DRGGroupers Software & Services

A Catalog, Manual and Reference

By **Brendan F. Hemingway**
Lead Developer, DRGGroupers

REVISED April, 2020 (v37b) *CMS V37.1 Release 1*

Library of Congress Control Number: applied for

ISBN 0-9819852-0-3

Printed in the United States of America

Cow and Calf Publishing
5 Spring Road
Branford, CT 06405

Table of Contents

Chapter 1 - *Introduction* 9
What Is New In This Version 9
ICD10 DRGs (v34 and Above) 9
DRG Basics 10
DRG Historical Note 10
Reimbursement 11
Federal DRGs 12
Other DRG Definitions 12
Diagnosis & Procedure Codes 13
Assigning DRGs 14
DRG Properties 15
New in 2008: POA, HAC 15
POA 15
HAC 15
How To Use This Book 16
Syntax Notation 16
Link to M+H Consulting 17

Chapter 2 - *DRG Assignment Software* 18
POA, HAC and Exempt Status 18
How Groupers Work 18
Input 18
Process 18
Output 18

Return Codes 20

Chapter 3 - *Our Product Line* 21

Chapter 4 - *DRG Assignment Service* 22
What Is The DAS? 22
Required Data Elements 22
Version Processing Options 23
Turn-Around Time 23
How Do I Send My Data? 23
What Do I Get Back? 24
What Do I Do Next? 24
Pricing and Custom Orders 25

Chapter 5 - *DRGFilt* 21
Staying Current 26
How to Use a Filter 26
Command Line Arguments 27
Control File 27
Job Control 28
Input Specification 29
Output Specification 30
Installation 31
Typical Scenario 31
Super Control File 32

Chapter 6 - *Excel-DRG* 37
Requires the DLL and Masks 37

Staying Current 26
Technical Details 38
Columns of the Spreadsheet 38
Column A: ID Number 38
Column B: Age 38
Column C: Sex 38
Column D: Discharge Disposition 39
Column E: Exempt flag 39
Column F-AD: Diagnoses 39
Column AE-BC: POA flags 39
Column BD-CB: Procedures 39
Column CC: DRG Version 39
Column CD: Use POA indicator 39
Column CE: DRG 40
Column CF: DRG Description 40
Column CG: Grouper Return Code 40
Column CH: MDC 40
Column CI: DRG Weight 40
Column CJ: GMLOS 40
Detailed Instructions 40
Installation 31

Chapter 7 - *Access-DRG* 42
Requires the DLL and Masks 37
Staying Current 26
Technical Details 38
Typical Usage 43

Chapter 8 - *Grouper DLL* 37
Staying Current 26
Technical Details 38
Distribution 45
Calling DLL Functions 45
VB Example: DECLARE 45
C Example: Linking 46
VB DLL API 46
MHDLLVER 47
MHDRG 47
Return Code -1 49
VB Sample Code 49
C Sample Code 53

Chapter 9 - *Perl Shared Object* 57
Staying Current 26
Technical Details 38
Distribution 45
Calling Perl SO Functions 58
Sample Perl Code 58

Chapter 10 - *PHP Shared Object* 61
Staying Current 26
Technical Details 38
Distribution 45
Calling PHP SO Functions 62
Sample PHP Code 62

Chapter 11 - *C-Callable Object* 65
Staying Current 26
Technical Details 38
Distribution 45
Calling CO Functions 66
Sample C Code 66

Appendix A: New in v37b 70
CMS Bulletin 70
Lowering AIX Support 70
Our Tech Blog 70

Appendix B: Return Codes 71

Appendix C: Discharge Status 72

Appendix D: Software Licenses 73
Reseller License 73
Non-standard Licenses 73
Contact Information 73
Standard Client Software License 73
Standard Server Software License 74

Glossary 75
Classification 75
CPT Codes 75
DRG 70

DRG Grouper 76
DRG Version 76
Federal DRGs 77
HAC (Hospital Acquired Complication) 77
ICD9cm Codes 77
ICD10 Codes 77
Major Diagnostic Category (MDC) 78
Masks File 78
POA (Present on Admission) 79
Return Code 71
Significant Code Bit String (dflg and sflg) 79

Index 80

Introduction

This book is provided by DRGGroupers as a service to our customers. It is intended to help our customers in any or all of three different ways:

1. as a catalog of our software and services
2. as a manual of how to use our software or services
3. as an introductory reference to DRGs

This book is not intended as a general reference for Diagnosis Related Groups (DRGs). For general information about DRGs we refer you to Wikipedia's DRG entry,

`en.wikipedia.org/wiki/Diagnosis_Related_Group`

and to our own on-line Frequently Asked Questions (FAQ) page for DRGs,

`drggrouperstechblog.blogspot.com/p/faq_04.html`

What Is New In This Version

If you are already familiar with our products or you just want to know what is new with this version, please skip to Appendix A.

While this book's main focus is our product line, we often are asked general questions about DRGs and different kinds of DRGs, so we provide our basic answers here to guide prospective customers (and avoid having to answer these questions on the phone or in email!)

ICD10 DRGs (v34 and Above)

On the first of October every year CMS (Centers for Medicare and Medicaid Services, formerly HCFA) releases a new version of the DRGs. So the 2016 DRGs were released on 01-October-2016.

The original DRGs were based on ICD9cm codes. After years of postponement, DRGs based on ICD10 codes were released, starting with v34. This was a relatively small conceptual change but it was a major change in the software realm. We had to revamp our product line in order to accomodate it.

Most importantly, we shifted our products from backward compatibility to single-version support. This meant adding the version number ot the names of our produects and simplifying our product line.

DRG Basics

A DRG classifies an inpatient hospital stay on the basis of diagnoses, procedures, age, gender and discharge status into one of several exclusive groups. For v36, there are 761 DRGs, numbered from 1 to 999 with some DRGs undefined.

There are three special DRG values: 0 (which means "not grouped") and 998 ("PRINCIPAL DIAGNOSIS INVALID AS DISCHARGE DIAGNOSIS") and 999 (which means "ungroupable"). The rest of the DRG values have descriptions, weights, LOS outlier trim points and mean LOS all of which depend on the DRG version. (Prior to v33, DRG 470 was "ungroupable.")

One step of the DRG assigment algorithm is assigning an inpatient stay to an MDC (Major Diagnostic Category). Every MDC leads to a limited list of DRGs. MDCs are sometimes used in analyses. There is a pre-MDC which has no number (we call it MDC zero to make our software developers' lives easier) and 25 MDCs numbered 1 through 25. Each DRG falls into one and only one MDC. The DRGs are not evenly distributed by MDC; for instance MDC 5 has 94 DRGs but MDC 20 has only 4.

DRG Historical Note

The original DRGs were invented at Yale University's Health Systems Management Group (HSMG) in the late 1970s. The principal researchers were John Thompson, a nursing guru, and Bob Fetter, an Operations Research kind of guy. Ron Mills, co-founder of the parent of DRGGroupers, was the technical lead and he was the one

who created the biostatistical analysis package, AUTOGRP, which made the underlying research possible in real-time.

DRGs were adopted by the United States federal government's Health Care Finance Administration (HCFA) and first released in 1982 as version 2 (version 1 was the unreleased version which HCFA evaluated). Every year, on October 1st, HCFA releases a new CMS DRG version.

In 2001, the United States federal government's Health Care Finance Administration ("HCFA") became "the Centers for Medicare & Medicaid Services" or "CMS".

In 2016, CMS completed the migration from ICD9cm codes to ICD10.

DRGs are good for providing a context in which to analyze hospital stays. DRGs were designed to allow hospitals to operate on a more industrial basis, with resource allocation and cost-center analysis, all of which were very hip in the late 1970s when DRGs were created. In a nutshell, DRGs predict likely resource consumption for any given hospital stay, allowing one to determine if the given hospital stay was too short, too long or just right.

Reimbursement

Originally, DRGs were designed to predict Length of Stay and were not concerned with reimbursement. However, following a study of their effectiveness as predictors of overall hospital resources required to treat different kinds of patients, DRGs were chosen by Medicare as the basis of the Prospect Payment System for hospitals. Since DRGs hit the scene as part of a reimbursement scheme, DRGs became linked with reimbursement in many people's minds.

Since DRGs measure resource consumption in the form of a normalized weight, using DRGs for reimbursement not only makes sense, it is easy: you multiply the DRG-specific weight by the facility-specific factor and voila! you have a reimbursement amount for a given inpatient stay. However, this additional step is called "pricing" and is not part of the grouper per se; it is a separate process which is not part of grouping. Software which makes this calculation is called a "pricer." For convenience, most pricer

providers bundle the DRG grouper in with their software, which had confused grouping and pricing in many people's minds.

We are hardly experts on buying pricing software, but if you are looking to buy it and are stuck, check out 3M™ Health Information Systems. They seem to have lots of pricers out there in the world, so someone is buying them.

Federal DRGs

In the United States of America, the "official" definition of DRGs is the one defined by CMS (formerly HCFA). Strictly speaking, the CMS grouping algorithm is public, and anyone can implement it in software. (There are books published so that one could even do without the software and assign DRGs by hand.) However, CMS has blessed NTIS as the distributor of the reference grouper, which is written in IBM 360 Mainframe assembler. If you have an IBM 360-compatible computer, you can buy that grouper through NTIS and run that.

Other DRG Definitions

While we usually mean "US Federal DRGs" when we say "DRGs," there are many different governments which have defined their own version of DRGs. New York state defined their own. New Jersey defined their own for a while. France has their own, as does Portugal. Australia recently joined the club with their own version.

