# COMMONWEALTH OF PENNSYLVANIA

# DEPARTMENT’S OF HUMAN SERVICES, INSURANCE, AND AGING

# INFORMATION TECHNOLOGY STANDARD

|  |  |
| --- | --- |
| Name Of Standard: | Number: |
| Database Management System (DMS) and Relational (RDMS) Programming Standards | **STD-DMS011** |
| Domain: **Data** | Category: Enterprise Database Coding  |
|  |  |
| Date Issued: | Issued By Direction Of: |
| **9/03/2002** | C:\Users\wtorres\Desktop\CliffSignature (3).jpg |
| Date Revised: |  |
| **9/16/2016** | Clifton Van Scyoc, Dir of Division of Technical Engineering  |

**Abstract:**

To define the functional responsibilities of application developers who develop programs in the Data Management System DMS 2200 and Relational Database Management System RDMS 2200 on the UNISYS 2200 mainframe.

**General:**

The following Department of Human Services (DHS)/Bureau of Information Systems’ (BIS) standards are supplemental requirements and/or restrictions to programming standards used in systems development organizations. DHS/BIS Division of Enterprise Applications is responsible for DMS 2200 (DMS) and RDMS 2200 (RDMS) application development.

**Standard:**

**General Responsibilities**

The DMS 2200 developer is responsible for the efficient and timely operation of the program in the systems to which he is assigned.

The Sr. Application Developer or project leader is responsible for ensuring all implementation activity for new software is coordinated with the DBA. A Senior Application Developer must be assigned the responsibility of confirming that the production software is working in the production environment.

**DMS Program Restrictions**

Develop all non-IQU database programs that use DMS databases with the ASCII (@ACOB) or UCS (@UCOB) compiler. This does not restrict non-DMS subroutines from development under other languages. Subroutines used in DMS update programs not developed with the ASCII compiler, require DBA approval for implementation. Do not use I-QU PLUS-1 from the third party vendor (KMSystem, Inc.) for production jobs. I-QU PLUS-1 is not the replacement of ACOB or UCOB. The DBA may grant production of some ad hoc reports or quick fixes using I-QU PLUS-1 due to the critical need.

Restrictions in Program Structure

1. A program shall have one entry point and one exit point and shall not exceed a maximum program size that is 45K wds for the I BANK and 65K wds for the D Bank. Any program not conforming to this restriction must be broken into smaller programs before placing into production. The DBA may approve some exceptions to this rule.
2. All DMS batch update programs must contain individual program commit points. Achieve this by executing FREE or DEPART/IMPART commands periodically during program execution. Program restart should be capable at the last successful commit point.
3. Do not sort records while IMPARTed to any database. You can use the RELEASE verb if the database is DEPARTed prior to the end of the INPUT PROCEDURE. This will ensure that the program is not IMPARTed to the database at the time the SORT occurs. The SORT actually occurs between the INPUT and OUTPUT procedures.
4. Do not use the RUN-UNIT-ID clause. Not using the RUN-UNIT-ID clause prevents DBP and UREP problems.

TIP/DMS Considerations

1. Do field editing before IMPARTing, to avoid tying up a DMR run unit slot before needing it.
2. Check the program logic flow to confirm that CRT-OUTPUT will not be performed without first performing CRT-INPUT. Performing CRT-OUTPUT before performing CRT-INPUT will cause the program to SYSERR.
3. To ensure proper updating of a database, the DEPART should be done before the CRT-OUTPUT.
4. Do not use DISPLAY UPON PRINTER within the general flow of the program when the program is mapped for production. You can use DISPLAY UPON PRINTER during testing and debugging, as well as in error recovery and rollback paragraphs.
5. All TIP programs using DMS must send DMS error code XXXXXX NUM XXXXXX to the terminal when DMS fatal errors occur.
6. DISPLAY UPON PRINTER is permitted for all other error displays within the DMS default general error and rollback paragraphs for TIP programs.
7. Programmers should initialize all data items in the program's housekeeping section unless the data items contain value clauses and never change through the program. This ensures proper initialization of re-entrant programs.
8. Area accesses of Initial Load, Exclusive and Protected are not permitted. Use of these options is the responsibility of the DBA and is not permitted within applications programming divisions.
9. If you are running a BATCH program and want to abort the run, you must ask the operator to "E" the Run-ID off. The run should not be x'ed off, as this will cause the run to stay as a ghost run unit in the UDS buffer, and may cause the UDS to abort and/or may cause database corruption.

**DMS Coding Requirements**

The DMS coding standards supplement the programming standards established and enforced by the Application Development Team.

**Comments**

All DMS programs must contain:

1. Description History at the beginning to describe what function the program performs,
2. Modification History after the Description History that lists the date and a number associated with the revision. In addition, the revision number is added throughout the program where the revision has been made.
3. Comments throughout the program to explain logic that will facilitate the understanding of the code for new developers.

Subschema Section

1. INVOKE SUBSCHEMA (Subschema-Name) IN FILE (Schema File) OF SCHEMA (Schema-Name).
	1. This clause specifies the Schema/Subschema to be used by the run unit and is required. If UCOB, schema file must be enclosed in single quotes.
2. COPY RECORDS INTO (Section-Name)
3. COPYING DATA-NAMES INTO (Section-Name)
4. DMCA AND RUN-UNIT-STATISTICS ARE (Section-Name)
	1. This Section-Name should use WORKING unless the other optional section names are specifically necessary within the program's logic.
5. ROLLBACK IS Paragraph-Name
	1. This clause is required for all database programs.
6. SAVE DATA INCLUDES DEFERRED-UPDATES
	1. DHS permits only transaction update programs performing 30 or fewer updates to use the SAVE DATA INCLUDES DEFERRED-UPDATES clause.
	2. Programs that open database areas must first initialize certain area names and area keys before the DMR accesses records within those areas.

