

Medicare Severity Diagnosis Related Groups (MS-DRG) Software

# **Software Installation Guide for z/OS Batch**

ICD-10

PBL-060

October 2025

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

If this product includes UB-04 information: Copyright 2025, American Hospital Association ("AHA"), Chicago, Illinois. Reproduced with permission. No portion of this publication may be reproduced, sorted in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without prior express, written consent of AHA.

# Table of Contents

<b>Preface .....</b>	<b>v</b>
Minimum system requirements for Java 17/64-bit.....	v
<b>Chapter 1: Introduction .....</b>	<b>7</b>
Versions of the software .....	7
Data format requirements .....	8
Information returned by the grouper .....	9
Grouper return code .....	11
Flags returned by the grouper .....	12
Ancillary buffer.....	16
<b>Chapter 2: Installing the MS-DRG Software .....</b>	<b>17</b>
Download instructions.....	18
Grouper program installation .....	18
JCL library.....	19
Load library.....	19
Object library .....	21
Source library.....	21
Java jar library .....	22
Interface methods .....	22
Miscellaneous files installation .....	23
Test Database File .....	23
ICD-10-CM/PCS MS-DRG Grouper Tables .....	24
Attributes.....	24
Procedure clusters .....	25
Grouper logic (drglogic).....	25
Diagnosis attributes (dxattlst) .....	28
Diagnosis table (dxlist) .....	29
Diagnosis pattern table (dxpatern) .....	29
Procedure attributes (prattlst).....	30
Procedure table (prlist).....	30
Procedure pattern table (prpatern).....	30
Procedure cluster definitions (prclstrs) .....	31
CC/MCC exclusions (exclusn) .....	32
Uploading grouper tables/files to the mainframe .....	33
DRG logic file .....	33
Diagnosis attribute list.....	33
Diagnosis list file .....	33
Diagnosis pattern list.....	34
CC/MCC exclusions file.....	34
Procedure attribute list.....	34

<i>Procedure list file .....</i>	<i>35</i>
<i>Procedure pattern list .....</i>	<i>35</i>
<i>Procedure clusters file .....</i>	<i>35</i>
<b>Chapter 3: Using the grouper with higher-level languages.....</b>	<b>37</b>
General strategy for COBOL driving program.....	37
Input to the grouper subroutines.....	37
Output from the grouper subroutines .....	38
Executor processing of the diagnosis and procedure buffers .....	38
<b>Chapter 4: Data processing.....</b>	<b>39</b>
Program input.....	39
Program output .....	40
<b>Chapter 5: The MS-DRG grouper executor .....</b>	<b>41</b>
Construction of the record mask.....	41
DRG determination.....	42
Testing for the ONLY surgery condition .....	42
Testing for the ONLY DX condition .....	43
Testing for the OWISE condition.....	43
Testing for the ANYCOMB condition.....	43
CC exclusion subroutine.....	43
Testing for the OTHOR condition .....	43
Testing for illogical principal diagnosis .....	44
Testing for multiple significant trauma.....	44
Finding codes that affect Initial DRG assignment .....	44
Final DRG.....	44
<b>Appendix A: Grouping results for the test database .....</b>	<b>45</b>

# Preface

This manual contains the information needed to use the Medicare Severity Diagnosis Related Groups (MS-DRG) Software (the grouper), version 43.0 in a mainframe environment. This software has the information to run in 64-bit using the same jar.

This manual provides technical personnel with the detail necessary to install, debug, and support the MS-DRG software. The first three chapters describe installing, testing, and running the grouper. Chapter 5 provides detailed information on the logic of the executor and the construction of the tables. The test database results are included in the INSTLCNT member in the JCL library.

Users already familiar with the MS-DRG software are encouraged to read this manual to ensure that installation, testing, and production runs perform without incident. If you have never used the software, we strongly recommend that you read the manual thoroughly to become familiar with it before installation.

The manual assumes that you are familiar with:

- IBM® MVS™ Job Control Language
- USS (UNIX System Services) on the mainframe

The current version of this software was developed and tested in the following environment:

- z/OS® version 2.5
- IBM® SDK for z/OS
- IBM Enterprise COBOL for z/OS 6.2.0
- Java Technology Edition version 17
- LE run-time environment

## Minimum system requirements for Java 17/64-bit

This section describes the minimum system requirements for Java 17/64-bit.

### **Operating System**

- The IBM® z/OS version 2.5.

### **LE run-time environment.64-bit**

- Internal Java-Calling program is now COBOL 6.4 and linked in AMODE = 64, RMODE = ANY. All other programs are COBOL 6.2 and linked in AMODE = 31, RMODE = ANY

### Minimum Requirement

- PUT Level: PUT2109 - HLE77D0, RSU2112 - HLE77D0
- For LE 6.4 Runtime, update to Java SE Runtime Environment version 17.0.11 (64-bit)

**Note:** Please review the sample JCL in the distribution for executing in 64-bit and Amode3164 statement in your JCL.

# Chapter 1: Introduction

This manual provides technical personnel with the detail necessary to install and understand the Medicare Severity Diagnosis Related Groups (MS-DRG) Software (the grouper) so they can install, interface with, and support it.

For example purposes, the required datasets are copied to disk and cataloged under the Userid, GROUPER.

**Note:** IBM Enterprise COBOL for z/OS will be upgraded from version 6.2 to version 6.4 for the April 2026 release and will remain backward compatible to version 6.2.

Effective with v39.1, grouping will be processed based on discharge date for claims with an effective date on 10/01/2021 (v39.0). See the following software version table for reference.

## Versions of the software

The following table lists the versions and date ranges of the MS-DRG software. As new versions are added, this table will be updated to indicate the versions of the program in the current release.

**Note:** This release completely replaces all prior Java versions of this software. You should completely replace the COBOL and JAR files from the previously installed version.

**Table 1. Versions of the software**

Version	Effective Date Range
v43.0	10/01/2025 - 12/31/9999*
v42.1	04/01/2025 - 09/30/2025
v42.0	10/01/2024 - 03/31/2025
v41.1	04/01/2024 - 09/30/2024
v41.0	10/01/2023 - 03/31/2024
v40.1	04/01/2023 - 09/30/2023
v40.0	10/01/2022 - 03/31/2023
v39.1	04/01/2022 - 09/30/2022
v39.0	10/01/2021 - 03/31/2022

\* The ending date of the current version will be modified to the actual ending date at the time of the next release.

## Data format requirements

The grouper executor is contained in a Java® JAR. The data formats required by the executor are shown in the table that follows.

The grouper must be implemented as a subroutine to a higher level language program that re-codes the information as necessary (page [37](#)).

**Table 2. Required data formats**

Name	Length in bytes	Description
Diagnosis	8	<p>First 7 bytes represent the diagnosis code. Left-justified, blank-filled, up to 25 accepted.</p> <p>The eighth byte represents the POA indicator.</p> <p>Y - Yes, present at the time of inpatient admission</p> <p>N - No, not present at the time of inpatient admission</p> <p>U - Insufficient documentation to determine if present on admission</p> <p>W - Clinically unable to determine if present at time of admission</p> <p>1 - Code is exempt from POA reporting (Used on 4010 form)</p> <p>Blank - Code is exempt from POA reporting (Used on 5010 form, effective 01/01/2011)</p>
Procedure	7	Left-justified, blank-filled, up to 25 accepted
Age	3	0 (zero) through 124, right-justified
Sex	1	0-2 (0-unknown, 1-male, 2-female)
Discharge Status	2	<p>01-Home, Self-Care</p> <p>02-Short Term Hosp</p> <p>03-SNF</p> <p>04-Custodial/supportive care (effective 10/1/2009)</p> <p>05-Canc/Child hosp</p> <p>06-Home Health Service</p> <p>07-Against Medical Advice</p> <p>20-Died</p> <p>21-Court/law enfrc</p> <p>30-Still A Patient</p> <p>43-FedHospital</p> <p>50-Hospice-Home</p> <p>51-Hospice-Medical Facility</p> <p>61-Swing Bed</p> <p>62-Rehab facility/rehab unit</p>