There are so many different DRG definitions because the federal DRGs are a bit of a compromise: The creators of the federal DRGs were constrained by the number of data elements that they could reasonably expect any given hospital in the country to collect. Furthermore, their baseline population is all Medicare patients, which skews the results somewhat.

As a result, the CMS DRGs are unambitious with respect to severity of illness and resource consumption and not appropriate to all hospital populations.

Many groups have tried to extend the basic DRG concept to fix these flaws. 3M™/HIS sells AP-DRGs ("All Payor" DRGs). Yale University's School of Medicine came up with RDRGs ("Refined"

DRGs). CMS itself is working on SDRGs ("Severity-adjusted" DRGs).

Since DRGGroupers is constantly asked about RDRGs®, we asked the nice folks at HSC to give us a blurb to put on our website to answer this question. Here is their reply:

The RDRG severity-of-illness software is a product of Health Systems Consultants, Inc. in New Haven, Connecticut. The software groups inpatient hospital discharge data into DRGs and into severity classes within DRGs. The DRGs produced are identical to those of the public domain DRG grouper from the Health Care Financing Administration (HCFA--now CMS). The software assigns patients to 511 DRGs and to 1198 Refinement Group (RGN) numbers and is updated each year to conform to the CMS DRGs. Since the software system can predict hospital resource use, it can be used to improve hospital casemix analysis, analyze hospital performance, evaluate physician performance, measure quality, develop budgets, and to reimburse hospitals.

The RDRG severity-of-illness software was developed from a Yale University study funded by CMS (formerly HCFA) entitled, "DRG Refinement with Diagnostic Specific Comorbidities and Complications: A Synthesis of Current Approaches to Patient Classification." The study, completed in 1989, was designed to adjust the DRG system for the severity of a patient's illness. For information about the RDRG software, please contact Karen Schneider at

`karen.Schneider@healthsyst.com`

or call Health Systems Consultants at (203) 785-0650.

Diagnosis & Procedure Codes

The official federal grouper, version 2 through 32 inclusive, only accepts ICD9cm codes (International Committee on Diseases, version 9, Clinical Modifications) for both diagnoses and procedures and ICD10 codes starting with v33. However, the American Medical Association has defined an alternative scheme for coding procedures, which they call CPT (Current Procedural Terminology). Many providers have chosen to code even in-house procedures using CPT. But if you want to group with CPT codes as input, then

you have to convert them to ICD9cm codes first. This conversion is not a simple one-to-one mapping. Many vendors sell CPT-to-ICD "crosswalks," but DRGGroupers is not one of them.

Assigning DRGs

The next chapter in this book will focus on DRG assignment software, so we will give only a cursory treatment here.

A DRG Grouper is a computer program or module which takes those 5 clinical and demographic data as input and gives a corresponding Diagnosis Related Group as output. The diagnoses and procedures are encoded as either ICD9cm (versions 2-32) or ICD10 (version 33 and above). The age is a small integer from 0 to 129. The sex is encoded as 1 for male, 2 for female and 3 for unknown (don't ask). The discharge status, also known as "discharge disposition," is usually encoded either using UHDDS or UB92 (both medical billing standards).

The standard CMS (formerly HCFA) grouper, ours included, will accept up to 10 diagnoses (versions 2 through 32, inclusive or up to 25 after v32), the first of which is presumed to be the Primary diagnosis. Likewise, up to 15 procedures (versions 2 through 32, inclusive or up to 25 after v32) are accepted, but their significance is determined by the grouping process, so their order is not important.

The relevance of any diagnosis or procedure code is determined by its attributes, which attributes are implemented as a "bit mask" or "mask" or "bit string." (You may see any of these terms in our documentation.)

The attributes guide the grouper in its use of any given code; for instance, the attributes say whether or not a code is gender-specific, or if it is allowed as a primary diagnosis. In addition to the information encoded in the attributes, the grouper applies logic to actually classify any given inpatient stay into a single DRG.

DRG Properties

Every DRG has certain properties. Those properties are:

- A DRG description (70 characters wide, version-dependant)
- An MDC (see the glossary for details)
- A Geometric Mean Length of Stay (GMLOS)
- A Weight (a normalized prediction of resource consumption)
- A Category: either "Surgical" or "Medical"
- A low "trim point" (the LOS below which lie the low outliers)
- A high "trim point" (the LOS above which lie the high outliers)

The DRGGroupers grouper returns all these and more: a set of flags for each of the Diagnosis Codes and Procedure Codes so that the caller can determine which codes were actually significant to the grouping.

New in 2008: POA, HAC

As of October 1st, 2008, there are two new concepts in DRG Assignment:

- A diagnosis attribute of "Present on Admission" (POA)
- Hospital Acquired Complications (HAC)

POA

The federal goverment is trying to get away from paying for medical mistakes; a common analogy is not paying a mechanic for a new car window if that mechanic accidentily breaks that window while fixing your brakes.

In order to avoid paying for medical mistakes, diagnoses are now flagged as having been present on admission (POA).

The values for the POA flag are not, as one might expect, simply Y or N; rather the following options are defined:

- Y for Yes -N for No -U for Unspecified -W for clinically undetermined -1 for unreported / not used / exempt from reporting

HAC

The logic which embodies handling the POA conditions is called the Hospital Acquired Complications ruleset. Some hospitals are exempt from the HAC, so the DRG assigment software has to be able to accept a flag which indicates that the institution from which these data come is exempt.

How To Use This Book

We expect that, in the usual case, the reader will start with this introductory section and then go to the chapter on the specific product about which the reader wishes to know more.

Note that the chapters often contain supporting information such as sample code for calling program modules or screen shots of interactive products.

However, we understand that the specific chapter may contain unfamiliar jargon, so we provide the glossary at the end of the book. We also understand that the specific chapter may contain unfamiliar concepts, so we provide a few explanatory chapters which immediately follow this introduction.

Syntax Notation

When giving the syntax of commands, we try to follow the standard notation. To us this means that optional arguments are written within square brackets, [like this], and required arguments are written within curly braces, {like this}.

```
command-name [optional] {required}
```

Lists of possibilities are given within vertical bars, like so:

```
[a|b|c]
```

which means "a or b or c";

Iteration is denoted by elipses, like so:

```
command-name {mode} [file...[file]]
```

which means "the command is called 'command-name', the mode argument is required and you don't have to give a file argument and you can give more than one file argument".

Link to M+H Consulting

DRG Groupers is a business unit of M+H Consulting. M+H Consulting is a medical information systems company who created DRG-related software to help them deliver service to their clients. A number of clients wanted to buy the software without needing any services, so M+H created DRGGroupers as a way of staying focused on their primary market. M+H continues provide software and technical support to DRG Groupers.

For more information on M+H Consulting, see their web site:

```
www.mhconsulting.com
```

DRG Assignment Software

Software to assign DRGs is also called a "DRG Grouper" or just a "grouper" for short.

If you are looking for general information about DRGs, please refer to the previous chapter in general and the section "DRG Basics" in particular.

If the jargon used here is unfamiliar to you, perhaps you will find definitions in the brief glossary of terms at the end of this book.

POA, HAC and Exempt Status

As of version 26, which was released in October of 2008, the notions of Present on Admission and exempt-from-Hospital-Acquired-Complications were introduced into DRG assigment. Our software supports these ideas if desired, and can ignore them as well, if required.

How Groupers Work

Like any other piece of software, groupers can be described with the Input / Process / Output model.

Input

There are five different parameters to a grouper which are needed for every Electronic Medical Record (EMR) to which a DRG is to be assign. Those parameters are:

- Patient age on admission, expected to be between 0 and 124
- Patient sex, coded: 1=male, 2=female
- Discharge status coded in either the UB92 standard or the UHDDS standard

- A list of up to twenty-five ICD10 diagnosis codes, in order of significance
- A list of up to twenty-fix ICD10 procedure codes, in order of significance (and sometimes called "surgery codes")

In order to assign a DRG, a grouper needs access to a list of bit masks for every possible ICD10 code and code cluster. This list is called "the masks file" and is DRG version-specific.

Process

The grouper uses these inputs to determine a DRG by applying both a specific DRG version's logic and the masks. As part of the process, a Major Diagnostic Category (MDC) is determined as well.

Output

In theory, a grouper could simply spit out a DRG for every input record. In practice, ther are a number of other, related, data elements that the user finds useful:

- A return code, indicating success or failure of the grouping attempt
- The MDC
- The DRG
- The Classification of the DRG
- A bit string of which Diagnosis codes were significant
- A bit string of which Procedure codes were significant

Return Codes

A return code of zero means that nothing went wrong. Any other value denotes a specific problem. For a list of the return codes for any DRGGroupers product, please refer to Appendix B.

Our Product Line

DRGGroupers sells both consulting services and grouper software ("groupers").

Our consulting services are documented elsewhere; this chapter is about our groupers.

We sell an application, DRGFilt, which is intended to be used by users to assign DRGs to arbitrary data sets. We support Linux, AIX and Windows. You buy DRGFilt and then run it on your data.

We sell embedable and callable DRG Assignment Engines which allow programmers to a DRG assignment to their software or data environments. For example, as proof-of-concept, we used our VB-callable DLL to add DRG assignment to an excel spreadsheet and to an MS-Access application and then we made these into products because many Excel users and MS-Access programmers did not want to do the integration themselves.