Procedure Division

1. IMPART
	1. This clause is required and must be executed before any other DML command. Open all non-DMS files before using DMS IMPART.
	2. The impart statement must be included within its own paragraph name and have its own access control value initialize before issuing the IMPART. This paragraph name must be meaningful containing the word IMPART and prefixed by four sequence numbers.
2. OPEN
	1. All programs must specify individual areas to be opened when in production. OPEN ALL is not permitted.
	2. Use of OPEN for Initial-Load, Exclusive, or Protected is not permitted without a documented reason and the approval of the DBA. The DBA controls its use.
	3. This clause is required for each area that will be accessed.
	4. All areas to be opened must be contained within a paragraph, as is logically possible, with a meaningful paragraph name containing OPEN. Prefix the paragraph name with a 4-digit sequence number. This allows for multiple open area paragraphs bearing the same paragraph name throughout the logic of the program. No other DML statements are permitted within these paragraphs except for the access control values for the areas to be opened.
3. CLOSE
	1. All areas that are opened must be closed before program termination. This can be accomplished by explicitly issuing a CLOSE command per each area OPENed or by issuing the command CLOSE ALL.
4. MODIFY/GET/FETCH/STORE/INSERT/DELETE/FIND
	1. The use of the optional record clause is required since omission can cause a modification of a record type that the programmer may or may not have intended.
	2. The FETCH-WITH-LOCK FEATURE is not permitted. This prevents concurrent run units from accessing records at the same time.
	3. The KEEP COMMAND is not permitted. This command prevents concurrent run units from altering the current record.

Error Handling/Problem Debugging

1. DMR ERROR
	1. All DMS programs are required to contain DMS error handling. General error and Rollback error paragraphs must be coded in the program.
	2. For Rollback errors, the ROLLBACK clause must be used to specify XXXX-ROLLBACK-ERROR-RTN as the rollback error paragraph.
	3. Handle DMS General errors in either of the following two ways:
		1. The first way is to use an ERROR RECOVERY clause to specify a general error paragraph. Any errors not handled by the DMS programmer with the ON ERROR clause are passed to this paragraph.
		2. The second way to handle General errors is not to include the ERROR RECOVERY clause, and after ALL DML commands, explicitly check the ERROR-NUM field in the DMCA for the status of the DML command. Then send any unexpected errors to the general error paragraph XXXX- GENERAL-ERROR-RTN coded in the program. We recommend this method for structured programming.
	4. The default error paragraph must not return to continue normal program logic but must drop into the rollback paragraph. The two error paragraphs provide two necessary functions that complement each other and are essential. The default general error paragraph handles that particular error. The rollback says that processing is over. It is the clean-up paragraph. Therefore, all error handling is in one place.
	5. The programmer must include his own display requirement of data areas, records, elements, table, which he feels will help him analyze the problem for a quick solution to the condition.
	6. Minimum error displays can be no less than those listed below from the DMCA:

1. ERROR-STATUS= 7. ERROR-AREA=

2. ERROR-NUM= 8. ERROR-RECORD=

3. COMMAND-SEQ-NUM= 9. ERROR-SET=

4. AREA-NAME= 10. CURRENT-AREA-KEY=

5. REA-KEY= 11. DATABASE-KEY=

6. RECORD-NAME=

The minimum error displays listed within the previous rule must exist within both general error and rollback paragraphs. Both paragraphs must contain the same minimum error displays.

ADMLP/Compilation

1. Do not use the "T" Option.
2. Since the ADMLP permits copying of Procs and subroutines, turn option C on to correctly copy in any user Procs.
3. All DMS programs must collect the correct CBEP$$DMS with the absolute, when mapping.
4. Remove all debugging aids when mapping for production.

**Programs Using Both DMS and RDMS Databases**

See RDMS Coding Requirements in this document.

Database Usage Standards

**Purpose**

To provide Database Usage Standards for application developers in applying the DMS/RDMS systems and programming standards. A working knowledge of DMS 2200 and/or RDMS 2200 is needed.

**References**

* Collection Standards for Mapping Programs
* System Development/User Interface Standards
* Database Environment Systems Design Standards
* DMS Programming Standards
* RDMS Programming Standards
* COBOL Programming Reference Manual
* UDS DMS 2200 COBOL DML Operations & Programming Ref.
* UDS RDMS 2200 SQL Programming Reference Manual
* UDS RDMS 2200 & IPF SQL Interface End Users Guide

**Applicable**

* All systems support personnel
* All application developers
* All database personnel

**General**

Follow the standard usage of DMS/RDMS set forth in this section at all times, unless granted a specific exception. Direct a request for exception, as well as any comments and suggestions, to the Section Head of the Database Administration Section.

Use the test databases provided for all testing purposes. Test and production databases are identical except for the Schema name, Subschema names, and area sizing (page usage), Qualifier and the EXEC/TIP files, which are attached to the areas.