Name	Length in bytes	Description
(continued) Discharge Status		(continued) 63-Long term care hospital 64-Nursing facility - Medicaid certified 65-Psych hosp/unit 66-Critical Access Hospital 69-Designated Disaster Alternative Care Site 70-Oth institution 81-Home-Self care w Planned Readmission 82-Short Term Hospital w Planned Readmission 83-SNF w Planned Readmission 84-Cust/supp care w Planned Readmission 85-Canc/child hosp w Planned Readmission 86-Home Health Service w Planned Readmission 87-Court/law enfrc w Planned Readmission 88-Federal Hospital w Planned Readmission 89-Swing Bed w Planned Readmission 90-Rehab Facility/ Unit w Planned Readmission 91-LTCH w Planned Readmission 92-Nursg Fac-Medicaid Cert w Planned Readmission 93-Psych Hosp/Unit w Planned Readmission 94-Crit Acc Hosp w Planned Readmission 95-Oth Institution w Planned Readmission
POA logic	1	Present On Admission (POA) logic indicator. X - Exempt from POA reporting Z - Requires POA reporting
Admit Date	8	Format = YYYYMMDD
Discharge Date	8	Format = YYYYMMDD (used internally for selecting grouping version)
Procedure Dates	200	Date of each procedure code Format = YYYYMMDD (for use with future POA logic)

## Information returned by the grouper

The information returned by the grouper is shown in the following tables.

The field DRG listed below represents the 3-digit MS-DRG number used by the Centers for Medicare & Medicaid Services (CMS) for DRG payment purposes. The 3-byte “initial DRG” field in the ancillary buffer represents the DRG prior to the application of the HAC logic. The ancillary buffer also contains 4-byte initial and final DRG numbers. These 4-byte DRG numbers are for

statistical purposes only. Each 3-digit DRG concept is split on MCC, CC, and non-CC to create the 4-digit DRG.

For example, as a 3-digit DRG, Non-specific CVA & precerebral occlusion w/o infarction is split into 067 (w MCC) and 068 (w/o MCC). As a 4-digit DRG, Non-specific CVA & precerebral occlusion w/o infarction is split into 0671 (w MCC), 0672 (w CC), and 0673 (w/o CC/MCC). There are also “initial” and “final” flags in the diagnosis flag buffer. The flags indicating which secondary diagnoses and procedures affect DRG assignment apply to the 3-digit DRG, but could be different for the 4-digit DRG.

The grouper executor may be implemented as a subroutine to be called from a higher-level language program. This chapter shows how this may be done for a COBOL programming environment. To create the subroutines, you must have copied file 1 (the grouper objlib) to the mainframe (page [20](#)).

**Table 3. Information returned by the MS-DRG software**

Name	Length in bytes	Description
RTC	2	<p>Grouper return code (page <a href="#">11</a>)</p> <p>00 - Record grouped</p> <p>01 - Diagnosis code cannot be used as principal dx</p> <p>02 - Record does not meet criteria for any DRG in the MDC that is indicated by principal dx</p> <p>03 - Invalid age</p> <p>04 - Invalid sex</p> <p>05 - Invalid discharge status</p> <p>06 - Illogical principal diagnosis</p> <p>07 - Invalid principal diagnosis</p> <p>09 - POA logic indicator = Z and at least one HAC POA is invalid, missing, or 1</p> <p>10 - POA logic indicator is invalid or missing and at least one HAC POA is N or U</p> <p>11 - POA logic indicator is missing or invalid, and at least one HAC POA is invalid, missing, or 1</p> <p>12 - (not valid effective 10/1/2010) POA logic indicator = Z and at least one HAC POA =1</p> <p>13 - (not valid effective 10/1/2010) POA logic indicator is invalid or missing and at least one HAC POA = 1</p> <p>14 - (not valid effective 10/1/2010) POA logic indicator = Z and there are multiple HACs that have different HAC POA values that are not Y, W, N, U</p>

Name	Length in bytes	Description
(continued) RTC		(continued) 15 - POA logic indicator is missing or invalid, and there are multiple HACs that have different HAC POA values that are not Y or W 97 - JVM FAILURE OCCURRED - THIS CLAIM REJECTED. CHECK SYSOUT FOR JVM STACK 98 - Date entered is prior to 10/01/2021 99 - Date entered is invalid or blank
Final MDC	2	Major Diagnostic Category number (00 - 25) assigned to patient record
Final DRG	4	Diagnosis Related Group number (0001 - 0999) assigned to patient record (after HAC logic is applied)
GRFLGS	5	See table for "Grouper flags returned by the MS-DRG software" (page <a href="#">12</a> )
DXFLGS	625	See table for "Diagnosis flags returned by the grouper" (page <a href="#">13</a> )
PRFLGS	500	See table for "Procedure flags returned by the grouper" (page <a href="#">14</a> )

## Grouper return code

The grouper return code (RTC) indicates whether or not the grouping process was successful for a given record. The following table describes the values for the Return Code.

**Table 4. Return code descriptions**

Return code	Description
1	The first listed diagnosis is a valid code but it can not be used as principal diagnosis. An example of this situation would be any one of the ICD-10-CM "V" codes, which are not indicative of the MDC into which this patient should be classified.
2	This code occurs when all of the DRG criteria for the MDC have been examined and the record does not match any of them.
3,4 and 5	These codes occur only for those DRGs that are part of grouping criteria (i.e., the grouper does not perform an automatic edit check of age, sex, and discharge status).
6	The principal diagnosis is considered illogical, meaning that it is unlikely that there would be an occurrence.

Return code	Description
7	The code used as principal diagnosis is not a valid ICD-10-CM code.
9, 10, 11, 15	These codes occur when there is at least one HAC on the record and there is an issue with either the POA logic indicator or the POA values assigned to the HAC.
97	JVM FAILURE OCCURRED - THIS CLAIM REJECTED. CHECK SYSOUT FOR JVM STACK
98	Date entered is prior to 10/01/2021.
99	Date entered is invalid or blank.

## Flags returned by the grouper

The following tables show the information returned by the grouper regarding DRGs, diagnoses, and procedures.

Some secondary diagnosis codes can be assigned to multiple Hospital Acquired Conditions (HACs). In the diagnosis flags table, the fields for Hospital Acquired Condition (HAC) assignment criteria and Hospital Acquired Condition (HAC) Usage have been expanded to five occurrences. This allows for the capture of all HACs met by a secondary diagnosis. In the procedure flags table, the Hospital Acquired Condition (HAC) assignment criteria field has been expanded to five occurrences to reflect all HACs the procedure was used in, along with the secondary diagnosis that satisfied the HAC criteria.

**Table 5. Grouper flags returned by the MS-DRG software**

Position	Description
1 and 2	Number of unique Hospital Acquired Conditions (HAC) met
3	Final CC/MCC impact on DRG assignment 0 = DRG assigned is not based on the presence of a CC or MCC 1 = DRG assigned is based on presence of MCC 2 = DRG assigned is based on presence of CC
4	Initial CC/MCC impact on DRG assignment 0 = DRG assigned is not based on the presence of a CC or MCC 1 = DRG assigned is based on presence of MCC 2 = DRG assigned is based on presence of CC
5	HAC Status 0 = HAC Not Applicable; Hospital is exempt or HAC criteria not met 1 = Criteria for one or more HACs met, Final DRG did not change 2 = Criteria for one or more HACs met, Final DRG changed 3 = Criteria for one or more HACs met, Final DRG changed to 999