For our programmer tools, we support either Linux or Windows in our on-line store. If you want to support some other environment or help in adding DRG assignment to your product or service, we are happy to do that but through our consulting services.

We use an outside porting lab named Ready to Run to provide us with UNIX platforms on which to develop and test, so for an up-to-date list of UNIX variants that we support, please see their web site:

`www.rtr.com`

DRG Assignment Service

If you want DRGs assigned to some records and you don't want to buy, install and run the software yourself, then our DRG Assignment Service (DAS) might be just what you are looking for.

Many of our customers are medical information analysts who do not have an on-going need for DRG assignment. Instead, they have the occasional dataset which they would like run through a grouper.

For these customers, we started a service which we call our DRG Assignment Service (DAS).

What Is The DAS?

The DAS works like this: you send us your dataset with some information about the file format and the record format. With that information, our software can find the data elements used in assigning DRGs. We run your dataset through our grouper and return the results to you along with a DRG summary report which gives you an overview of your dataset's DRG profile: a frequency distribution of DRGs assigned and errors encountered.

Required Data Elements

1. Patient Sex, usually represented by 'F', 'M' and 'U'

2. Patient Age at admission, or Date of Birth and admission date, from which we can calculate age at admission

3. Patient discharge disposition; see the Glossary for details. (We can default to 'unknown' if your data does not include this element.)

4. Before v33: Up to ten ICD9cm diagnosis codes, in order of importance

5. Starting with v33: Up to twenty-five ICD10 diagnosis codes, in order of importance

6. Before v33: Up to fifteen ICD9cm procedure codes, in order of importance

7. Starting with v33: Up to twenty-five ICD10 procedure codes, in order of importance

Version Processing Options

As part of the standard DAS, we offer two processing options: either we will use any supported DRG version on all your records or we will use the discharge date (which must be included in your dataset) to determine which federal DRG version was in use at the time of discharge and then use that version. Other methodologies for determining what version to use are possible, but would fall under the category of custom set-up programming and would require a custom quote.

Turn-Around Time

We guarantee three business days, but we can often deliver same day turnaround time if data is transferred electronically in both directions. For on-line transfers, we prefer SCP but will use FTP or email as required.

How Do I Send My Data?

You can send your data to us in any of several common formats:

- Microsoft Access database, either Access 2007 or later;
- ASCII or EBCDIC text file with one record per line (either variable-width fields with a separator or fixed-width fields are ok);
- SQL INSERT statements in a text file (either ASCII or EBCDIC);
- MySQL database dump.

What Do I Get Back?

As part of the standard DAS, we will return any combination of the following data elements as fields in the output dataset:

1. Grouper Return Code (what error, if any, arose during grouping);
2. DRG Version used to group the record;
3. DRG Number assigned the record;
4. Major Diagnostic Category (MDC) of the assigned DRG;
5. The resource consumption weight of the assigned DRG;
6. The official description of the assigned DRG.

What Do I Do Next?

That depends on which return option you specify. The return options are:

1. At the beginning of the output dataset;
2. At the end of the output dataset;
3. As contiguous fields starting at a given field number;
4. As separate records with the same ID as the grouped input record (assuming that the input record has an ID field and that you have told us which field to use as the ID).

If we send you the DRG as part of your dataset, then you load we send you ON TOP OF your existing data. This would be any of the first three options listed above.

If we send you a separate DRG dataset, the fourth option listed above, then you load that separate dataset into a new table and link your existing data to that table by means of the record identifier.

Pricing and Custom Orders

In order to keep things simple all the way around, we offer this service as a flat fee by number of records in the input. If you can provide your data in any of the supported input formats mentioned above, then you qualify for the flat fee schedule. You can find the latest fee schedule at

`www.drggroupers.net/services`

If you cannot provide your data in one of these formats, it is likely that we can still help you, but you would have to contact us for a custom price quote. You can find our up-to-date contact information on-line:

`www.drggroupers.net/contact-us/`

DRGFilt

If you are looking to assign DRGs to a dataset in batch mode, then DRGFilt might be just what you are looking for.

DRGFilt is also configurable as a web service and as the basis of a CGI script.

DRGFilt was originally implemented as a UNIX-style "filter," ie a console application which reads from standard input and writes to standard output and writes its error messages to standard error. Non-filters usually read from a file and write to file.

With version 33, DRGFilt's control file format and support was greatly expanded when we switch to an INI file form. Now DRGFilt jobs can be set up with all the parameters in the INI file. This format is described in detail below. Note that the original filter functionality is still available.

Staying Current

With the move to the ICD10-based algorithm we changed our products to support a single version so the supported version is now in the name, eg "drgfiltv36" or "drgfiltv36.exe" etc.

How to Use a Filter

There are two ways to use a filter: either as an application or a step in a chain of processes.

Example of a filter as a process:

% filter < in-file > out-file

In this example, the program "filter" is reading from a file called "in-file" and writing to a file called "out-file".

Example of a filter as a link in a chain of processes:

% prog1 arg1 | filter | prog2

In this example, the chain starts with a program called "prog1" which takes an argument, "arg1". The output of prog1 is passed as input to a program called "filter" and the output of the program called filter is passed as input to prog2.

A typical use of DRGFilt is to assign DRGs to exported data. Almost any database management system (DBMS) can export data as text and import text as data. So the database administrator (DBA) usually does the following:

1. export the data to be grouped as a comma separated value file (CSV). Let us call this file "the export CSV."

2. run the export CSV through DRGFilt, creating new file with the old data and the DRG in it. Let us call this file "the import CSV."

3. import the import CSV into the DBMS, either overwriting the data in the database or just importing the DRG by matching on a key

Command Line Arguments

DRGFilt's calling syntax is as follows:

drgfiltvXX {control-file.ini} [ini-file-group-name] [-q | -v | -l log-file]

Because DRGFilt is a filter, you can also call it in this way:

prog1 | drgfiltvXX {control-file.ini} [ini-file-group-name] [-q | -v | -l log-file] | prog2

or

prog1 | drgfiltvXX {control-file.ini} [ini-file-group-name] [-q | -v | -l log-file] > output

The first argument is required: the name of the control file. See "Control File" below.

The second argument is optional: name of the group within the control file which will be used. This argument is optional because this argument defaults to "drgfilt".

In any position, "-q" means "quiet" or "minimal informational output" while "-v" means the opposite: "verbose" or "maximal informational output". In any position, "-l" should be followed by a valid file name to which informational output will be directed.

Control File

The control file controls how DRGFilt behaves. The model is the old-time mainframe "job control" file: you can specify all variable aspects of input, process and output.

The format is a version of the Windows .INI file format because we find this a clear and simple format for parameter specification.

Each job definition starts with a name, which is inside square brackets [like this]. (The .INI file folks call this a "group" so we do too.) Parameters are then specified in the "name = value" format, one per line. While parameters can be given in any order, we like to follow a simple convention to make the files easier for human to read:

```
[group-name]
job control
input specification
output specification
```

Job Control

These are parameters which control DRGFilt's operation.

The "fixed-example" job processes the CMS test data set. The "csv-example" job processes the same test data set, but reformatted as a CSV which we find much more common in the real world.

```
[fixed-example]
format = fixed          ; input file format: either fixed or csv
verbose = 0             ; no debugging information
```

```
blip = 1000             ; give progress report every 1,000 records
blipeol = 0             ; 1=newline for progress report, 0=carriage return
batchver = 33           ; assign version 33 DRGs to this batch
maskdir = .             ; directory in which to find drgmasks.v33
                        ; no input file or output file, so this job is a filter

[csv-example]
format = csv (,)        ; separator character is in parens
base = 0                ; field indices are zero-based, as opposed to 1-based,
inheaders = 1           ; 1=input column headers, 0=no input column headers
outheaders = 1          ; 1=want column headers, 0=no column headers
crlf = 0                ; type of end-of-line: 0=crlf (DOS), 1=lf (Unix)
blip = 1000             ; give progress report every 1,000 records
maskdir = .             ; directory in which to find drgmasks.v33
batchver = 33           ; the DRG version to apply to this entire batch
infile = adt.csv        ; input file name
outfile = x.csv         ; output file name
verbose = 1             ; give debugging output
```

Input Specification

In order to group, DRGFilt needs the five inputs mentioned earlier: patient sex, patient age, at least 1 and no more than 25 diagnosis codes, anywhere from 0 to 25 procedure (surgery) codes, and a discharge status. The input specification tells DRGFilt how to get these elements out of the input.

```
; fixed-example, continued
; fixed-width input variables: name = length@offset
age =   3@0
sex = 1@3
ds =  2@4
dxl=8           ; dx length does not require an offset
poa= 7
dx = 200@23
sgl=7@0         ; pr length, offset is optional & ignored
surg= 175@223
exmp = 1@6
; -----
; these are already present from the CMS grouper:
; you could overwrite the incoming values if you wanted to
; drg = 3@603
```

```
; mdc = 2@600
; rc = 2@598
; -----

; csv-example, continued
; input variables: name = index
inid = 0
bdt =  1
sex = 2
exmp = 3
ds =  4
; these there keywords are allowed to have lists as parameters.
; lists can contain single entries, ranges, or both. for example
; 1,2,7-22,50
dx = 5-29
poa = 30-54
surg= 55-79
adt = 80
; calcver gives the index of a field to be used as the date from which
; we calculate the appropriate DRG version. If any record's calculated
; DRG version does not match the batchver, we skip that record
calcver = 80            ; NOTE: same index as "adt" because we are using the sa
```

Output Specification

The control file also tells DRGFilt where you want it to write the results of its grouping.

```
; fixed-example, continued
; these are written out by DRGFilt
rc  = 2@1760
mdc = 2@1762
drg = 3@1764

; csv-example, continued
; these are written out by DRGFilt
outid = 0               ; patient ID from input, whatever inid pointed to
rc  = 1
mdc = 2
drg = 3
desc = 4
weight = 5
```

```
morp = 6
outver = 7
```

Installation

To install DRGFilt, you put the DRGFilt executable in a directory in the users' search path (often /usr/local/bin), and you put the masks file(s) and control file(s) in a directory which is readable by the users (often /usr/local/drggrouper or /usr/share/drggrouper).