**Database Overview**

There are many mainframe databases within the Department of Human Services. Acquire schematic diagrams and lists of schema areas/set/records and table layouts through the database design analyst within the Database Design Unit. Application programmers use these diagrams and Proc listings to understand the record formats, relationships, and access paths available to the processing logic. The next section briefly describes the S20MRGPRDM, the largest database developed within the Department of Human Services (DHS).

**S20MRGPRDM**

This database, the largest database developed within DHS, contains Client Information (CIS), Provider/Reference System (PRV/REF), and Income Eligibility (IEVS) data. It is a TIP/DMS structure. DHS does not use EXEC Files (non-TIP) in production database environments. The CIS database contains information on all recipient clients throughout the Commonwealth. DHS uses this information primarily for on-line processing, updating, and inquiries concerning the five major benefit programs (CASH, MEDICAL, FOOD STAMP, SOCIAL SECURITY INCOME, and SOCIAL SERVICES). Many other interfaces are supported for outside agencies.

The PRV database contains information on all medical providers concerning the eligibility to render service with the many medical provider types. DHS uses it primarily for on-line inquiry and to support on-line provider relation updating.

As indicated, many other application DB’s are currently supported within BIS: S03PCHPRDO, S04WISPRDO, S05CBPRDO, S08TOSPRDO, S22MPBPRDO, S24LIHPRDO, S25CCBPRDO, S26SPCPRDO, S27HSTPRDO, and so forth.

**Program Overview**

Application developers must consider several items before writing or modifying programs. This usage section provides information, concerning our use of DMS/DML and RDMS standards, in a step-by-step manner so the developers can code the required standards within the logical flow of the program.

All information for any database is strictly confidential and all data listings, reports, and so forth, must be shredded or destroyed. Obtain a copy of any needed Schema documentation from your Database Design Section analyst in the DBA Section.

A programmer who is required to make a change to an existing production DML program must modify it to access a test database before testing can begin. That is, if you are testing a program, it must invoke a test Schema and reference areas of a test database. Do no testing against production databases. Compile, map, and execute runstreams must assign a test Schema file.

When a program is fully tested and is ready for production, convert it from its test image, to a production mode. Do this by modifying the program compile runstreams to reference the production Schema Residence File and the Production DMR, then recompiling, mapping, and placing the program into production.

**Error Recovery Methods**

Any DMS/RDMS program presently using internal sorts is not a candidate for error/recovery because there is no effective manner for check-pointing sorts. Therefore, future design strategies should deal with isolating, where possible, internal COBOL sorts outside of DMS/RDMS Programs.

DMS/DML and RDMS programs are not permitted to execute sorting logic while IMPARTed with the database.

**No Error/Recovery**

Advantages

1. Does not require additional facilities.
2. Cost effective for programs of short duration and does not carry the possibility of requiring a lengthy restore of data files.

Disadvantages

1. Requires re-running step.
2. May require restoration of data areas.

Description

This method is efficient for short-running programs where the time gained by a restart (i.e. starting at the last logical break) over that of a reload (i.e. rerunning that program) is inconsequential. In addition to the analysis, programming, testing, and implementation time, physical hardware would have to be committed.

**Error-Recovery with DMS/RDMS Updating and Non-DMS/RDMS File Handling Using Database Quick-Before-Looks and Program Checkpoints**

Advantages

1. Job can be restarted without duplicating processing already done.
2. Eliminates need for database reload.

Disadvantages

1. Requires additional file space for quick look files and checkpoint files.
2. Checkpointing adds to processing time.

Description

Use this technique where the database is being updated (thereby requiring a restore of the database in the event of error) and non-DMS/RDMS files are also involved.

**Error/Recovery with DMS/RDMS Retrieval and Non-DMS/RDMS File Handling Using Program Checkpoints**

Advantages

1. Saves processing already accomplished.

Disadvantages

1. Requires facilities for checkpoint files.
2. Requires additional time to take checkpoints.

Description

Use this approach with programs of long duration that are accessing the database. Allocating quick look space is not necessary.

**DMS/RDMS Files Only Using Quick–Before–Looks**

Advantages

1. Avoids need to reload database.
2. Preserves previous processing.

Disadvantages

1. Requires quick look file space.

Description

Error recovery for DMS/RDMS files is accomplished through FREEs and quick-before-looks. Each FREE acts as a checkpoint with run unit rollback replacing the database recovery.

**Coding Requirements**

Subschema Section

**Add:** INVOKE SUBSCHEMA (Subschema-Name) IN FILE (EXEC-File-Name) OF SCHEMA (Schema-Name)

The invoke clause must invoke a test Schema.

**Add:** KEY FOR INVOKE (Access-Control-Lock)

Provide the Subschema access-control-lock key for any use of specified Subschema/Schema. This key must be authorized for use by individual programs.

**Add:** COPYING RECORDS INTO (Section-Name)

COPYING DATA-NAMES INTO (Section-Name)

DMCA AND RUN-UNIT-STATISTICS ARE (Section-Name)

The section name should use WORKING-STORAGE unless the other optional section names are specifically necessary within the program's logic.

**Add:** ERROR RECOVERY IS (Paragraph-Name)

ROLLBACK IS (Paragraph-Name)

**Procedure Division**

Selection of DMS Procs is based upon TIP or non-TIP programs.

1. OPEN statements must individually open only those areas used by the program (no open all permitted). Do not use exclusive and protected usage mode.
2. All DML verbs not defaulting to the general error paragraph must contain an **ON ERROR** clause and FIND/FETCH must contain an **AT END**.
3. An error paragraph written by the user may not contain a DEPART or STOP RUN statement. In order to exit a program due to an error, control must be passed to the default general error paragraph.