**Table 6.           Diagnosis flags returned by the grouper**

<b>Position</b>	<b>Description (25 characters per diagnosis)</b>
1	0 = Diagnosis invalid 1 = Diagnosis valid
2	Diagnosis affects DRG 0 = Diagnosis not used to assign DRG 1 = Diagnosis affected the initial DRG only 2 = Diagnosis affected the final DRG only 3 = Diagnosis affected both initial and final DRG
3	CC/MCC Categorization 0 = Diagnosis is not considered a Major CC or CC for this patient 1 = Diagnosis is a Major CC for both initial and final DRG 2 = Diagnosis is a CC for both initial and final DRG 3 = Diagnosis is a MCC for initial DRG and a Non-CC for final DRG 4 = Diagnosis is a CC for initial DRG and a Non-CC for final DRG 5 = Diagnosis is a MCC but not considered due to PDX/SDX exclusion 6 = Diagnosis is a CC but not considered due to PDX/SDX exclusion 7 = Diagnosis is a MCC but not considered due to DRG logic exclusion 8 = Diagnosis is a CC but not considered due to DRG logic exclusion
4 and 5	Hospital Acquired Condition (HAC) assignment criteria #1 00 = Criteria to be assigned as an HAC not met 01 = Foreign Object Retained After Surgery 02 = Air Embolism 03 = Blood Incompatibility 04 = Stage III and IV pressure ulcers 05 = Falls and Trauma 06 = Catheter Associated UTI 07 = Vascular Catheter-Associated Infection 08 = Infection following CABG 09 = Manifestations of poor glycemic control 10 = DVT/PE following knee or hip replacement 11 = Infection following bariatric surgery 12 = Infection following certain orthopedic procedures of spine, shoulder or elbow 13 = Surgical site infection (SSI) following cardiac implantable electronic device (CIED) procedures 14 = Iatrogenic pneumothorax w/ venous catheterization

Position	Description (25 characters per diagnosis)
6	Hospital Acquired Condition (HAC) Usage #1 0 = Dx not on HAC list, not applicable 1 = Dx on HAC list and HAC criteria met 2 = Dx on HAC list and HAC criteria not met 3 = Dx on HAC list, but HAC not applicable due to PDX/SDX exclusion 4 = HAC not applicable, hospital is exempt from POA reporting
7 and 8	Hospital Acquired Condition (HAC) assignment criteria #2 Refer to positions 4 and 5 for a list of values.
9	Hospital Acquired Condition (HAC) Usage #2 Refer to position 6 for a list of values.
10 and 11	Hospital Acquired Condition (HAC) assignment criteria #3 Refer to positions 4 and 5 for a list of values.
12	Hospital Acquired Condition (HAC) Usage #3 Refer to position 6 for a list of values.
13 and 14	Hospital Acquired Condition (HAC) assignment criteria #4 Refer to positions 4 and 5 for a list of values.
15	Hospital Acquired Condition (HAC) Usage #4 Refer to position 6 for a list of values.
16 and 17	Hospital Acquired Condition (HAC) assignment criteria #5 Refer to positions 4 and 5 for a list of values.
18	Hospital Acquired Condition (HAC) Usage #5 Refer to position 6 for a list of values.
19-25	Filler

**Table 7. Procedure flags returned by the grouper**

Position	Description (20 characters per procedure)
1	0 = Procedure invalid 1 = Procedure valid
2	<p>Procedure affects DRG*</p> <p>0 = Procedure did not affect DRG assignment 1 = Procedure affected the initial DRG assignment only 2 = Procedure affected the final DRG assignment only 3 = Procedure affected both initial and final DRG assignment</p> <p>* When there are two or more procedures on the record that could impact either the initial, final or both DRG assignments: If one of these procedures is in the first procedure position, that procedure will be be flagged as 1, 2 or 3 with the following exceptions:</p> <ul style="list-style-type: none"> <li>a. If a single procedure designating a complete system is tied with a combination pair that also designated a complete system, the single procedure will be flagged regardless of position.</li> <li>b. If multiple combinations of lead/device pairs are tied then only one pair will be flagged regardless of position.</li> <li>c. If the two procedures tied are an OR and non-OR, the OR will be flagged regardless of position.</li> </ul> <p>If none of the tied procedures is in the first procedure position, then the procedure with the lowest ascii/index value will be flagged as 1, 2 or 3.</p>
3	0 = Procedure is not an OR procedure 1 = Procedure is an OR procedure
4 and 5	<p>Hospital Acquired Condition (HAC) assignment criteria #1</p> <p>00 = Criteria to be assigned as an HAC not met 01 = Used as part of HAC 08 - Infection following CABG 02 = Used as part of HAC 10 - DVT/PE following knee or hip replacement 03 = Used as part of HAC 11 - Infection following bariatric surgery 04 = Used as part of HAC 12 - Infection following certain orthopedic procedures of spine, shoulder or elbow 05 = Used as part of HAC 13 - Surgical site infection (SSI) following cardiac implantable electronic device (CIED) procedures 06 = Used as part of HAC 14 - Iatrogenic pneumothorax w/ venous catheterization</p>
6 and 7	<p>Hospital Acquired Condition (HAC) assignment criteria #2</p> <p>Refer to positions 4 and 5 for a list of values.</p>
8 and 9	<p>Hospital Acquired Condition (HAC) assignment criteria #3</p> <p>Refer to positions 4 and 5 for a list of values.</p>

Position	Description (20 characters per procedure)
10 and 11	Hospital Acquired Condition (HAC) assignment criteria #4 Refer to positions 4 and 5 for a list of values.
12 and 13	Hospital Acquired Condition (HAC) assignment criteria #5 Refer to positions 4 and 5 for a list of values.
14-20	Filler

## Ancillary buffer

The version number identifies the version of the grouper that is running.

**Table 8. Additional flag information**

Length in bytes	Description
5	1 byte reserved space (zero-filled) followed by 4-byte final Base DRG (after HAC logic applied)
1	Final DRG Medical/Surgical Indicator 0 = Error DRG (998 or 999) 1 = Medical DRG 2 = Surgical DRG
4	1 byte reserved space (zero-filled) followed by 3-byte initial DRG (prior to HAC logic)
5	1 byte reserved space (zero-filled) followed by 4-byte initial Base DRG (prior to HAC logic)
1	Initial DRG Medical/Surgical indicator 0 = Error DRG (998 or 999) 1 = Medical DRG 2 = Surgical DRG
8	Version ID returned by the grouper (PPPVVVUU) <ul style="list-style-type: none"> <li>▪ PPP = 001 (MS-DRG)</li> <li>▪ VVV = 430 (Grouper version 43.0). The VVV will be returned based on the date and version used in table 1 (page <a href="#">7</a>).</li> <li>▪ UU = 00 (update 00)</li> </ul>



## Chapter 2: Installing the MS-DRG Software

This software can only be executed using the 64-bit version of Java version 17. Downloading and installing the Medicare Severity Diagnosis Related Groups (MS-DRG) Software (the grouper) consists of three steps:

1. Downloading and unzipping the file to your local machine
2. Allocating and FTPing the files to the mainframe
3. Testing the grouper

The first two steps are discussed in this chapter.

Chapter 4 shows how to link-edit the grouper programs for use as subroutines for a higher-level language. One COBOL program using the test database is included in the distribution, and the JCL for using it to test the installation is included in chapter 4.

**Note:** IBM Enterprise COBOL for z/OS will be upgraded from version 6.2 to version 6.4 for the April 2026 release and will remain backward compatible to version 6.2.

**Note:** This release completely replaces all prior Java versions of this software. You should completely replace the COBOL and JAR files from the previously installed version.

---

**Important!** Program names in tables 11-14 will be updated in the next release.

---

The content of the downloaded file folder is shown in the following table.

**Table 9. MS-DRG media contents**

File	File name	LRECL	BLKSIZE	Description
1	OBJLIB	80	27920	Object library
2	SRCLIB	80	32720	Source library
3	LOADLIB	0	6233	Load library
4	JAR	USS	USS	Binary Jar files and ASCII Control Files for the Unix System

The content of the miscellaneous folder is shown in the following table.

**Table 10. MS-DRG miscellaneous folder contents**

File	File name	LRECL	BLKSIZE	Description
1	TESTDB	2000	18000	Test database
2	MDCDSC	80	27920	MDC titles
3	DRGDSC3	200	27800	DRG titles (3-digit)
4	DRGDSC4	200	27800	DRG titles (4-digit)
5	DRGLOGIC	600	27600	DRG logic
6	DXATTLST	120	27960	Diagnosis attribute list
7	DXLIST	45	27990	Diagnosis list
8	DXPATTERN	60	27960	Diagnosis pattern
9	EXCLUSN	30	27990	Exclusion
10	PRATTLST	120	27960	Procedure attribute list
11	PRCLSTRS	65	27950	Procedure clusters
12	PRLIST	20	27980	Procedure list
13	PRPATTERN	160	27840	Procedure pattern
14	JCL	80	27920	Sample JCL. See table 11 (page <a href="#">19</a> ).