Typical Scenario

How is DRGFilt ususally used? Here is the typical scenario:

1. A database administrator (DBA) changes the database schema to include a DRG for every record (and perhaps other DRGFilt outputs as well, often the return code and version).

2. The DBA exports a dataset from a database management system, creating the file which is input to DRGFilt. Let us call this file "the export file."

3. An analyst creates or customizes a control file for DRGFilt which matches the format of the export file.

4. A programmer or power user runs the export file through DRGFilt which creates output, which we will call "the results file."

5. The DBA imports the results file, overwriting the contents of the database.

And, voila! every record now has a DRG assigned to it.

Super Control File

Here is the control file we use to exercise every DRGFilt feature, which should give you examples of all the new features:

```
; new format DRGFilt control file, with different groups for different purposes

;---------------
; this group is to process the standard fixed-width input file from CMS
;---------------
[fixed]
format = fixed          ; input file format
verbose = 0             ; no debugging information
blip = 1000             ; give progress report every 1,000 records
blipeol = 0             ; 1=newline for progress report, 0=carriage return
batchver = 33           ; assign version 33 DRGs to this batch
maskdir = .             ; directory in which to find drgmasks.v33

; fixed-width input variables: name = length@offset
age =   3@0
sex = 1@3
ds =  2@4
dxl=8           ; dx length does not require an offset
poa= 7
dx = 200@23
sgl=7@0         ; pr length, offset is optional & ignored
surg= 175@223
exmp = 1@6
; -----
; these are already present from the CMS grouper:
; you could overwrite the incoming values if you wanted to
; drg = 3@603
; mdc = 2@600
; rc = 2@598
; -----

; these are written out by DRGFilt
rc  = 2@1760
mdc = 2@1762
drg = 3@1764

;---------------
; this group is to validate date-handling of a CSV
;---------------
```

```
[csv-adt]
format = csv (,)        ; separator character is in parens
base = 0                ; field indices are zero-based, as opposed to 1-based,
inheaders = 1           ; 1=input column headers, 0=no input column headers
outheaders = 1          ; 1=want column headers, 0=no column headers
crlf = 0                ; type of end-of-line: 0=crlf (DOS), 1=lf (Unix)
blip = 1000             ; give progress report every 1,000 records
maskdir = .             ; directory in which to find drgmasks.v33
batchver = 33           ; the DRG version to apply to this entire batch
infile = adt.csv        ; input file name
outfile = x.csv         ; output file name
;outfile = blank file.csv
verbose = 1             ; give debugging output

; input variables: name = index
inid = 0
bdt =  1
sex = 2
exmp = 3
ds =  4
; these there keywords are allowed to have lists as parameters.
; lists can contain single entries, ranges, or both. for example
; 1,2,7-22,50
dx = 5-29
poa = 30-54
surg= 55-79
adt = 80
; calcver gives the index of a field to be used as the date from which
; we calculate the appropriate DRG version. If any record's calculated
; DRG version does not match the batchver, we skip that record
calcver = 80            ; NOTE: same index as "adt" because we are using the sa

; these are written out by DRGFilt
outid = 0               ; patient ID from input, whatever inid pointed to
rc  = 1
mdc = 2
drg = 3
desc = 4
weight = 5
morp = 6
outver = 7

;---------------
; this group is to process the CSV we created from the standard fixed-width inp
```

```
;---------------
[csv]
format = csv (,) ; separator character is in parens
base = 0 ; indices are zero-based
;base = 1 ; indices are one-based
inheaders = 1           ; 1=input column headers, 0=no input column headers
outheaders = 1 ; 1=want column headers, 0=no headers
crlf = 0 ; type of end-of-line: either crlf or lf
blip = 1000
batchver = 33
maskdir = .
infile = testdbv33.csv
outfile = testdb.out.csv

; input variables: name = index
inid = 0
age =  1
sex = 2
exmp = 3
ds =  4
dx = 5-29
poa = 30-54
surg= 55-79

; these are written out by DRGFilt
outid = 0 ; patient ID from input
rc  = 1
mdc = 2
drg = 3
desc = 4
weight = 5
morp = 6
outver = 7

;---------------
; this group is to process the CSV we created from the standard fixed-width inp
;---------------
[csv-validate]
format = csv (,) ; separator character is in parens
base = 0 ; indices are zero-based
;base = 1 ; indices are one-based
inheaders = 1           ; 1=input column headers, 0=no input column headers
outheaders = 0 ; 1=want column headers, 0=no headers
crlf = 0 ; type of end-of-line: either crlf or lf
```

```
blip = 1000
blipeol = 0 ; 1=newline for progress report, 0=carriage return
maskdir = .
batchver = 33
infile = testdbv33.csv
outfile = testdb.validate.csv
verbose = 1

; input variables: name = index
inid = 0
age =  1
sex = 2
exmp = 3
ds =  4
dx = 5-29
poa = 30-54
surg= 55-79

; these are written out by DRGFilt
outid = 0 ; patient ID from input
drg = 1

; this group was added to support drgfilt-as-CGI helper
[cgi]
verbose = 0;
format = csv (^)        ; separator character is in parens
base = 0                ; indices are zero-based
inheaders = 0           ; 1=input column headers, 0=no input column headers
outheaders = 0          ; 1=want column headers, 0=no headers
crlf = 0                ; type of end-of-line: either crlf or lf
blip = 1                ; want something in the log
blipeol = 1             ; 1=newline for progress report, 0=carriage return
batchver = 33
maskdir = /var/www/cgi-bin/drgstuff
log = /tmp/me.out

; input variables: name = index
inid = 0
age =  1
sex = 2
exmp = 3
ds =  4
dx = 5-29
poa = 30-54
```

```
surg= 55-79

; these are written out by DRGFilt
rc  = 0
mdc = 1
drg = 2
outver = 3
weight = 4
mean = 5
morp = 6
desc = 7
dflg = 8                ; string of flags for which dx codes were used
sflg = 9                ; string of flags for which pr codes were used
; eof
```

Excel-DRG

Excel-DRG is for end users who are comfortable using Microsoft Excel and whose data is either already in a spreadsheet or can be put into a spreadsheet.

If you are not comfortable using Excel or if you cannot easily get your data into and out of spreadsheets then Excel-DRG is not for you.

Note: as of version 26, released in 2008, the DRG version can have either a 'p' or an 'e' or both appended to it. If there is a trailing 'p', it is assumed that the last character of every diagnosis code is a POA flag. If the there is a trailing 'e', then the source institution is presumed to be exempt from the HAC. Thus "26p" specifies POA support, while "26" does not.

"25p" does not make sense because before version 26, there was no POA or HAC concept.

"26pe" specifies POA support (ie there are flag) but that the institution is exempt.

If you specify a version after 26 and you do not want POA and HAC support, it is safer to specify "XXe" rather than just "XX", eg "30e" instead of just "30".

Requires the DLL and Masks

Excel-DRG depends on our DLL for Visual BASIC and our masks files, both of which are included in the initial purchase price.

Note that you will have to refer to the chapter on the Grouper DLL for information about return codes, inputs and outputs.

Staying Current

Because Excel-DRG calls our DLL to do the actual grouping, you can keep it up-to-date by buying a new DLL every year and leaving Excel-DRG alone, assuming that the DLL API remains the same.

Technical Details

Excel-DRG is a Microsoft® Excel spreadsheet and macro which calls our Grouper DLL on every row. The results of the DLL call populate columns in the spreadsheet.

Our grouper DLL has the same inputs and outputs as all our other groupers. For details about how our groupers work, see chapter 2, "DRG Assignment Software." In particular, there is a section on inputs to our grouper and its outputs.

Typically, the user makes a copy of the original spreadsheet and works with the copy for every grouping project. The user then pastes or imports her data into the copy, sets the version column and calls the macro.

Columns of the Spreadsheet

The following is a brief description of the various columns of the spreadsheet.

Column A: ID Number

This column exists to link this data to your original dataset. A better name would have been "primary key".

Column B: Age

Required grouper input. See Chapter two for details.

Column C: Sex

Required grouper input. See Chapter two for details.

Column D: Discharge Disposition

Required grouper input. See Chapter two for details.

Column E: Exempt flag

Required grouper input. See Chapter two for details.

Column F-AD: Diagnoses

Up to twenty-five diagnosis codes. Required grouper input. See Chapter two for details.

Column AE-BC: POA flags

Up to twenty-five Present On Admission codes, corresponding to the diagnosis codes: the first POA is for the first diagnosis code, the second POA is for the second diagnosis code, and so on. Required grouper input. See Chapter two for details.

Column BD-CB: Procedures

Up to twenty-five procedure codes. Required grouper input. See Chapter two for details.

Column CC: DRG Version

Required grouper input. See Chapter two for details.

Column CD: Use POA indicator

Tells the grouper whether or not to expect POA flags. See Chapter two for details.