**Standard Database Program Compile and Mapping Usage**

This section provides the developer with the necessary understanding of information to compile a COBOL-DML program and to map it into an executable absolute. All compiles and maps must use the test schema absolute under the qualifier TSTDMS\* until the program is ready to be placed into production.

Test Compilation Runstream

Refer to "Collection Standards for Mapping Programs - Standard number 30.21.”

For DMS/RDMS ACOB compiles, do not use the “T” Option because this causes the change of the subschema's picture definitions.

Map Runstreams (Multi-Banking, Multi-thread DMR)

Refer to "Collection Standards for Mapping Programs - Standard number 30.21.”

Special Considerations:

@MAP,I ,FILENAME.ABSOLUTE

(Option “I” is mandatory, others are optional depending upon your desired results.)

Everything will initially be created in TPF$. The “S” option can replace the I Option to give the full listing required to use a post mortem dump (PMD) in locating the cause of an error at execution time.

IBANK,M USERIB,01000

This format is used only in bank-named collections to specify the beginning of the source language defining the named bank. It was initially based on main PSR.

DBANK,MC USERDB,0160000

Same as IBANK directive except the start address is above 01000. This address is mandatory and cannot be altered without authorization from the Database Management Division. A different address may cause problems during execution of program and have overlapped addresses of the DMR.

IN qqqDMS\*SCHEMA$xxx.S$WORK/(Subschema-Name)

This element is the relocatable produced by the Subschema translator. The Subschema name is specified in the program identification division.

IN .D$WORK(1st 6 char of program name)

IN .(1st 6 char of program name)

Tip Program Addition Requirements

Refer to Collection Standards for Mapping Programs - Standard Number 30.21.

MINGAP 65000

Establishes the minimum gap usage for TIP banked programs.

MINSIZE 65000

Establishes the minimum size for TIP banked programs.

Compile and Map Runstream Sample

Refer to Collection Standards for Mapping Programs - Standard Number 30.21, page six through nine.

Creation of a New Version of a Test Database

If a version of a database is required that does not exist on a backup tape, the user design analyst and the database design analyst will create a different version load of the test data. This version will then be available on request. Make a request for a new version of test data no more frequently than once a month.

Debugging Aids

Dumps

If a DML error occurs, the program flow goes to the general error paragraph. A brief listing on the type of DML error is printed, all areas are closed, and the run unit DEPARTs with rollback.

QLP

Query Language Processor (QLP) is a useful tool for dumping small amounts of a database. Production Control registers and schedules QLP programs. The DBA Division is responsible for QLP maintenance and Subschema generation. Use of this processor is restricted, though you may be able to use it via Database Management Division involvement and approval.

Restrictions on QLP Use

Use QLP for retrieval only. Update via a DMS AD HOC request, after the Database Group reviews the QLP runstream and Production Control schedules the QLP runstream through a "Special Production Request".

1. Do not use the report writer feature in QLP, since it uses a large amount of system resources, which dramatically tie up the system.
2. Avoid repetitive use of QLP for the same information. (Consider writing a COBOL program to retrieve the information.)
3. Avoid queries, which scan large amounts of the database. The DBA has the responsibility to regulate the uses of QLP and may reject QLP process to require the user to develop the processes as a COBOL program.
4. Update only one record per QLP command.
5. The WHERE conditional clause must be used when changing or deleting a record.

Utility Dumps

The DBA Section will provide processes to dump the contents of any test database or specific parts of a test database. The DBA maintains and provides a complete listing of each test database together with statistics. The utility dump programs can provide the programmer with, 1) the option of obtaining a dump of a smaller portion of the database that is more suited to the testers’ specific needs, or 2) the option of obtaining his own copy of the contents of a test database. Contact the DBA Section staff concerning execution of the Database Editor processor utility for helpful use of its printing features.

**DMS/DML or RDMS Program Production Release**

Normal Procedure

Approval to release a test program to production is given only after the following steps are successfully completed:

Completion of Thorough Testing

This step is the responsibility of the project systems analyst and requires the execution of test runs to ensure that, 1) the test program is capable of correctly processing all possible variations of input, and 2) that all phases of program logic have been executed.

Conversion to Production Mode

There is a two-phase approach, which consists of, 1) the phase to convert the test source program into the production mode, and, 2) the actual implementation of the production requested program into the production mainstream by DHS operations.

(See *Data Dictionary* section of this document.)

DBA Request Phase I

The finalized error-free test source program must be converted before the program can become part of the production environment. The responsible application division changes all references to test Schemas and test DMR within the program so that the production schema and areas can be accessed.

Applications Development personnel are responsible for recompiling and mapping of the converted production program and transfer of the absolute into production status.

The Sr. Application Developer and developer must ensure that, 1) testing has occurred and the source, compile, and mapping are error free, and 2) implementation is provided with the time and date for production status.

Implementation of the Requested Program Phase II

When the program and the accompanying runstreams, listing, etc. are considered to be accurate, the Applications Development personnel request that operations copy the program into production libraries. It is the applications area's responsibility to see that DHS, Division of Infrastructure and Management and Operations, Batch Services and Support Section is provided the information that they need in order to support the new application.

