BIN(&OFFSETUS)) VALUE(&OFFSETUSB)
     /*     size of user space used                     */
      CHGVAR   VAR(&USLEN) VALUE(%SST(&GENHDR 105 4))

 /*******************************************************************/
 /*   RETRIEVE NAMES DATA FROM THE USER SPACE                       */
 /*******************************************************************/
     CALL PGM(QSYS/QUSRTVUS) PARM(&USRSPC              +
           &START &USLEN &RECVAR0100 &ERRCODE)

NEXTENTRY:
      CHGVAR VAR(&UPNAME)                              +
             VALUE(%SST(&RECVAR0100 &OFFSETUSB 10))
     /*****************************************************/
     /*  Change the user profile's accounting code to the    */
     /*  user profile name.                                   */
     /*****************************************************/
      CHGUSRPRF USRPRF(&UPNAME) ACGCDE(&UPNAME)
        MONMSG     MSGID(CPF0000)

      CHGVAR VAR(&OFFSETUSB) VALUE(&OFFSETUSB + &ENTRYLENB)
      CHGVAR VAR(&COUNT) VALUE(&COUNT + 1)
      IF COND(&COUNT *LT &NUMENTB) THEN(GOTO CMDLBL(NEXTENTRY))


ABORT:
  /*******************************************************************/
  /*  Delete the user space                                          */
  /*******************************************************************/
     DLTUSRSPC USRSPC(QTEMP/PROFILESUS)
       MONMSG     MSGID(CPF0000)

ENDPGM
```

Disclaimer

Information is provided ″AS IS″ without warranty of any kind. Mention or reference to non-IBM products is for informational purposes only and does not constitute an endorsement of such products by IBM.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve the performance improvements stated here.

© Copyright IBM Corp. 2004

Printed in USA