Column CE: DRG

Optional grouper output.

Column CF: DRG Description

Optional grouper output.

Column CG: Grouper Return Code

Optional grouper output.

Column CH: MDC

Optional grouper output.

Column CI: DRG Weight

Optional grouper output

Column CJ: GMLOS

GMLOS stands for "Geometric Mean Length of Stay" and is often abbreviated to just "LOS". Optional grouper output.

Detailed Instructions

Open your copy of mhdrgvb35.xls. The spreadsheet contains the CMS test dataset. Before deleting these rows and populating the spreadsheet with your own data, we recommend that you try to group the test records. To do this, hit Ctrl-Shift-D (for DRG). You should hear a beep and see a message that the records were grouped. Next, you can delete these rows. DO NOT ADD COLUMNS, DELETE COLUMNS, OR MOVE COLUMNS! Next, populate the spreadsheet with your own data following the format specified by the first row (column headers). Once you have copied your data into the spreadsheet, run the grouper by hitting Ctrl-Shift-D.

Installation

Excel-DRG is delivered as a zip file. In the zip file you will find:

1. mhdrgvb36.xls (the spreadsheet itself)

2. vbdrgv36.exe (windows installer for the DLL)

In order for the Excel grouper to work, you must install both the DLL (which automatically installs the masks file) and the Excel spreadsheet. Double-click on mhdrgvb.exe to install the DLL. Double-click on vbdrgv36.exe to install both DLL and the masks file.

Once those are installed, you can populate the spreadsheet with discharge data and run the Grouper. We highly recommend that you copy mhdrgvb35.xls and use the copy as your working spreadsheet.

Access-DRG

Access-DRG is an interactive grouper.

Access-DRG is for Microsoft® Access users who are comfortable using MS-Access and whose data is either already in an MS-Access database or can be put into an MS-Access database.

If you are not comfortable using MS-Access or if you cannot easily get your data into and out of MS-Access databases, then Access-DRG is not for you.

Note: as of version 26, released in 2008, the DRG version can have either a 'p' or an 'e' or both appended to it. If there is a trailing 'p', it is assumed that the last character of every diagnosis code is a POA flag. If the there is a trailing 'e', then the source institution is presumed to be exempt from the HAC. Thus "26p" specifies POA support, while "26" does not. "25p" does not make sense.

Note: as of version 33, released in 2015, our API was expanded to include an exempt flag and a POA indicator. See the appendix entitled "What's New" for details.

Requires the DLL and Masks

Access-DRG depends on our DLL for Visual BASIC and our masks files, both of which are included in the initial purchase price.

Staying Current

Because Access-DRG calls our DLL to do the actual grouping, you can keep it up-to-date by buying a new DLL every year and leaving Access-DRG alone.

Note: as of version 33, you may require an update to Access-DRG in order to assign DRGs to versions other than the version which was current when you made the purchase.

Technical Details

Access-DRG is a Microsoft® Access database and VBA module which calls our Grouper DLL on every row. The results of the DLL call populate columns in the database.

Our grouper DLL has the same inputs and outputs as all our other groupers. For details about how our groupers work, see chapter 2, "DRG Assignment Software." In particular, there is a section on inputs to our grouper and its outputs.

Typical Usage

Typically, one populates the database in Access-DRG however one likes (text file importation, ODBC calls, etc) and then fills in the form and then pushes the button to assign DRGs to every row.

After the DRG assignment has taken place, the DRGs are available either for export or for dynamic access.

Grouper DLL

The Grouper DLL is a programmer's tool.

The DLL is the standard core grouping engine wrapped in a Win32-friendly package. This means that the DLL takes essentially the same input as every other grouping product and gives essentially the same output. Therefore you can use technical information provided about the inputs and outputs of the other grouping products and apply that information to this product.

Sadly, C and BASIC have different function calling conventions. This means that a DLL compiled for C cannot be used by BASIC. Thus all our DLLs come in two flavors. Please be sure that you order the right one.

Choose the VB-Callable DLL (vbdrgv36.dll) to run with Access and Visual BASIC applications.

Choose the C-Callable DLL (mhdrg.dll) to run with applications created with Visual C++ (or some other Win32 C compiler).

Note: as of version 26, released in 2008, the DRG version can have either a 'p' or an 'e' or both appended to it. If there is a trailing 'p', it is assumed that the last character of every diagnosis code is a POA flag. If the there is a trailing 'e', then the source institution is presumed to be exempt from the HAC. Thus "26p" specifies POA support, while "26" does not. "25p" does not make sense.

Staying Current

In order to stay up-to-date you have to buy a new one every year, which includes the appropriate masks file.

Technical Details

The Grouper DLL is a Win32 Dynamic Link Library, written in ANSI C and compiled either for use with Visual BASIC software or for use with Visual C++ software.

Distribution

The Grouper DLL is distributed with an installer to allow you to install or uninstall it cleanly.

Calling DLL Functions

Whenever you call a function from a DLL, you must specify the following:

1. the name of DLL file

2. the function prototype, ie what arguments the function takes as input and what result the function gives as output.

The syntax for specifying this information varies from programming language to programming language, but the essential information to be conveyed is the same across all programming languages.

VB Example: DECLARE

In Visual BASIC, the statement you need is the DECLARE statement. Refer to the documentation that came with your VB compiler for details.

Here is a DECLARE statement that worked for us in VB 5:

```
1: Private Declare Function mhicd Lib "mhicdvb.dll" ( _
2:     ByVal which As String, _
3:     ByVal file As String, ByVal code As String,  _
4:     ByVal retval As String, _
5:     ByVal length As Integer) As Integer
```

All five lines are actually a single statement; the underscores at the end are continuation marks.

Line 1 specifies the DLL (mhicdvb.dll) and the function name (mhicd).

Line 2 specifies that the first argumeent to mhicd() is called "which", is passed by value and is a string.

Line 3 specifies that the second argumeent to mhicd() is called "file", is passed by value and is a string.

Line 4 specifies that the third argumeent to mhicd() is called "retval", is passed by value and is a string.

Line 5 specifies that the fourth argumeent to mhicd() is called "length", is passed by value and is an integer. This line also specifies that mhicd() itself returns an integer value.

C Example: Linking

The ways in which DLLs are exposed to applications varies between different C, or Visual C++, or C# implementations. You will have to consult the documentation for your particular environment for the details of calling out to DLL functions from your software.

We use LCC-Win32 and in this environment, the secret is all in the linking:

```
lc -ansic tryit.c mhdrg.lib mhicd.lib -o tryit.exe
```

Note the references to mhdrg.lib and mhicd.lib; these are stubs which allow the application to call out mhdrg.dll and mhicd.dll respectively.

VB DLL API

The Application Program Interface (API) describes how to call the various functions in the DLL. The API described in this section is the VB-callable DLL or the C-callable DLL.

MHDLLVER

The function mhdllver() is provided to give the programmer access to both the highest support DRG version and the internal versions of the DLL itself and its grouper logic.

```
Dim VerStr As String * 80
Dim MaxVer As Integer

'get the DLL version
MaxVer = mhdllver(VerStr, 79)
```

There are two parameters: a buffer into which to put the version string and a maximum length of that buffer. Note that you have to allocate the memory for the buffer outside the call because this function does not allocate memory for you.

In Visual BASIC, this means using the DIM command with an explicit length.

MHDRG

In-place assignment means that a function takes a point to a variable in the main program and then sets the variable in the main program through that pointer. The alternative is returning a value to the main program.

The function mhdrg() actually assigns the DRG and returns the results. In order to return multiple values from a single function call, this function does mostly in-place assignment. In-place assignment requires that the DECLARE statement for this function NOT use the BYVAL keyword for these parameters

In order to get around the different ways in which different environments handle binary numbers, most of the parameters are strings even if they represent numbers. Thus patient age is a string, not an integer.

```
ReturnCode = mhdrg(drg, mdc, myver, masksdir,_
    mydstat, myage, mysex, mydxbuf, mypxbuf)
```

There are eight parameters:

1. a pointer to an integer into which to put the DRG;

2. a pointer to an integer into which to put the MDC; *** NEW IN V30 RE-RELEASE: MDC IS NOW SOMETIMES DEPENDANT ON THE THE PRIMARY DIAGNOSIS AND SO MUST BE RETURNED AS PART OF THE DRG ASSIGNMENT *** If the DRG has the pre MDC, which we represent as 0, then mhinfo() will return 0 for that DRG, but you will not know the particular MDC for any given instance of that DRG. *** NEW IN V31 RELEASE: mhinfo() NOW LEAVES THE MDC VALUE AS-IS, FOR BACKWARD COMPATIBILITY.

3. a pointer to a string into which to put the DRG version;

4. a pointer to a string into which holds the full path to the directory in which to find the masks files;

5. a pointer to a string into which holds the Discharge Status;

6. a pointer to a string which holds the patient age;

7. a pointer to a string which holds the patient sex (1=male, 2=female, 0=unknown);

8. a pointer to a string which holds the diagnosis codes, separated by commas *** NEW IN V30 RE-RELEASE: VB-callable DLL diagnosis string ends with '^' and each code can end with a tilde (~) and then the POA flag

9. a pointer to a string which holds the procedure codes, separated by commas *** NEW IN V30 RE-RELEASE: VB-callable DLL procedure string ends with '^'

Note: as of version 26, released in 2008, the DRG version can have either a 'p' or an 'e' or both appeneded to it. If there is a trailing 'p', it is assumed that the last character of every diagnosis code is a POA flag. If the there is a trailing 'e', then the source institution is presumed to be exempt from the HAC. Thus "26p" specifies POA support, while "26" does not.