There are many divisional/production files, having various names and contents, outlined in the Element Transfer Request Instructions located at <http://mydhs/oa/bis/appdevinfo/developerforms/index.htm>. These files are the following:

Program source elements

Program absolute elements

Runstreams for executing program absolutes

COBOL procedure elements (copy library)

COBOL subroutine relocatable

All batch programs running against a production database are executed form the production runstream file using production absolute elements. The complete documentation on the transfer request of programs, elements, etc. can be found in the Element Transfer Request Instructions.

Changing Production Programs

The altering of production program source code is conceivable because of unanticipated changing program requirements. In order to accomplish this procedure, the authorized division personnel then transfers a copy of the approved test source program from his division's source production file into the authorized personnel's selected file. The same procedure for the altered test source program is to be followed.

RDMS Programming Standards

**Purpose**

Define the functional responsibilities of RDMS application developers assigned to develop programs, which operate within the RDMS environment within the Department of Human Services.

**References**

* Collection Standards for Mapping Programs
* Systems Development/User Interface Standards
* Database Environment System Design Standards
* Database Usage Standards
* DMS Programming Standards

**Applicable**

All RDMS Application Developers

**General**

The following RDMS Standards are presented to function as supplemental requirements and/or restrictions to the programming standards being used by the Application Team.

**General Responsibilities**

Each RDMS application developer is responsible for the efficient and timely operation of the programs they maintain. Specifically, they must:

All program testing is restricted to the test database and in accordance with the application developer’s test plan. Review all testing with Senior Application Developer. Ensure all new implementation activity is coordinated with DBA and follow up the implementation to assure it is operating correctly. This is done in cooperation with all application developers.

**Programming Guideline for Efficiency**

1. In order to maximize program execution efficiency the following guidelines are to be followed during program development.
2. All accesses to RDMS data is to be reviewed and verified via the RDMS "EXPLAIN" command. The output access path identified via this process should match the programmers intended access path.
3. Do not include primary or secondary index fields in an update of a row if these fields are not being altered.
4. Repeated use of a cursor is made more efficient by opening and closing of the cursor rather than repeatedly declaring and dropping the cursor.
5. Use common-storage when possible to eliminate multiple accesses to a table where the same row is being accessed repeatedly.
6. When determining whether to issue a singleton select versus using a cursor apply the following:
	1. 2 or less rows retrieved - singleton select
	2. 3 or more rows retrieved - declare a cursor

RDMS Program Restrictions

General

1. All non-IQU database programs which access an RDMS Database are developed on the Systems using either ASCII Cobol (@ACOB) compiler or UCS Cobol (@UCOB) compiler or MAPPER runs using MRI. This does not restrict non-RDMS subroutines from development under other languages. Subroutines used in RDMS update programs that are not developed with either of the above mentioned compilers require DBA approval for implementation. I-QU PLUS-1 from the third party vendor (KMSystem, Inc.) cannot be used for production jobs. It is not the replacement of ACOB or UCOB. Only some adhoc reports or quick fixes using I-QU PLUS-1 may be granted by DBA due to the critical need.
2. Use of GRANT, REVOKE, CREATE, ALTER and DROP COMMANDS are restricted to the Database Group exclusively.
3. The use of the UNLOAD command, the ORDER BY clause of the DECLARE CURSOR command and the DISTINCT clause of the SELECT command are restricted to batch programs only. Any and all use of these instructions must be reviewed and approved by DBA prior to placement into any program.
4. Non-programmatic RDMS database updates such as with IPF SQL or IQU are restricted to the development/test system. If such an effort is required to be done on production system, it must be coordinated with Database Management Division.

Program Structure

A program has one entry point and one exit point and must not exceed a maximum program size that is 45K wds for the I BANK and 65K wds for the D Bank. Any program not conforming to this restriction must be broken into smaller programs before placing into production.

All RDMS batch update programs must contain individual program commit points. This is achieved by executing COMMIT or END/BEGIN THREAD commands periodically during program execution. Program restart can then commence with the last successful commit point if necessary. Always specify the recovery option desired. (Refer to Standard 35.07.08 Section B1a for determining recovery options).

No sorting of records is permitted while a THREAD is open. Use of the RELEASE verb is permitted provided that the END THREAD command has been executed prior to the end of the INPUT PROCEDURE. This will ensure that the program does not have a THREAD opened to any database at the time the SORT occurs. The SORT actually occurs between the INPUT and OUTPUT procedures.

TIP/RDMS Considerations

1. Field editing must be done before executing the BEGIN THREAD command to avoid tying up a DMR run unit slot prior to needing it.
2. Program logic flow must be checked to be sure that CRT-OUTPUT cannot be performed without first performing CRT-INPUT. Otherwise, the program will SYSERR.
3. To ensure proper updating of a database takes place, the COMMIT or END THREAD should be done prior to the CRT-OUTPUT.
4. Usage of display upon printer is not permitted within the general flow of the program when it is mapped for production. Display upon printer is permitted during testing and debugging. It is also allowed in the error recovery and rollback paragraphs.
5. All TIP programs using RDMS must send RDMS error status XXXX and RDMS aux-info XXXXXXXXXX to the terminal when RDMS fatal errors occur.
6. DISPLAY UPON PRINTER is permitted for all other error displays within the RDMS default general error and rollback paragraphs for TIP programs.
7. Application developers are to initialize all data items in the program's housekeeping section unless the data items contain value clauses and never change through the program. This ensures re-entrant programs are initialized properly.
8. Explicit locking of the RDMS tables in EXCLUSIVE, PROTECTED or SHARED mode is not permitted. Use of these modes require DBA approval for implementation.
9. RDMUTL processor is used to "LOAD" the tables in place of inserting a large number of rows. Use of the RDMUTL processor must be coordinated with the DBA group.
10. If you are executing a BATCH program and want to abort the run, you must ask the operator to "E" the Run-ID off. If the run is x'ed off if it will stay as a ghost run unit in the UDS buffer. It also may cause UDS to abort.