## eDownload instructions

This section contains instructions for downloading program files from the Internet or from a CD for the Medicare Severity Diagnosis Related Groups (MS-DRG) Software.

### Grouper program installation

All required software for executing the MS-DRG grouper is contained in the folders in this directory.

This directory contains the following folders:

- Load library - MS-DRG grouper load modules
- Object library - MS-DRG grouper object modules
- Source library - MS-DRG grouper source programs

- Test database file
- Sample JCL
- Java® jar
- Environment file

### JCL library

**Table 11. Sample JCL library members**

Number	Name	Description
1	BLDPDSE	Sample JCL used for electronic download
2	COBTST64	Run sample COBOL program (COBTEST) (Java 64-bit)
3	INSTLCNT	Readme for install test database record count
4	COBGO64	Run test database, executing COBTST64 Load library members (31-bit compiled COBOL calling Java 64-bit)

The following steps download the JCL library.

1. Allocate a PDSE on your mainframe with the following characteristics:
  - DSN = [e.g. YOURID.GROUPER.JCL]
  - RECFM = FB
  - LRECL = 80
  - BLKSIZE = 27920
  - SPACE = (TRK(2,1,4),RLSE)
2. FTP in ASCII mode all the files in the sample JCL library folder into the PDSE allocated in step 1.

### Load library

The load library is a sequential file, FTPLOAD. The load library consists of the load modules for the MS-DRG Grouper.

1. Pre-allocate a sequential dataset on your mainframe to receive the file using the following file characteristics:
  - DSN = [e.g. YOURID.GROUPER.FTPLOAD]
  - RECFM = FB
  - LRECL = 80
  - BLKSIZE = 3120

- SPACE = (CYL(1,1),RLSE)
2. FTP in BINARY mode the FTPLOAD file into the sequential dataset you allocated above.
 

**Important!** You must FTP the load module files in BINARY.
  3. Pre-allocate a load library PDSE on the mainframe using the following file characteristics:
    - DSN = [e.g. YOURID.GROUPER.LOADLIB]
    - RECFM = U
    - BLKSIZE = 6233
    - SPACE = (CYL(1,3,2),RLSE)
  4. Create a BLDPDSE JCL member as follows:
    - Add your JOBCARD
    - Modify dataset names as necessary
      - ♦ INDATASET = sequential dataset that was FTP'd to the mainframe in the step above.
      - ♦ DATASET = pre-allocated load library PDSE(Library) that was created in the step above.

**Note:** This JCL executes the utility, IKJEFT01, a terminal monitor program that executes the TSO commands via batch processing. This will populate the LOAD LIBRARY from the FTP'd load sequential file. A copy is included in the JCLLIB folder.
  5. After you modify the BLDPDSE, execute the JCL.

**Table 12. Load library contents**

Number	Name	Description
1	DR430R64	Sample program to execute files (31-bit compiled COBOL calling Java 64-bit)
2	DR430J64	Java wrapper Control program (64-bit compiled COBOL calling Java 64-bit)
3	COBTST64	Sample COBOL interface program (31-bit compiled COBOL calling Java 64-bit)

*Object library***Table 13. Object library contents**

Number	Program	Description
1	DR430R64	Sample program to execute files (31-bit compiled COBOL calling Java 64-bit)
2	DR430J64	Java wrapper Control program (64-bit compiled COBOL calling Java 64-bit)
3	COBTST64	Sample COBOL interface program (31-bit compiled COBOL calling Java 64-bit)

**Important!** Object module files must be FTP'd in BINARY.

The following steps download the object library.

1. Pre-allocate a load library PDSE on the mainframe using the following file characteristics:
  - DSN = [e.g. YOURID.GROUPER.OBJLIB]
  - RECFM = FB
  - LRECL = 80
  - BLKSIZE = 27920
  - SPACE = (CYL(1,1,2),RLSE)
2. FTP in **BINARY mode** all the files in the object library folder into the PDSE allocated in step 1.

*Source library*

There are several datasets included on the media that are not needed for the grouping process but may be useful to grouper users.

The folder contains the source library for all the grouper programs, tables, and the COBOL test programs. The library contains three members, as listed in the following table.

**Table 14. Source library contents**

Number	Name	Description
1	DR430R64	Sample program to execute files (31-bit compiled COBOL calling Java 64-bit)
2	COBTST64	Sample COBOL interface program (31-bit compiled COBOL calling Java 64-bit)

The following steps are required to FTP the source library to the mainframe.

1. Pre-allocate a load library PDSE on the mainframe using the following file characteristics:
  - DSN = [e.g. YOURID.GROUPER.SRCLIB]
  - RECFM = FB
  - LRECL = 80
  - BLKSIZE = 32720
  - SPACE = (CYL(1,1,4),RLSE)
2. FTP in ASCII mode all the files in the source library folder into the PDSE allocated in step 1.

### *Java jar library*

The MSDRGMF430 software is provided to users to allow them to group MS-DRG claims using the v43.0 software in a Java® environment and can be called from an outside program. This software uses the 64-bit version of Java.

The following files are needed to execute the MSDRGMF430 software:

- MSDRGMF430.jar – the jar file to be run by calls from an outside program.

Users wishing to embed this jar in their own applications may do so by invoking the following method:

```
String results = grouper.process(inputRecord)
```

The method takes a String in the specified input format (page [39](#)) and returns the grouping results as a String in the specified output format (page [40](#)).

1. On the USS system, create a folder 'dist'.
2. FTP the modules in the java-jar folders in Binary mode into that folder.
3. One member is included: MSDRGMF430.jar file.
4. FTP the DRGENV43064 in ASCII mode and place the file at the same level as the dist folder.
5. Update the above files with the correct path in your environment.
6. Min and max default heap size updated to 256m in the environment files.

### *Interface methods*

An example program that calls the grouper might look something like this:

```
public void myProgram() {  
  
    String inputRecord = null;  
  
    Msdrgr grouper = new Msdrgr();  
}
```

```
// populate the inputRecord string

String results = grouper.processMsdrg(inputRecord);

// review results

}
```

## Miscellaneous files installation

### Test Database File

The following steps load the test database file to the mainframe.

1. Allocate a sequential file (PS) on your mainframe using the attributes below.
  - DSN=YOURID.GROUPER.TESTDB
  - RECFM=FB
  - LRECL=2000
  - BLKSIZE=18000
  - SPACE=(CYL,(20,1),RLSE)
2. FTP the TESTDB file in ASCII mode from the miscellaneous folder to the mainframe, YOURID.GROUPER.TESTDB.

The following table provides a record description for the test database.

**Table 15. Test database layout**

Field	Location	Name	Description
1	1-3	AGE	Age on admission, in years
2	4-4	SEX	Gender
3	5-6	Discharge Status	Discharge status
4	7-7	POALOG	POA logic indicator
5	8-15	ADATE	Admission date (YYYYMMDD)
6	16-23	DDATE	Discharge date (YYYYMMDD)
7	24-223	DX1-25	Diagnosis codes (DX1=Principal)
8	224-398	PROC1-25	Procedure codes
9	399-598	PRDATES (1-25)	Procedure dates (YYYYMMDD)
10	599-637	CLAIM-OPT-DATA	39-byte field for claim description
11	638-638	DEBUG-IND	1-byte field for internal use

Field	Location	Name	Description
12	639-838	FILLER	Filler
13	839-840	RTC	Return code from the grouper
14	841-842	MDC	MDC number returned by the grouper
15	843-846	DRG	Final DRG number returned by the grouper
16	847-851	GRFLGS	Output grouper flags
17	852-1476	DXFLGS	Output diagnosis flags (25x25)
18	1477-1976	PRFLGS	Output procedure flags (25x20)
19	1977-2000	BUFF	Output ancillary buffer

**Note:** All diagnoses are left-justified in the first seven characters of an eight-character field, with each diagnosis' POA indicator occupying the eighth character of the field. Procedures are left-justified in seven-character fields. Unused characters in the diagnosis and procedure fields must be blank.