"25p" does not make sense because before version 26, there was no POA or HAC concept.

"26pe" specifies POA support (ie there are flag) but that the institution is exempt.

If you specify a version after 26 and you do not want POA and HAC support, it is safer to specify "XXe" rather than just "XX", eg "30e" instead of just "30".

Return Code -1

The DLL has a special return code: minus 1 (-1). This code is a generic "initialization failure" code, denoting any of the following conditions:

- missing "sex" parameter
- missing "age" parameter
- missing "version" parameter
- missing "masks path" parameter
- "version" parameter < lowest supported version or > highest supported version

VB Sample Code

What follows is a chunk of Visual BASIC which calls our Grouper DLL:

```
Option Explicit

Private Declare Function mhdllver Lib "vbdrgv36.dll" (ByVal Buf As String, _
    ByVal BufLen As Integer) As Integer
Private Declare Function mhdrg1 Lib "vbdrgv36.dll" (drg As Integer, _
    ByVal DRGVersion As String, ByVal MasksPath As String, ByVal DischStat As S
    ByVal PtAge As String, ByVal PtGender As String, ByVal DXList As String, _
    ByVal ProcList As String, ByVal POAPresent As String, ByVal ExemptFlag As S
Private Declare Sub mhinfo Lib "vbdrgv36.dll" (ByVal drg As Integer, _
    ByVal DRGVersion As String, ByVal MasksPath As String, ByRef mdc As Integer
```

```
    weight As Double, los As Double, ByVal Desc As String, ByVal DescLen As Int
Private Declare Function mhdrgver Lib "vbdrgv36.dll" (ByVal MPath As String, _
    ByVal Buf As String, ByVal BufLen As Integer) As Integer
Private Declare Sub mherrdesc Lib "vbdrgv36.dll" (ByVal errBuffer As String, By

Public Function AssignDRG()

On Error GoTo Err_Group

    Dim ReturnCode As Integer
    Dim drg As Integer, mdc As Integer
    Dim Desc As String * 80
    Dim weight As Double, los As Double
    Dim masksdir As String
    Dim myver As String, mydstat As String, myage As String, mysex As String, m
    Dim mydxbuf As String * 256
    Dim mypxbuf As String * 256
    Dim tempStr As String
    Dim N As Integer, needcomma As Integer
    Dim Val As String
    Dim NumRecords As Long, NumErrors As Long
    Dim LastRow As Long
    Dim MyID As Long                                    'user's record number
    Dim myWS As Object

    Application.ScreenUpdating = False
    Application.Cursor = xlWait
    'start in first row, DRG column
    Range("CE2").Select
    NumRecords = 0
    NumErrors = 0

    'get the path to the masks directory out of the registry, if you can
    masksdir = "C:Program FilesndHsks
    Set myWS = CreateObject("WScript.Shell")
    tempStr = myWS.RegRead("HKLMtwarendH0
    If Len(tempStr) > 0 Then
        masksdir = tempStr & "sks
    End If

    'loop through records making sure DRG info is blank
    'first, find last row
    Do Until IsEmpty(ActiveCell.Offset(0, -81).Range("A1").Value) = True
        ActiveCell.Offset(1, 0).Range("A1").Select
```

```
Loop

'select DRG columns and all populated rows
LastRow = ActiveCell.Row
LastRow = LastRow - 1
Range("CE2:CJ" & LastRow).Select

'clear selection
Selection.ClearContents

Range("CE2").Select
'loop through records assigning drg info
Do Until IsEmpty(ActiveCell.Offset(0, -82).Range("A1").Value) = True
    myver = ActiveCell.Offset(0, -2).Range("A1").Value
    MyID = ActiveCell.Offset(0, -82).Value

    'abort if version is blank
    If myver = "" Then
        MyID = ActiveCell.Offset(0, -82).Value
        MsgBox "Version is empty for record # " & MyID & ". Cannot group.",
        NumErrors = NumErrors + 1
        GoTo NextOne
    End If

    myexempt = ActiveCell.Offset(0, -79).Range("A1").Value
    mydstat = ActiveCell.Offset(0, -78).Range("A1").Value
    mysex = ActiveCell.Offset(0, -80).Range("A1").Value
    myage = ActiveCell.Offset(0, -81).Range("A1").Value
    mypoa = ActiveCell.Offset(0, -1).Range("A1").Value

    'Loop through controls, getting their current values

    'make string out of the diagnosis codes
    tempStr = ""
    needcomma = 0
    For N = -77 To -53
        Val = ActiveCell.Offset(0, N).Range("A1").Value
        If Len(Val) > 0 Then
            'append POA flag, if present
            If ActiveCell.Offset(0, N + 25).Value <> "" Then
                Val = Val & "~" & ActiveCell.Offset(0, N + 25).Range("A1").
            End If
            If needcomma <> 0 Then
                tempStr = tempStr & ","
```

```
                End If
                needcomma = 1
                tempStr = tempStr & Val
            End If
        Next N
        mydxbuf = tempStr & "^" ' explicit end-of-data marker

        'make string out of the procedure codes
        tempStr = ""
        needcomma = 0
        For N = -27 To -3
            Val = ActiveCell.Offset(0, N).Range("A1").Value
            If Len(Val) > 0 Then
                If needcomma <> 0 Then
                    tempStr = tempStr & ","
                End If
                needcomma = 1
                tempStr = tempStr & Val
            End If
        Next N
        mypxbuf = tempStr & "^" ' explicit end-of-data marker

        'call the M+H grouper with what you got
        ReturnCode = mhdrg1(drg, myver, masksdir, mydstat, myage, mysex, mydxbu
        ActiveCell.Offset(0, 2).Range("A1").Value = ReturnCode

        If ReturnCode <> 0 Then          'drg assignment failed, alas!
            Call mherrdesc(Desc, 80)
            ActiveCell.Offset(0, 0).Range("A1").Value = drg
            ActiveCell.Offset(0, 1).Range("A1").Value = Desc
            NumErrors = NumErrors + 1
        Else                            'drg assignment worked, hurray!
            'get the particulars of this DRG from M+H dll
            Call mhinfo(drg, myver, masksdir, mdc, weight, los, Desc, 80)
            ActiveCell.Offset(0, 0).Range("A1").Value = drg
            ActiveCell.Offset(0, 1).Range("A1").Value = Desc
            ActiveCell.Offset(0, 3).Range("A1").Value = mdc
            ActiveCell.Offset(0, 4).Range("A1").Value = weight
            ActiveCell.Offset(0, 5).Range("A1").Value = los
        End If
NextOne:
        ActiveCell.Offset(1, 0).Range("A1").Select
        NumRecords = NumRecords + 1
    Loop
```

```
    Application.ScreenUpdating = True
    Application.Cursor = xlDefault
    Range("CE2").Select

    Beep
    MsgBox NumRecords & " records were grouped with " & NumErrors & " error(s).

Exit_Group:
   Exit Function

Err_Group:
    Application.Cursor = xlDefault
    MsgBox Err.Number & "-" & Err.Description
    Range("CE2").Select
    Resume Exit_Group

End Function
```