RDMS Coding Requirements

The RDMS standards are provided to supplement the programming standards established and enforced by the Bureau of Application Development and Delivery.

**Comments**

All RDMS programs must contain:

1. Description History at the beginning to describe what function the program performs,
2. Modification History after the Description History that lists the date and a number associated with the revision. In addition, the revision number is added throughout the program where the revision has been made.
3. Comments throughout the program to explain logic that will facilitate the understanding of the code for new developers.

Working-Storage Section

The following variables must be declared in any program that executes RDMS/RDML commands:

For ASCII COBOL:

01 RDMS-ERROR-STATUS PIC 9(04) .

01 RDMS-AUX-INFO PIC S1(36) .

For UCS COBOL:

77 SQLCODE PIC S9(09) USAGE BINARY.

01 RDMCA.

05 ERROR-STATUS PIC 9(04).

05 AUX-INFO PIC S9(09) USAGE BINARY.

01 RDMCA-RED REDEFINES RDMCA.

05 RDMS-ERROR-STATUS PIC 9(04).

05 RDMS-AUX-INFO PIC S9(09) USAGE BINARY.

For UCS COBOL ONLY:

Use the "BEGIN DECLARE" command to mark the beginning of the SQL variables portion of the WORKING STORAGE SECTION that you intend to use in embedded SQL statements.

Use the "END DECLARE" command to mark the end of the SQL variable portion of the WORKING STORAGE SECTION.

For ASCII COBOL ONLY:

The RDML-COMMAND is the first parameter to be passed to RSA, followed by the above mentioned two variables, followed by any other variables to be passed to RSA e.g.

ENTER MASM `ACOB$RDMR' using RDML-COMMAND,

RDMS-ERROR-STATUS,

RDMS-AUX-INFO,

--------,

--------,

--------,

etc.

A copy Proc that reflects the table structure should be included for each table accessed.

For development of table - definition copy Procs see the Section 35.07.10.

Procedure Division

BEGIN THREAD: This statement is required before execution of any other database command.

1. The begin thread statement must be included within its own paragraph name and should not include the optional UDSMSG clause. The paragraph name must be meaningful, contain the words begin-thread and be prefixed by a four digit sequence number.
2. Retrieval only programs must specify the recovery option as "read(none)".
3. Transaction update programs performing 30 or fewer updates should specify the recovery option as "update(deferred)".
4. Transaction update programs performing more than 30 updates should specify the recovery option as "update(quicklooks)".
5. Batch update programs must specify the recovery option as "update(quicklooks)".

Use Defaults

1. All programs must establish default qualifier (schema) name via the USE DEFAULT QUALIFIER command. This must be done immediately after doing begin thread.
2. As far as possible, programs must try to establish default versions also via the USE DEFAULT VERSION command.

For UCS COBOL Only

1. Retrieval-only programs should use the "WHENEVER SQLERROR" command for directing the program flow to handle fatal RDMS errors.
2. Update programs may also use this command at the discretion of the programmer/analyst.
3. All IF statements must have a corresponding END-IF.
4. Use INITIALIZE statement for initializing program variables.
5. Use the EVALUATE statement wherever possible instead of nesting three or more levels of IF statements.
6. Use in-line "PERFORM....END-PERFORM" statement for executing a small loop.

Cursors

1. Random cursors are not to be used.
2. A maximum of five cursors are to be active per program at any given time. An active cursor should be dropped as soon as it is no longer needed.

Error Handling/Problem Debugging

RDMS Error

1. All RDMS programs are required to contain RDMS error handling. General error and Rollback error paragraphs must be coded in the program.
2. After each RDML command explicitly check for the RDMS-ERROR-STATUS field for an error condition. Any unexpected errors should transfer control to the RDMS error paragraph XXXX-RDMS-ERROR-RTN coded in the program.
3. The XXX-RDMS-ERROR-RTN should display relevant RDMS status information along with any debugging aids, obtain the error message information from RDMS 'GETERROR' routine, rollback and bring the program to an orderly termination.
4. FOR UCS COBOL: If the program is using embedded SQL, it may use the following to transfer control to the RDMS error paragraph:

EXEC SQL

WHENEVER SQLERROR GOTO XXXX-RDMS-ERROR-RTN

END-EXEC

1. ***The default error paragraph must not return to continue normal program logic but must retrieve the error-message and execute the rollback paragraph.***

A sample RDMS-ERROR-RTN is presented in Section 35.07.09.

1. Minimum error displays can be no less than those listed below:

a. RDMS-ERROR-STATUS f. ERROR-MSG-LINE (1)

b. RDMS-AUX-INFO g. ERROR-MSG-LINE (2)

c. QUALIFIER/SCHEMA h. ERROR-MSG-LINE (3)

d. VERSION i. ERROR-MSG-LINE (4)

e. TABLE j. ERROR-MSG-LINE (5)

Compilation/Mapping Considerations for ASCII Cobol Program

All RDMS program must include the following two relocatable while mapping the absolutes

1. QQQQQQ\*RSA.CBEP$$RSA
2. QQQQQQ\*RSA.RDMR-ACOBDAT

Where QQQQQQ represents the following qualifiers:

DMR$$1 (Application Group 1 Production)

TDMR$$7 (Application Group 7 Test)

DDMR$$8 (Application Group 8 Development)

All debugging aids must be removed when a program is being mapped for production.