## ICD-10-CM/PCS MS-DRG Grouper Tables

Under ICD-9-CM, the MS-DRG Grouper Tables were included with the distribution of the mainframe MS-DRG grouper and were collectively referred to as the “EBCDIC Tables”. Though developed in ASCII on non-mainframe computers, the tables were necessarily converted to the EBCDIC character representation of mainframes in order to be included in the mainframe distribution.

Whether expressed in ASCII or EBCDIC, the ICD-10-CM/PCS MS-DRG Grouper Tables consist of nine files, as documented below. All of the files are comma-delimited tables relating codes to attributes.

In the documentation of the comma-delimited files below, the information between successive commas is called a field. Field 1 is the information before the first comma, field 2 is that between the first and second comma, and so on. Information itself containing a comma will be enclosed in double quotation marks. No fields so enclosed are allowed themselves to contain double quotation marks.

### Attributes

Since there are approximately 140,000 ICD-10-CM/PCS codes, with new ones being added every year, yet only a few codes on any given patient record, efficiency dictates that we transform the codes on the record into a small set of attributes, and then base the DRG-assignment logic on those attributes. An attribute is a patient-level yes/no result – either the patient record has the attribute or it does not. Hence in most implementations of the grouper, the attributes are represented with individual bits.



The grouper maintains five sets of attributes:

- Patient characteristics based on age, sex and discharge status, computed once when the patient data is provided.
- Principal diagnosis attributes, obtained by looking up the first listed diagnosis and saving its attributes from the diagnosis tables.
- Secondary diagnosis attributes, obtained by looking up each diagnosis on the record after the first, and combining the attributes found in the diagnosis tables using a Boolean inclusive-OR.
- “Any” diagnosis attributes, obtained via a Boolean inclusive-OR of the principal diagnosis attributes and the secondary diagnosis attributes.
- Procedure attributes, obtained by looking up each procedure on the record, and combining the attributes found in the procedure tables using a Boolean inclusive-OR for non-restricted procedures. Procedure clusters (see below) must also be discovered and their attributes included. This must be done each time the working MDC changes since procedure restriction is MDC dependent.

### *Procedure clusters*

When found on a patient record, some collections of ICD-10-PCS procedure codes have a different set of attributes, independent of those of the codes that make them up (their “components”). These are called procedure clusters. A routine in the grouper, upstream of the DRG assignment logic, searches the patient record for clusters. (Multiple clusters, and duplicates of the same cluster, are possible.) When a cluster is found, it is added to the list of procedures found on the record. Clusters may be “restricted” by MDC. A restricted cluster inhibits the use of its component attributes for the MDC’s DRG assignment logic.

For example, cluster 0SRT0JZ+0SPC0JZ may be recognized on the patient record if both codes appear (in any order and not necessarily together). This creates a new “procedure code” @0045. The cluster @0045 has a different set of attributes than either 0SRT0JZ or 0SPC0JZ, and is further “restricted” for MDC 08. If the grouper logic determines that the MDC is 08, it ignores the attributes of 0SRT0JZ and 0SPC0JZ and only uses those of @0045. These take the grouper to DRGs 466-468 rather than DRG 463-465. If the PDX were not for MDC 08, however, the cluster would not restrict the individual interpretation of the component codes and their own attributes could come into play as well as those of @0045.

### *Grouper logic (drglogic)*

The decisions that the grouper makes to assign a patient discharge to a DRG are specified in the DRGLOGIC file. The process for determining the DRG assignment is identified below:

1. Evaluate the claim for certain procedure or procedure combinations within PRE MDC.
2. Failing to match on any of the DRG logic described in Step 1, the software then evaluates the appropriate MDC based on the principal diagnosis, evaluating each DRG within the MDC in hierarchical order.

3. If a match is made on a DRG formula, then the appropriate grouper results are returned and grouping stops.

In versions prior to v40.0, the DRGLOGIC file took on a pseudocode feel, with the specifications resembling a computer language like C, C# or Java, but with a few special conventions to keep the file small. Users of this file were then tasked with implementing the DRG logic in their desired language/platform, following the steps described above. This file was modified by hand, and as such was error-prone, and several discrepancies between this and the actual grouper software have been identified over the years.

When the MS-DRG grouper implementation switched over to being Java based the internal development process became more table driven which necessitated the conversion of the manual process to a new database driven process. This new process reduced errors, improved efficiency, and streamlined the creation of the DRGLOGIC file. Users of this file are still tasked with implementing the DRG logic in their desired language/platform, following the steps described above.

**Table 16. DRG logic table**

Field	Contents
MDC	MDC Value. Range: 0-25
DRG	DRG Value. Range: 001-999
MEDSURG	Medical/Surgical indicator of DRG. 1=Medical 2=Surgical
BASEDRG	Base DRG value. Range: 001-999
RETURNCODE	Grouper Return Code. See table 4 (page <a href="#">11</a> ) for values and descriptions.
REROUTEMDC	The MDC to reroute to. Range: 0-25
FORMULA	DRG Formula. Attributes necessary to satisfy a DRG. See table 17 for a description of operators used.
SUPPRESSION	Attributes that should be suppressed from the severity calculation, separated by the vertical bar (   ).

The following formula operators are used throughout the DRGLOGIC file for formula operations.

**Table 17. Operators used within the FORMULA table**

Operator	Operator Description	Example	Example Description
&	AND	A & B	Codes on both list A <i>AND</i> list B
	OR	A   B	Codes on either list A <i>OR</i> list B
~	NOT	A & ~B	Codes on list A <i>AND NOT</i> list B
( and )	Group terms together	A   (B & C)	Codes on List A <i>OR</i> combination of codes on list B <i>AND</i> list C

Some DRG logic requires use of a function, or counter, to help check if the logic for a particular DRG has been satisfied. The following table contains a list of the counters and descriptions used within the DRGLOGIC, along with the lists used by the counter and the DRG(s) that use the counter.

**Table 18. Descriptions of counters used in DRGLOGIC file**

Counter Name	Description	Lists Used	Used in DRGs
SINGLEFUSIONCTR	Check if claim contains a minimum of two codes from either list and a minimum of one code from the other list.	sglantfuse sglpostfuse	426 427 428
SGLPOSTFUSECTR	Check if claim contains a minimum of two codes from the list.	sglpostfuse	426
SGLANTSECTXCTR	Check if claim contains a minimum of two codes from the list.	sglantsectxfuse	426,447
SGLANTCTR	Check if claim contains a minimum of two codes from the list.	sglantfuse	447,448
SGLPOST1CTR	Check if claim contains a minimum of two codes from the list.	sglpostfuse1	447,448

Counter Name	Description	Lists Used	Used in DRGs
BILATERAL	Check if claim contains at least one code from two different lower extremity site lists.	righthip rightknee rightankle lefthip leftknee leftankle	461,462
FOUR_STENTS	Check if claim has a combination of lists that adds up to at least 4 stents (i.e. stent1+stent3, stent4, stent2+stent3).	stent1 stent2 stent3 stent4	321
FOUR_VESSELS	Check if claim contains a combination of lists that adds up to at least 4 vessels (i.e. vessel1+vessel3, vessel4, vessel2+vessel3).	vessel1 vessel2 vessel3 vessel4	321
MULTST	Check if claim contains two or more significant traumas of different body site lists.	sthead stchest stabdom stkidney sturin stpel stuplimb stlolim	965

### Diagnosis attributes (dxattlst)

Each row of the table identifies a diagnosis attribute potentially used by the grouper logic.

**Table 19.      Diagnosis attributes**

Field	Contents
ID	Attribute number. Bits in mainframe implementation are stored left-to-right in this order.
OPERAND	Attribute name as used in <i>drglogic</i> .
DESCRIPTION	Attribute description.

*Diagnosis table (dxlist)*

Each valid ICD-10-CM diagnosis code is found in the table.