C Sample Code

What follows is a trivial C program which calls our Grouper DLL:

```
/* tryit.c (c) 2002 M+H Consulting, LLC C-callable demo (BFH) */
/* Thu Oct  8 19:22:54 EDT 2015 BFH update for v36 */
/* Tue Nov 18 11:52:35 EST 2014 BFH test case for huron debugging */
/* Sun Oct 14 18:08:14 EDT 2012 BFH test case for v30 bug */
/* Sun Oct  3 11:57:46 EDT 2010 BFH support f28 */
/* 10/06/2008 made win32 and unix versions the same */
/* 10/11/2003 added bit-string of used dx's and used procedures */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define DELIM "^"
#define RETURNED 10
#define DRGVER "v36p"

#ifdef WIN32
/* standard library function */
extern char * strtok(char *, char *);
```

```
/* DLL functions */
//extern int mhicd();           /* mhicd.dll */
extern char * mhdrg1(char *, char *, char *, char *, char *, char *, char *, in
extern char * mherrdesc(void);  /* cdrgv36.dll */
#else
extern char * errdesc();        /* cdrgv36.dll */
extern char * mherrdesc(void);  /* cdrgv36.dll */
extern char * mhdrg1();         /* cdrgv36.dll */
#endif

int main() {
    char * retval;
    char * p;
    char * a[RETURNED];
    char * expected;
    int i;

    expected = "DRG: 614        MDC: 10         GRC: 00";

    /* call the grouper, store results in reval */
    retval = mhdrg1(
        DRGVER,                 /* which DRG version you want */
#ifdef WIN32
        "/mnh/dev/dll-i10/v36", /* path to masks files */
#else
        "/drbd/mnh/src/SWIG",   /* path to masks files */
#endif
        "1",                    /* discharge status */
        "55",                   /* patient age on admission */
        "1",                    /* patient sex (1=male, 2=female) */
        /* ICD DX codes */
      //"D352   YE2740  NN390   YE871   NF05    NR630   NR110   YR0602  YD649
        "B5881  YI2101   ",
        /* ICD procedure codes */
      //"0GB00ZZ0YUY47Z02HR30Z0L8S3ZZ0P5N0ZZ0RJP0ZZ",
        "021009302UG3JZ",
        8,                      /* length of each ICD DX code */
        7,                      /* length of each ICD procedure code */
        "y",                    /* there are POA flags */
        "0"                     /* this case is not exempt from HAC rules */
    );

    /* if there was an error, alert the user */
    if (retval == NULL) {
```

```
#ifdef WIN32
        printf("M+H grouper argument or environment error: %s0, mherrdesc());
#else
        printf("M+H grouper argument or environment error: %s0, errdesc());
#endif
        exit(-1);
    }

    /* show the raw return value */
    printf("Raw return value from mhdrg1:55s0,retval);

    /* deconstruct return from mhdrg1() */
    p = (char *) strtok(retval,DELIM);
    for (i = 0; i < RETURNED && p != NULL; i++) {
        a[i] = p;
        p = (char *) strtok(NULL,DELIM);
    }

    /* last two return values are bit-strings of which ICD codes were
     * used in the grouping
     */

    puts("Deconstructed return value from mhdrg1:");
    printf("rc=%s, mdc=%s, drg=%s, ovn=%s, weight=",a[0],a[1],a[2],a[3]);
#ifdef OLD_WAY
    printf("%s, mean=%s, porm=%s0esc=%s0x flags:%ld, px flags: %ld0,a[4],a[5],a
#else
    printf("%s, mean=%s, porm=%s0esc=%s0x flags:%s, px flags: %s0,a[4],a[5],a[6
#endif

    /* show what we should get */
    printf("0xpected result: %s0,expected);

    /* last two return values are bit-strings of which ICD codes were used in t
    puts("0ignificant ICD codes:");
#ifdef OLD_WAY
    for (mask = 1L,i = 0; i < 32; i++, mask *= 2L) {
        if (mask & dxu) {
            printf(" * DX code %2d was used0,(i+1));
        }
    }
    for (mask = 1L,i = 0; i < 32; i++, mask *= 2L) {
        if (mask & pxu) {
            printf(" * Proc code %2d was used0,(i+1));
```

```
        }
    }
#else
    p = a[8];
    for (i = 0; i < 32 && p[i] != ' 00'; i++) {
        if (p[i] == '1') {
            printf(" * DX code %2d was used0,(i+1));
        }
    }
    p = a[9];
    for (i = 0; i < 32 && p[i] != ' 00'; i++) {
        if (p[i] == '1') {
            printf(" * Proc code %2d was used0,(i+1));
        }
    }
#endif
    exit(0);
}

/* This program should produce the following output:
 -------------------------
Raw return value from mhdrg1:
0^15^391^21^0.1536^3.10^M^NORMAL NEWBORN

Deconstructed return value from mhdrg1:
rc=0, mdc=15, drg=391, ovn=21, weight=0.1536, mean=3.10, porm=M
desc=NORMAL NEWBORN
dx flags:1, px flags: 0

Significant ICD codes:
 * DX code  1 was used
-------------------------
 */
/* eof */
```

Perl Shared Object

The Perl shared object is a programmer's tool. If you want to include this product as part of software that you sell or lease outside your organization, you will need a Reseller's License. Please refer to Appendix D for details.

Since the Perl shared object (SO) is a compiled object, it is platform-specific. You have to buy the Perl SO that was compiled for your specific platform. Currently, we only supply the SO under UNIX and UNIX-derived operating systems.

We use an outside porting lab named Ready to Run to provide us with UNIX platforms on which to develop and test, so for an up-to-date list of UNIX variants that we support, please see their web site:

`www.rtr.com`

Note: as of version 26, released in 2008, the DRG version can have either a 'p' or an 'e' or both appended to it. If there is a trailing 'p', it is assumed that the last character of every diagnosis code is a POA flag. If the there is a trailing 'e', then the source institution is presumed to be exempt from the HAC. Thus "26p" specifies POA support, while "26" does not. "25p" does not make sense.

Staying Current

In order to stay up-to-date you have to buy a new Perl SO every year.

Technical Details

The Perl SO is an interface, written in ANSI C, from which a wrapper was created with SWIG, an open source project to support wrapper generation.

Distribution

The Perl SO is distributed as a file to be installed in the appropriate directory with the appropriate permissions. Since the directory in which the Perl intepreter looks for shared objects varies from installation to installation, we cannot give precise installation instructions.

Calling Perl SO Functions

To call a function in a Perl SO all you have to do is load the SO with the Perl "use" command. Then you can call functions within the SO as if the SO were a Perl package.

Sample Perl Code

What follows is a test program we use to make sure that the Perl SO is working. The package in which the Perl SO is wrapped is called called 'mhdrg'.

The call to mhdrg1() should be self-explanatory. You will, of course, have to substitute the correct version number, path to masks files and grouper inputs and output for your installation and your specific data.

mhdrg1() takes the usual inputs and gives the usual outputs. For details about how our groupers work, see chapter 2, "DRG Assignment Software." In particular, there is a section on inputs to our grouper and its outputs.

```
# tryit.pl (c) 2002 M+H Consulting, LLC MH DRG test script
# Wed Oct  3 12:36:40 EDT 2018 BFH upgrade from v356 to v36
# Sun Oct 22 18:12:37 EDT 2017 BFH upgrade from v34 to v35
# Tue Oct  4 14:50:57 EDT 2016 BFH upgrade from v33 to v34
# Sun Oct 14 18:32:29 EDT 2012 BFH v30 bug test case
# Wed Sep 14 06:53:33 EDT 2011 BFH command line option to use source dir for ma
use perldrgv36;                 # M+H DRG package

$ver = "v36";
#$expected = "DRG: 614  MDC: 10         GRC: 00";

# call grouper, store results in $retval
```

```
#$retval = mhdrg1(
#        $ver,                  # which DRG version you want
#        "/drbd/mnh/src/SWIG",  # path to masks files
#        "1",                   # discharge status */
#       "25",                   # patient age on admission */
#       "2",                    # patient sex (1=male, 2=female) */
#       # ICD DX codes */
#       "M154   Y",
#       # ICD procedure codes */
#       "0SRF0J90SRF0KZ",
#       8,                      # length of each ICD DX code */
#       7,                      # length of each ICD procedure code */
#       "1",                    # are there POA flags?
#       "n",                    # is this case exempt?
#    );
$retval = mhdrg1(
        $ver,                   # which DRG version you want
        "/drbd/mnh/src/SWIG",   # path to masks files
        "1",                    # discharge status */
        "52",                   # patient age on admission */
        "2",                    # patient sex (1=male, 2=female) */
        # ICD DX codes */
        "I440   Y",
        # ICD procedure codes */
        "",
        8,                      # length of each ICD DX code */
        7,                      # length of each ICD procedure code */
        "1",                    # are there POA flags?
        "z",                    # is this case exempt?
    );

# if there was an error, alert the user
if ($retval eq '') {
    $msg = "M+H grouper argument or environment error: " . perldrgv36::errdesc(
    print STDOUT $msg;
    die $msg;
}

# just for prettiness, remove trailing blanks
$retval =~ s/;

# show the raw return value
printf STDOUT ("Raw return value from mhdrg1:59s0,$retval);
```

```
# deconstruct return from mhdrg1()
($rc,$mdc,$drg,$ovn,$weight,$mean,$porm,$desc,$dxflags,$pxflags) = split(//,$re
print STDOUT "Deconstructed return value from mhdrg1:0;
print STDOUT "rc=$rc, mdc=$mdc, drg=$drg, ovn=$ovn, weight=";
print STDOUT "$weight, mean=$mean, porm=$porm0esc=$desc,dxflags=$dxflags,pxflag
print STDOUT "EXPECTED: $expected0;
exit(0);

# This script should produce the following output:
#-------------------------
# Raw return value from mhdrg1:
# 0^15^391^20^0.1517^3.10^M^NORMAL NEWBORN
#
# Decontructed return value from mhdrg1:
# rc=0, mdc=15, drg=391, ovn=20, weight=0.1517, mean=3.10, porm=M
# desc=NORMAL NEWBORN
#-------------------------
```

PHP Shared Object

The PHP shared object is a programmer's tool. If you want to include this product as part of software that you sell or lease outside your organization, you will need a Reseller's License. Please refer to Appendix D for details.

Since the PHP shared object (SO) is a compiled object, it is platform-specific. You have to buy the PHP SO that was compiled for your specific platform. Currently, we only supply the SO under UNIX and UNIX-derived operating systems.

We use an outside porting lab named Ready to Run to provide us with UNIX platforms on which to develop and test, so for an up-to-date list of UNIX variants that we support, please see their web site:

`www.rtr.com`

Note: as of version 26, released in 2008, the DRG version can have either a 'p' or an 'e' or both appended to it. If there is a trailing 'p', it is assumed that the last character of every diagnosis code is a POA flag. If the there is a trailing 'e', then the source institution is presumed to be exempt from the HAC. Thus "26p" specifies POA support, while "26" does not. "25p" does not make sense.

Staying Current

In order to stay up-to-date you have to buy a new PHP SO every year.

Technical Details

The PHP SO is an interface, written in ANSI C, from which a wrapper was created with SWIG, an open source project to support wrapper generation.

Distribution

The PHP SO is distributed as a file to be installed in the appropriate directory with the appropriate permissions. Since the directory in which the PHP intepreter looks for shared objects varies from installation to installation, we cannot give precise installation instructions.

Calling PHP SO Functions

To call a function in a PHP SO all you have to do is load the SO with the PHP "dl" command. Then you can call functions within the SO as if the SO were a PHP package.