**Program Using Both DMS and RDMS Databases**

1. BEGIN THREAD command should be executed before IMPART and the END THREAD command should be executed after DEPART.
2. COMMIT command would cause all the changes made to DMS as well as RDMS databases to be permanent.
3. RDMS rollback would cause both the databases to be rolled back to the last commit point and would internally do a "DEPART WITH ROLLBACK". If the DMS database is being accessed after a RDMS rollback, an IMPART must be performed before issuing any other DMS command.
4. DEPART WITH ROLLBACK would rollback both the databases to the last commit point.
5. FREE or DEPART command does not commit your database changes when both DMS and RDMS databases are open in the same program.
6. In a program that uses both DMS as well as RDMS database, it is recommended that make your updates permanent using the COMMIT command and rollback the databases using the ROLLBACK command of RDMS.
7. A sample of error-handling routine is presented in section 35.07.10.

**RDMS Table Definitions and COBOL COPY PROCS**

Whenever a new RDMS table is created, DBA will generate the table description in a Cobol Copy Proc format using UREP processor. The copy Proc would be placed as an element on Host 5 with the following naming convention:

DPWDMS\*UREP$PROC.Tnnn

where Tnnn is Table Number.

1. Use of the UREP processor to generate the RDMS-table definition Cobol Copy Proc will be limited to the DBA group.
2. The DBA group will generate these copy Procs using UREP once (only when the table is initially established).

It would be the responsibility of the BADD staff to initially copy and modify/edit the UREP generated copy Procs and perform maintenance on these copy Procs.

It would be the responsibility of the BADD staff to maintain the table definition COBOL Copy Procs consistent with the actual RDMS table definition.

The following is a list of conventions that UREP follows while generating the table definition copy Procs:

For SMALLINT columns it generates PIC S9(5) USAGE BINARY fields.

For INTEGER columns it generates PIC S9(10) USAGE BINARY fields.

For character columns it generates PIC X(...) fields with the size of the field being same as the size of the column.

For DECIMAL & NUMERIC columns UREP generates PIC S9(..)V9(..) fields with SIGN LEADING SEPARATE clause. The total number of numeric positions in the COBOL field are one less than the total numeric positions specified in the column definition. Consider a column TOO1-DATE that is defined as NUMERIC (8,0) in the RDMS database. The UREP generated copy Proc would have the following Cobol definition:

01 TOO1-DATE PIC S9(7) SIGN LEADING SEPARATE.

A composite field column may be redefined in the copy Proc and broken down into individual fields that best serve the application. For example, consider a column T001-DATE that stores an 8-digit date. In the Cobol copy Proc it may be described as:

05 T001-DATE PIC 9(8).

05 T001-DATE-RED REDEFINES TOO1-DATE.

07 T001-DATE-YYYY PIC 9(4).

07 T001-DATE-MM PIC 9(2).

07 T001-DATE-DD PIC 9(2).

BADD staff must use the "RDMSCAT" utility developed by DBA to report the RDMS table definition and verify the consistency of the Cobol Copy Proc against the database.

To use the "RDMSCAT" utility, enter the following from your DEMAND terminal on host IKE-C:

@RDMSCAT (only applicable to the Application Group 8/ Development Environment)

It would then ask for schema name/table name about which you want the description.

Sample Error-Handling Routine

\*\*\*\*\*\*\*\*\*\*\*\*\*RDMS DATABASE ERROR ROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*

EXEC SQL

 WHENEVER SQLERROR CONTINUE

END-EXEC

DISPLAY '\*\*\*\* RDMS DATABASE ERROR \*\*\*\*' UPON PRINTER.

DISPLAY 'ERROR STATUS CODE = ' RDMS-ERROR-STATUS UPON PRINTER.

DISPLAY 'AUX-INFORMATION = ' RDMS-AUX-INFO UPON PRINTER.

DISPLAY 'QUALIFIER = ' RDMS-TBL-QUAL UPON PRINTER.

DISPLAY 'TABLE NAME = ' RDMS-TBL-NAME UPON PRINTER.

DISPLAY 'VERSION NAME = ' RDMS-VERSION-NAME UPON PRINTER.

EXEC SQL

 GETERROR INTO :RDMS-ERROR-LINE-1,

:RDMS-ERROR-LINE-2,

:RDMS-ERROR-LINE-3,

:RDMS-ERROR-LINE-4,

:RDMS-ERROR-LINE-5

END-EXEC.

DISPLAY RDMS-ERROR-LINE-1 UPON PRINTER.

DISPLAY RDMS-ERROR-LINE-2 UPON PRINTER.

DISPLAY RDMS-ERROR-LINE-3 UPON PRINTER.

DISPLAY RDMS-ERROR-LINE-4 UPON PRINTER.

DISPLAY RDMS-ERROR-LINE-5 UPON PRINTER.

DISPLAY SPACES UPON PRINTER.

DISPLAY RDMS-COMMAND-LN-1 UPON PRINTER.

DISPLAY RDMS-COMMAND-LN-2 UPON PRINTER.