**Table 20.      Diagnosis table**

Field	Contents
CODE	Diagnosis code. ICD-10-CM codes are represented without their decimal point.
CODEGENDER	0 = fields 3 and 4 relate to both sexes. 1 = fields 3 and 4 relate to male patients only. 2 = fields 3 and 4 relate to female patients only.
EXCLUSIONGROUP	The "exclusion category" to be used to find CC/MCC exclusions when this diagnosis is used as a PDX (principal diagnosis). If 0, then no CC or MCC is excluded. For other values, see <i>exclusrn</i> .
PATTERNNUMBER	The pattern number of the row in <i>diagnosis_patterns.txt</i> which gives the attributes of this diagnosis.

*Diagnosis pattern table (dxpattern)*

Each distinct combination of MDC, diagnosis category and attributes has its own row. The pattern number is arbitrary and is used only as a link between *dxlist* and this table.

**Table 21.      Diagnosis pattern table**

Field	Contents
PATTERNNUMBER	Pattern number.
MDC	MDC.
DXCAT	Diagnosis category (DXCAT in <i>drglogic</i> ).
HACNUMBER	HAC group. 0 if diagnosis is not in a HAC group.
OPERAND	List of attributes, separated by the vertical bar (   ).

*Procedure attributes (prattlst)*

Each row of the table identifies a procedure attribute potentially used by the grouper logic.

**Table 22. Procedure attributes**

Field	Contents
ID	Attribute number. Bits in mainframe implementation are stored left-to-right in this order.
OPERAND	Attribute name as used in <i>drglogic</i> .
DESCRIPTION	Attribute description.

*Procedure table (prlist)*

Each valid ICD-10-PCS procedure code occupies one row in the table.

**Table 23. Procedure table**

Field	Contents
CODE	Procedure code or cluster. Procedure clusters are identified by an "@" sign, followed by a 4-digit number. This is a "cluster ID", which will match the cluster ID field in <i>prclstrs</i> .
PATTERNNUMBER	The pattern number of the row in <i>prpattern</i> which gives the attributes of this procedure or procedure cluster.

*Procedure pattern table (prpattern)*

Each distinct combination of procedure attributes has its own row. The pattern number is arbitrary and is used only as a link between *prlist* and this table.

**Table 24. Procedure pattern table**

Field	Contents
PATTERNNUMBER	Pattern number.
OPERAND	List of procedure attributes, separated by the vertical bar (   ). The special attribute "incluster" indicates that the procedure is in at least one cluster.

*Procedure cluster definitions (prclstrs)*

Each component of each distinct procedure cluster will have a row in this table.

**Table 25. Procedure cluster definitions**

Field	Contents
PROCCODE	Procedure code.
CLUSTERCODE	Cluster ID of one cluster the procedure is a component of.
CHOICE	The choices group for the cluster this procedure is a member of. For example, “1” means that this procedure is one of the first set of procedures listed for the cluster, a “2” means it is one of the second set of procedures listed for the cluster. A cluster may have only one procedure in a choices set. The procedures in the cluster (its components) may be recognized on the patient record in any order and need not be listed together.
NUMREQUIREDPROCS	The number of procedures required to recognize the cluster. For example, a “3” means that the cluster is recognized if there is at least one procedure on the record from its first set, one from its second and one from its third.
MDCRESTRICTION	The list of MDCs for which the cluster is restricted, separated by the vertical bar (   ). “ALL” signifies that the cluster is restricted for all MDCs. The “F” signifies that the cluster is restricted for the unrelated DRGs (DRGs 981-989) in Appendix F of the MS-DRG Definitions Manual.

***A note on cluster processing:***

Consider a claim comes in with the procedure codes in the following order (the order of the procedure codes does not matter, the internal processing for clusters will be the same):

```
0SRB019
0SPB0EZ
0SUB09Z
```

We will use PDX A0223 to illustrate, which will drive the grouping to MDC 08.

The software loops across the procedure codes in ASCII order in order to identify all clusters on the claim. In this example, it finds two:

```
0117 (attributes: OR d468 lefthip revision)
0SPB0EZ
0SRB019
```

```
0037 (attributes: OR d468 revision)
0SPB0EZ
0SUB09Z
```

These clusters are then added in the order they are found to the claim, along with their attributes, to the internal mask for grouping.

The software then moves through its DRG logic to find a DRG match based on the attributes it has found on the overall claim. The PDX will drive the grouping to look at MDC 8 logic, so it looks down that set of DRGs in hierarchical order until it finds a match.

It matches on DRG 468 because the attributes (DRG 468 requires the revision attribute) are present on the claim. Grouping stops here – the logic does not try to find any other matching DRGs.

Next the software needs to mark the codes used to get to the DRG. It loops across all of the codes (and clusters) on the claim and sees that cluster 0117 contains the necessary attributes (revision) and marks that cluster as affecting DRG assignment. The software does not mark the other cluster because it did not use it for DRG assignment. If there are multiple codes (or clusters) on a claim that have the attributes used to group to a particular DRG (in this case both cluster 0117 and 0037), only one code (or cluster) with the matching attribute is marked as affecting DRG assignment (in this case 0117).

### *CC/MCC exclusions (exclsn)*

A principal diagnosis (PDX) may exclude certain secondary diagnoses (SDX) from being considered complications/co-morbidities (CC) and/or major complications/co-morbidities (MCC). The diagnosis table in *dxlist* gives an “exclusion category” for each PDX. A zero indicates that the diagnosis has no CC/MCC exclusions. All numbers 2 and over refer to the first field of this table. All SDX listed in this table with a given exclusion category are prevented from being considered CCs or MCCs when the PDX has the exclusion category listed.

**Table 26. CC/MCC exclusions**

Field	Contents
EXCLUSIONGROUP	Exclusion category.
CODE	Excluded secondary diagnosis. ICD-10-CM codes are listed without their decimal point.



## Uploading grouper tables/files to the mainframe

### *DRG logic file*

The following steps load the DRG logic file to the mainframe.

1. Allocate a sequential dataset using the following attributes:
  - DSN=YOURID.GROUPER.**DRGLOGIC**
  - LRECL=600
  - BLKSIZE=27600
  - RECFM=FB
  - SPACE=(CYL(1),RLSE)
2. FTP the DRGLOGIC file from the miscellaneous folder in ASCII mode into a mainframe sequential dataset, "YOURID.GROUPER.**DRGLOGIC**".

### *Diagnosis attribute list*

The following steps load the Diagnoses attribute list to the mainframe.

1. Allocate a sequential dataset using the following attributes:
  - DSN=YOURID.GROUPER.**DXATTLST**
  - LRECL=120
  - BLKSIZE=27960
  - RECFM=FB
  - SPACE=(TRK(1),RLSE)
2. FTP the DXATTLST file from the miscellaneous folder in ASCII mode into a mainframe sequential dataset, "YOURID.GROUPER.**DXATTLST**".

### *Diagnosis list file*

The following steps load the Diagnosis list file to the mainframe.

1. Allocate a sequential dataset using the following attributes:
  - DSN=YOURID.GROUPER.**DXLIST**
  - LRECL=45
  - BLKSIZE=27990
  - RECFM=FB
  - SPACE=(CYL(4),RLSE)
2. FTP the DXLIST file from the miscellaneous folder in ASCII mode into a mainframe sequential dataset, "YOURID.GROUPER.**DXLIST**".

### *Diagnosis pattern list*

The following steps load the Diagnosis pattern list to the mainframe.

1. Allocate a sequential dataset using the following attributes:
  - DSN=YOURID.GROUPER.**DXPATTERN**
  - LRECL=60
  - BLKSIZE=27960
  - RECFM=FB
  - SPACE=(TRK(1),RLSE)
2. FTP the DXPATTERN file from the miscellaneous folder in ASCII mode into a mainframe sequential dataset, "YOURID.GROUPER.**DXPATTERN**".

### *CC/MCC exclusions file*

The following steps load the CC/MCC exclusions file to the mainframe.

1. Allocate a sequential dataset using the following attributes:
  - DSN=YOURID.GROUPER.**EXCLUSN**
  - LRECL=30
  - BLKSIZE=27990
  - RECFM=FB
  - SPACE=(TRK(100),RLSE)
2. FTP the EXCLUSN file from the miscellaneous folder in ASCII mode into a mainframe sequential dataset, "YOURID.GROUPER.**EXCLUSN**".