DISPLAY RDMS-COMMAND-LN-3 UPON PRINTER.

 || IF IMPART-DEPART = '1'

 (\*\*) ¦¦ CLOSE ALL

 ¦¦ END-IF.

PERFORM 9999-RDMS-ROLLBACK

.

.

.

normal program termination.

.

.

.

¦¦ \*\*\*\*\*\*\*\*\*\*\*\*DMS DATABASE ERROR ROUTINES\*\*\*\*\*\*\*\*\*\*\*\*

¦¦

¦¦ DISPLAY '\*\*\*\* GENERAL DMS DATABASE ERROR \*\*\*\*'

¦¦ UPON PRINTER

¦¦

¦¦ PERFORM 9999-DISPLAY-ERROR.

¦¦

¦¦ IF IMPART-DEPART = '1'

(\*\*) ¦¦ CLOSE ALL

¦¦ END-IF.

¦¦

¦¦ PERFORM 9999-RDMS-ROLLBACK.

¦¦ .

¦¦ .

¦¦ .

¦¦ normal program termination

¦¦ .

¦¦ .

¦¦ .

9999-RDMS-ROLLBACK.

ROLLBACK.

IF RDMS-ERROR-STATUS NOT = ZEROES

DISPLAY '\*\*\* ROLLBACK UNSUCCESSFUL CONTACT DBA \*\*\*'

 UPON PRINTER

DISPLAY 'ERROR-STATUS = ' RDMS-ERROR-STATUS

 UPON PRINTER

 DISPLAY 'AUX-INFO = ' RDMS-AUX-INFO

 UPON PRINTER

 END THREAD

 . .

 . abort program

 .

 .

END-IF.

 END THREAD.

IF RDMS-ERROR-STATUS NOT = ZEROES

 DISPLAY '\*\*\* END THREAD UNSUCCESSFUL CONTACT DBA \*\*\*'

 UPON PRINTER

 DISPLAY 'ERROR-STATUS = ' RDMS-ERROR-STATUS

 UPON PRINTER

 DISPLAY 'AUX-INFO = ' RDMS-AUX-INFO

 UPON PRINTER

 .

 . abort program

 .

 .

END-IF.

 ¦¦ 9999-DISPLAY-ERROR.

 ¦¦ .

 ¦¦ .

 ¦¦ .

 (\*\*) ¦¦ Display DMS related error information as per DMS standards

 ¦¦ .

 ¦¦ .

 ¦¦ .

 (\*\*): applicable for programs that use both DMS and RDMS databases.

**NOTE:** For programs using ASCII Cobol, replace the Embedded SQL by interpretive interface calls.

**DMS/RDMS Program Debugging Requests for DBA Assistance**

**Purpose**

To describe the functional responsibilities of the DMS/RDMS applications programmer for obtaining Database Management Section assistance in diagnosing/debugging DMS/RDMS related programming problems.

**References**

* DMS/RDMS Programming Standards
* Database Usage Standards
* Collection Standards for Mapping Programs

**Applicable**

All DMS/RDMS Programmers

**General Responsibilities**

The DMS/RDMS programmer is responsible for assembling and submitting to the DBA Division the following materials as the minimum information to support any request for assistance in diagnosing a database related programming problem:

1. Description of the error encountered to include the host environment and a narrative of the events surrounding the error.
2. Current file/element names for the applicable source, compilation and absolute elements involved.
3. Hard-copy or on-line access to the compilation listing for the absolute encountering the error.
4. Screen print or TIPERS listing with the error identified for transaction programs.
5. Hard-copy or on-line access to the execution ECL for batch programs.
6. Hard-copy or on-line access to the breakpoint file for batch programs.
7. Hard-copy or on-line access to a DOWN listing comparing the previous working version of the program and the version encountering the error.
8. Completed "Program Debugging Request" form submitted to DBA (see next page).

**PROGRAM DEBUGGING REQUEST**

Date of Request \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Requestor\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Program ID\_ DMS\_\_\_ RDMS\_\_\_ TIP\_\_\_ BATCH\_\_\_\_\_ Host/Env\_\_\_\_\_

Schema/Subschema Invoked / Date/Time of Error

DMS Error Status/Number / RDMS Error Status

Program Source (QUAL\*FILE.ELEMENT)

Compile stream (QUAL\*FILE.ELEMENT)

Absolute element (QUAL\*FILE.ELEMENT)

Execution ECL (QUAL\*FILE.ELEMENT)

Breakpoint file (QUAL\*FILE.ELEMENT)

Screen Print/TIPERS \_\_\_ (attached)

Compilation Listing \_\_\_\_\_ (attached) or QUAL\*FILENAME.

DOWN Listing\_\_\_\_ \_\_\_\_\_ (attached) or QUAL\*FILENAME.

Description of events surrounding the error.

Can the error be recreated and how?

**Exemptions from this Standard:**

There will be no exemptions to this standard.

**Refresh Schedule:**

All standards and referenced documentation identified in this standard will be subject to review and possible revision annually or upon request by the DHS Information Technology Standards Team.

**Standard Revision Log:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Change Date** | **Version** | **Change Description** | **Author and Organization** |
|  9/03/2002 | 1.0 | Initial creation. | Database Management |
| 5/5/2016 | 1.0 | Update | William Torres |
| 9/08/2016 | 1.1 | Update | DMS and P. Gillingham |
| 9/16/2016 | 1.2 | Comments required in programs | P. Gillingham |