### *Procedure attribute list*

The following steps load the Procedure attribute list to the mainframe.

1. Allocate a sequential dataset using the following attributes:
  - DSN=YOURID.GROUPER.**PRATTLST**
  - LRECL=120
  - BLKSIZE=27960
  - RECFM=FB
  - SPACE=(TRK(1),RLSE)
2. FTP the PRATTLST file from the miscellaneous folder in ASCII mode into a mainframe sequential dataset, "YOURID.GROUPER.**PRATTLST**".

### *Procedure list file*

The following steps load the Procedure list file to the mainframe.

1. Allocate a sequential dataset using the following attributes:
  - DSN=YOURID.GROUPER.**PRLIST**
  - LRECL=20
  - BLKSIZE=27980
  - RECFM=FB
  - SPACE=(CYL(2),RLSE)
2. FTP the PRLIST file from the miscellaneous folder in ASCII mode into a mainframe sequential dataset, "YOURID.GROUPER.**PRLIST**".

### *Procedure pattern list*

The following steps load the Procedure pattern list to the mainframe.

1. Allocate a sequential dataset using the following attributes:
  - DSN=YOURID.GROUPER.**PRPATTERN**
  - LRECL=160
  - BLKSIZE=27840
  - RECFM=FB
  - SPACE=(TRK(5),RLSE)
2. FTP the PRPATTERN file from the miscellaneous folder in ASCII mode into a mainframe sequential dataset, "YOURID.GROUPER.**PRPATTERN**".

### *Procedure clusters file*

The following steps load the Procedure clusters file to the mainframe.

1. Allocate a sequential dataset using the following attributes:
  - DSN=YOURID.GROUPER.**PRCLSTRS**
  - LRECL=65
  - BLKSIZE=27950
  - RECFM=FB
  - SPACE=(TRK(10),RLSE)
2. FTP the PRCLSTRS file from the miscellaneous folder in ASCII mode into a mainframe sequential dataset, "YOURID.GROUPER.**PRCLSTRS**".



# Chapter 3: Using the grouper with higher-level languages

The grouper executor may be implemented as a subroutine to be called from a higher-level language program. This chapter shows how this may be done for a COBOL programming environment. To create the subroutines, you must have copied file 1 (the grouper objlib) to the mainframe (page [20](#)).

## General strategy for COBOL driving program

A typical COBOL grouping utility might operate as follows:

- Opens the input and output datasets
- Reads records from the input dataset
- Reformats and recodes the input data to a form acceptable to the grouper
- Calls the grouper
  - The COBOL wrapper (DR430J64) uses JNI and StringUtils. These Utilities can fail if not set up properly. Refer to the following resource for return code values that can indicate the cause of a failure:  
<https://www.ibm.com/docs/en/cobol-zos/6.2?topic=java-manipulating-strings>
- Stores the grouper return information on the output record
- Writes a new dataset containing the original data and the grouping information

A COBOL program (COBTST64 for 31-bit compiled COBOL calling Java 64-bit) using the sample database is included on the installation media. Refer to the sample JCL in the JCL folder.

## Input to the grouper subroutines

You must ensure that each diagnosis code is left-justified in an 8-byte field and that all of the diagnoses are in contiguous locations in the buffer whose address is in the first pointer described above. Empty fields may be interspersed throughout the buffer. A detailed discussion of the way in which fields in the buffer are processed is located at the end of this chapter.

Similarly, each procedure code must be left-justified in a 7-byte field, and all of the procedure codes must be in contiguous locations in the buffer whose address is in the third pointer described above.

Each diagnosis and procedure code must be blank-filled if it is shorter than the maximum field length. *Zero filling is not allowed.*

The patient's age must be right-justified in a 3-byte field. Valid ages for grouping are between 0 and 124. The age may be either zero- or blank-filled.

The patient's sex must be contained in a 1-byte field, in the range 0 through 2 (Unknown/Male/Female, respectively).

The discharge status must be contained in a 2-byte field which is coded according to the conventions shown in the "Required data formats" (page [8](#)) table. The code must be right-justified and may be either zero- or blank-filled.

## Output from the grouper subroutines

On return from the grouper executor, the DRG, MDC, return code, and the grouper, diagnosis, and procedure flags fields are filled in, along with the buffer of additional DRG information. The DRG and MDC numbers are right-justified. The grouper return code is filled in according to the conventions detailed in chapter 1 (page [8](#)).

## Executor processing of the diagnosis and procedure buffers

The way in which the grouper retrieves diagnosis and procedure codes for processing is to loop through the related buffers using the counts addressed by the second and fourth pointers. If any diagnosis or procedure field is all zeroes or all blanks, then that field is considered empty and the code is flagged as invalid and is ignored. Codes are saved in an internal work area that is subsequently used for construction of the record mask. Because processing is done this way, it is possible to pass a buffer that contains both valid and empty fields.

For example, assume there is a record containing a maximum of five diagnosis codes, three of which are coded for this abstract. The number of diagnoses passed would be five, and the buffer could look like any of the following:

The principal diagnoses must be in the first field of the buffer. If the field is empty or invalid, the record is assigned DRG 999 (ungroupable) with a return code of 7 (invalid principal diagnosis).

# Chapter 4: Data processing

The following sections describe the layouts for the input and output data processing.

## Program input

The total length of the input record is 638 bytes. The expected input format for this program is outlined in the following table. For more information on field details see the Required Data formats table.

**Table 27. Input format**

Field	Length	Description
CLAIM-AGE	PIC X(03)	3-byte numeric field containing the patient's age in years.
CLAIM-SEX	PIC X(01)	1-byte numeric field containing the patient's sex.
CLAIM-DISCHARGE-STATUS	PIC X(02)	2-byte numeric field containing the patient's discharge status.
CLAIM-POA	PIC X(01)	1-byte field containing the POA logic indicator
CLAIM-ADMIT-DATE	PIC X(08)	8-byte numeric field containing the patient's admission date (YYYYMMDD)
CLAIM-DISCHARGE-DATE	PIC X(08)	8-byte numeric containing the patient's discharge date (YYYYMMDD)
DIAG_CODES	PIC X(200)	200-byte buffer containing the diagnosis codes for the record to be grouped. First 7 bytes represent the diagnosis code. Left-justified, blank-filled, up to 25 accepted. The eighth byte represents the POA indicator.
PROC-CODES	PIC X(175)	175-byte buffer containing the procedure codes for the record to be grouped. Left-justified, blank-filled, up to 25 accepted.
CLAIM-PROCDATES	PIC X(200)	200-byte buffer containing the dates of the procedure codes. The buffer can hold up to a maximum of 25 dates, 8-bytes each (YYYYMMDD).
CLAIM-OPT-DATA	PIC X(39)	39-byte field for claim description
DEBUG-INDICATOR	PIC X(01)	Y = Internal interface debug (for internal use only)

## Program output

The total length of the input record is 1162 bytes. The expected output format from this program is outlined in the following table.

**Table 28.      Output format**

Field	Length	Description
OUT-CMSDRG-RTC	PIC X(02)	2-byte numeric field to hold the grouper return code
OUT-CMSDRG-MDC	PIC X(02)	2-byte numeric field to hold the MDC number
OUT-CMSDRG-DRG	PIC X(04)	4-byte numeric field to hold the DRG number
OUT-CMSDRG-OUTGRFLAGS	PIC X(05)	5-byte field to hold the grouper flags
OUT-CMSDRG-OUTDXFLAGS	PIC X(625)	625-byte field to hold the diagnosis flags
OUT-CMSDRG-OUTPRFLAGS	PIC X(500)	500-byte field to hold the procedure flags
OUT-CMSDRG-OUTBUFF	PIC X(24)	24-byte field to hold the buffer of additional DRG information



# Chapter 5: The MS-DRG grouper executor

To use the information in this chapter, you should have:

- At least a rudimentary understanding of the underlying logic on which all DRG decisions are based.
- Access to the *Medicare Severity Diagnosis Related Groups Definitions Manual*, which explains the principles on which all decisions are made.

The executor essentially makes its decisions by comparing indicators for each DRG within an MDC. Indicators are set by the elements found on the patient record. These sets of indicators are referred to as masks. The content of the masks is listed in the MS-DRG grouper tables (page [24](#)).

The tables are represented as hexadecimal constants in the Jar and are present in memory when the grouper is loaded for execution. All table lookups are in-memory binary searches.

The executor begins its basic task by creating masks that are indicative of the conditions found on the patient record. These are called the record masks.

Once the record masks have been constructed, the corresponding DRG masks for the MDC indicated by the principal diagnosis are compared to them, until a match is found or the DRG masks for the MDC are exhausted.

Because the internal format of the grouper tables is optimized in the Jar for fast lookups and is therefore difficult to read, the nine principal tables included in Jar are provided as flat files on the media. See chapter 2 (page [17](#)) for table layout details.

## Construction of the record mask

The following list describes how the executor constructs the record masks.

1. Sex is tested for validity (0-2).
  - An error indicator is turned on if sex is out of that range.
  - If not, the appropriate indicator is set in the record mask.
2. Discharge status is tested for validity (01-07, 20, 21, 30, 43, 50, 51, 61-66, 69, 70, 81-95)
  - An error indicator is turned on if discharge status is out of range.
  - Otherwise, the appropriate indicators are set in the record mask.
3. The first listed diagnosis (assumed principal) is looked up in the Diagnosis Table.
  - If no entry is found, the record is assigned DRG 999, RTC 7 and no further processing occurs.
  - If an entry is found, but the MDC number is 0, the record is assigned DRG 999, RTC 7 and no further processing occurs.

- Otherwise, the MDC and DXCAT are saved and the indicators for this diagnosis code are moved to the mask where principal diagnosis indicators are positioned.
  - 4. All secondary diagnoses are looked up in the Diagnosis Table and their bit indicators “OR’d” together in the mask reserved for secondary diagnosis indicators. Additionally, if any of the secondaries is a complication or comorbidity, the CC exclusion subroutine is called to determine if the CC flag in the record mask should be set. A complete discussion of the CC exclusion subroutine appears later in this chapter (page [43](#)).
- Any secondary diagnosis for which there is no Diagnosis Table entry does not cause an error, but is instead ignored. MDC and DXCAT numbers are of no importance for secondaries.
- 5. Once all diagnoses have been processed, the indicators for principal and secondary are “OR’d” together in yet another indicator section mask for ALLDX criteria.
  - 6. All procedure codes are looked up in Procedure Table and their bit indicators “OR’d” together in the mask reserved for procedure indicators. As with secondary diagnoses, invalid procedure codes do not generate errors, but are ignored.

## DRG determination

Once the record masks have been constructed, the executor loops through the DRG masks for the MDC indicated by the principal diagnosis, comparing them with the record masks.

1. The comparison is done by moving the record mask to a work area and ANDing it with the current DRG mask.
2. The result of the ANDed work mask is then compared with the DRG mask.
  - If the results are identical, the associated DRG number is assigned and the processing to find and return the diagnosis and procedure flags is executed.
  - Otherwise, looping continues until a match is found or the DRG list is exhausted, at which time DRG 999 is assigned.

The rest of this section discusses some special conditions in the grouper logic.

### Testing for the ONLY surgery condition

When the DRG mask indicates that ONLY specific surgeries can be present, the executor loops through the saved O.R. surgeries from the record, making decisions as follows:

1. The O.R. portion of the DRG mask is moved to a work area.
2. The work mask is ANDed with the mask of the saved O.R. surgery.
  - If the result of the ANDing is zero, this indicates that the surgery found on the record is other than the ONLY surgery allowed. The executor ceases looping and gets the next DRG mask.
  - Otherwise, the process continues until all saved O.R. surgeries have been tested.

## Testing for the ONLY DX condition

The testing for this condition is virtually identical to that done for the ONLY surgery condition, except that the comparison is done on saved diagnoses against the ALLDX portion of the DRG mask.

## Testing for the OWISE condition

This condition exists for DRGs 794, 963-965 and 997. This is essentially the “fall through” DRG for the MDC and is assigned when no other DRG criteria have been met. The “anydx” bit in the DRG mask is turned on, leaving a mask with only that bit on, thereby guaranteeing a match.

## Testing for the ANYCOMB condition

This condition exists only for DRG 461-462 in MDC 8. The test is done by comparing all coded O.R. procedures with the procedure portion of the DRG mask and adding one to an accumulator for each procedure that has a matching mask. If the resulting count is less than two, this record does not meet the “anycomb” condition, and the next DRG mask is retrieved.

## CC exclusion subroutine

A large subset of the diagnosis codes are flagged as complication/comorbidity codes (CC) or major complication/comorbidity codes (MCC). Many of these codes are not really CC/MCC codes at all times because there are many conditions for which the secondary diagnosis is a natural side effect of the principal diagnosis. The CC/MCC exclusion table is organized to reflect a direct relationship between a principal diagnosis and selected secondaries.

Because the ICD-10-CM codes are non-contiguous and do not lend themselves well to defining ranges of codes, an index number is associated with each diagnosis and the CC/MCC exclusion table is constructed entirely from those index numbers.

To determine whether a secondary should be considered a CC/MCC, the executor accesses the CC/MCC table, using the principal diagnosis CC/MCC exclusion category as the key each time a secondary flagged as CC/MCC is encountered.

- If no entry is found for the exclusion category, that means that there are no exclusions and the secondary is considered a CC/MCC code.
- If an entry is found, then the secondary is excluded as a CC/MCC.

## Testing for the OTHOR condition

This test is similar in logic to the test for the ONLY conditions, except that it tests for procedures in addition to the O.R. criteria in the DRG mask. When the DRG mask indicates that other O.R. procedures must be present, the executor loops through the O.R. procedures from the record, making decisions as follows:

1. The O.R. portion of the DRG mask is moved to a work area.
2. The work mask is ANDed with the mask of the saved O.R. procedure.
  - If the result of the ANDing is zero, this indicates that the procedure is other than the specific procedure required (e.g., T&A) and therefore satisfies the other O.R. criteria. When that occurs, looping ceases and processing continues for the DRG.
  - Otherwise, the loop continues until a procedure satisfies the other condition. If all saved procedures are exhausted without finding one that satisfies the other condition, then processing for that DRG is ended.

## Testing for illogical principal diagnosis

When a DRG has been matched, and the DRG number is 999, the cause is an illogical principal diagnosis. To indicate this, the return code is changed to 6.

## Testing for multiple significant trauma

The principal diagnosis is tested to see if it is a trauma code. If it is, processing continues to test for multiple significant trauma. Otherwise, no further trauma testing is done.

To qualify as multiple significant trauma, two significant trauma codes from different body sites must be present. The diagnosis mask contains special trauma indicators, with each body site trauma represented by a different flag.

The mask of the first diagnosis (either principal or secondary) that is flagged as a significant trauma is saved. The mask of each subsequent diagnosis that is also flagged is compared with the initial saved mask. If they are not the same, the record is flagged as a multiple significant trauma episode. If they are the same, the next diagnosis is tested until the multiple condition is satisfied or the diagnoses are exhausted.

## Finding codes that affect Initial DRG assignment

After the DRG has been determined, the grouper executor analyzes the saved diagnosis and procedure masks, comparing them against the masks which were used to determine MDC and DRG. Codes which were necessary for the determination of the MDC/DRG are flagged with an "affect flag."

## Final DRG

If no Hospital Acquired Conditions (HACs) are found on the record, then the initial DRG becomes the final DRG. Otherwise, the record is re-grouped demoting the HAC secondary diagnosis which may or may not change the DRG assignment based on what DRG it was initially assigned to, and/or the presence of other codes that are CCs or MCCs.

# Appendix A: Grouping results for the test database

The JCL folder contains a file, INSTLCNT, that contains the number of records in the test database. The printout in your output should match record counts. The Sysout has the number of records processed with no fallouts. The test database used a POA indicator of Z. There were no POAs assigned to the diagnosis codes.

The test, when performed on an IBM® Z15 8561-T01 computer, used 192K of virtual storage and took less than 1 CPU second.