In our case, we use the following idiom that we found on the Internet:

```
if (!extension_loaded('mhdrg')) {
    if (!dl('mhdrg.so')) {  // this should be in a protected directory
        exit;
    }
}
```

Sample PHP Code

What follows is a test program we use to make sure that the PHP SO is working. The package in which the PHP SO is wrapped is called called 'mhdrg'.

The call to mhdrg() should be self-explanatory. The rules for calling mhdrg() are the same in Perl or PHP, so we refer you to the chapter on the Perl SO for documentation on the mhcall() itself.

mhdrg() takes the usual inputs and gives the usual outputs. For details about how our groupers work, see chapter 2, "DRG Assignment Software." In particular, there is a section on inputs to our grouper and its outputs.

```
<?php
    # tryit.php (c) 2002 M+H Consulting, LLC MH DRG test script
```

```
printf("tryit.php (c) 2002-2015 M+H Consulting, LLC: MH DRG test script");
$ver = "v36";

if (!extension_loaded('phpdrgv36')) {
    if (!dl('phpdrgv36.so')) {      // this should be in a protected direct
        printf("<h1>ERROR</h1><p>Could not load phpdrgv36.so...</p>");
        exit;
    }
}

# call grouper, store results in $retval
$retval = mhdrg1(
    $ver,                   # which DRG version you want
    "/drbd/mnh/src/SWIG",   # path to masks files
    "1",                    # discharge status */
    "25",                   # patient age on admission */
    "2",                    # patient sex (1=male, 2=female) */
    # ICD DX codes */
    "M154   Y",
    # ICD procedure codes */
    "0SRF0J90SRF0KZ",
    8,                      # length of each ICD DX code */
    7,                      # length of each ICD procedure code */
    "Y",                    # are there POA indicators?
    "n"                     # is this case exempt?
);

# if there was an error, alert the user
if ($retval == '') {
    $error = errdesc();
    print("Error: $error<br>");
    exit;
}

# show the raw return value
printf("Raw return value from mhdrg1:<br>%s<br><br>",$retval);

# deconstruct return from mhdrg1()
list($rc,$mdc,$drg,$ovn,$weight,$mean,$porm,$desc) = explode("^",$retval,-1
print("Deconstructed return value from mhdrg1:<br>");
print("rc=$rc, mdc=$mdc, drg=$drg, ovn=$ovn, weight=");
print("$weight, mean=$mean, porm=$porm<br>desc=$desc<br>");
print(" EXPECTED: RC: 0 MDC: 10 DRG: 614<br>");
exit;
```

```
    # This script should produce the following output:
    #-------------------------
    # Raw return value from mhdrg1:
    # 0^15^391^20^0.1517^3.10^M^NORMAL NEWBORN
    #
    # Decontructed return value from mhdrg1:
    # rc=0, mdc=15, drg=391, ovn=20, weight=0.1517, mean=3.10, porm=M
    # desc=NORMAL NEWBORN
    #-------------------------
?>
?>
```

C-Callable Object

The C-Callable object is a programmer's tool. If you want to include this product as part of software that you sell or lease outside your organization, you will need a Reseller's License. Please refer to Appendix D for details.

Since the C-Callable object (CO) is a compiled object, it is platform-specific. You have to buy the CO that was compiled for your specific platform. Currently, we only supply the CO under UNIX and UNIX-derived operating systems. For Win32 environments, we provide the Grouper DLL instead.

We use an outside porting lab named Ready to Run to provide us with UNIX platforms on which to develop and test, so for an up-to-date list of UNIX variants that we support, please see their web site:

`www.rtr.com`

Note: as of version 26, released in 2008, the DRG version can have either a 'p' or an 'e' or both appended to it. If there is a trailing 'p', it is assumed that the last character of every diagnosis code is a POA flag. If the there is a trailing 'e', then the source institution is presumed to be exempt from the HAC. Thus "26p" specifies POA support, while "26" does not. "25p" does not make sense.

Staying Current

In order to stay up-to-date you have to buy a new CO every year which comes with the correct masks file.

Technical Details

The CO is written in ANSI C and should link happily into any executable created with any standard UNIX C or C++ or C# compiler.

Distribution

The CO is distributed as a file to be installed in the appropriate directory with the appropriate permissions.

Calling CO Functions

To call a function in a CO all you have to do is link your executable with our CO and presto! you can call functions within the CO as if the CO were a part of your software.

Sample C Code

What follows is a test program we use to make sure that the CO is working.

The call to mhdrg1() should be self-explanatory. The rules for calling mhdrg1() are the same in Perl or PHP, so we refer you to the chapter on the Perl CO for documentation on the mhcall() itself.

mhdrg1() takes the usual inputs and gives the usual outputs. For details about how our groupers work, see chapter 2, "DRG Assignment Software." In particular, there is a section on inputs to our grouper and its outputs.

```
/* tryit.c (c) 2002 M+H Consulting, LLC C-callable demo (BFH) */
/* HISTORY:
 * Thu Oct 15 08:58:03 EDT 2015 BFH new mhdrg1() API
 * Thu Oct  8 19:22:54 EDT 2015 BFH update for v33
 * Tue Nov 18 11:52:33 EST 2014 BFH test case for huron debugging
 * Sun Oct 14 18:08:14 EDT 2012 BFH test case for v30 bug
 * Sun Oct  3 11:57:46 EDT 2010 BFH support f28
 * 10/06/2008 made win32 and unix versions the same
 * 10/11/2003 added bit-string of used dx's and used procedures
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define DELIM "^"
#define RETURNED 10
```

```
#define DRGVER "v36"

#ifdef WIN32
/* DLL functions */
extern int mhicd();             /* mhicd.dll */
extern char * mhdrg1();         /* mhdrg.dll */
extern char * mherrdesc();      /* mhdrg.dll */
#else
char * mhdrg1();
extern char * errdesc();        /* mhdrg.dll */
#endif

int main() {
    char * retval;
    char * p;
    char * a[RETURNED];
    char * expected;
    int i;

    expected = "DRG: 614        MDC: 10         GRC: 00";

    /* call the grouper, store results in retval */
    retval = mhdrg1(
        DRGVER,                 /* which DRG version you want */
        "/drbd/mnh/src/SWIG",   /* path to masks files */
        "1",                    /* discharge status */
        "55",                   /* patient age on admission */
        "1",                    /* patient sex (1=male, 2=female) */
        /* ICD DX codes */
        "D352   YE2740  NN390   YE871   NF05    NR630   NR110   YR0602  YD649
        /* ICD procedure codes */
        "0GB00ZZ0YUY47Z02HR30Z0L8S3ZZ0P5N0ZZ0RJP0ZZ",
        8,                      /* length of each ICD DX code */
        7,                      /* length of each ICD procedure code */
        "y",                    /* are there POA indicators? */
        "n"                     /* is this case exempt from HAC? */
    );

    /* if there was an error, alert the user */
    if (retval == NULL) {
        printf("M+H grouper argument or environment error: %s0, errdesc());
        exit(-1);
    }
```

```
    /* show the raw return value */
    printf("Raw return value from mhdrg1:68s0,retval);

    /* deconstruct return from mhdrg1() */
    p = strtok(retval,DELIM);
    for (i = 0; i < RETURNED && p != NULL; i++) {
        a[i] = p;
        p = strtok(NULL,DELIM);
    }

    /* last two return values are bit-strings of which ICD codes were
     * used in the grouping
     */

    puts("Deconstructed return value from mhdrg1:");
    printf("rc=%s, mdc=%s, drg=%s, ovn=%s, weight=",a[0],a[1],a[2],a[3]);
#ifdef OLD_WAY
    printf("%s, mean=%s, porm=%s0esc=%s0x flags:%ld, px flags: %ld0,a[4],a[5],a
#else
    printf("%s, mean=%s, porm=%s0esc=%s0x flags:%s, px flags: %s0,a[4],a[5],a[6
#endif

    /* show what we should get */
    printf("0xpected result: %s0,expected);

    /* last two return values are bit-strings of which ICD codes were used in t
    puts("0ignificant ICD codes:");
#ifdef OLD_WAY
    for (mask = 1L,i = 0; i < 32; i++, mask *= 2L) {
        if (mask & dxu) {
            printf(" * DX code %2d was used0,(i+1));
        }
    }
    for (mask = 1L,i = 0; i < 32; i++, mask *= 2L) {
        if (mask & pxu) {
            printf(" * Proc code %2d was used0,(i+1));
        }
    }
#else
    p = a[8];
    for (i = 0; i < 32 && p[i] != ' 00'; i++) {
        if (p[i] == '1') {
            printf(" * DX code %2d was used0,(i+1));
        }
```

```
    }
    p = a[9];
    for (i = 0; i < 32 && p[i] != ' 00'; i++) {
        if (p[i] == '1') {
            printf(" * Proc code %2d was used0,(i+1));
        }
    }
#endif
    exit(0);
}

/* This program should produce the following output:
 -------------------------
Raw return value from mhdrg1:
0^15^391^21^0.1536^3.10^M^NORMAL NEWBORN

Deconstructed return value from mhdrg1:
rc=0, mdc=15, drg=391, ovn=21, weight=0.1536, mean=3.10, porm=M
desc=NORMAL NEWBORN
dx flags:1, px flags: 0

Significant ICD codes:
 * DX code  1 was used
-------------------------
 */
/* eof */
```

Appendix A: New in v37b

We are updating our product line, and this manual, in April, 2020 to mirror CMS's release 1 of version 37.1, which we call "v37b" in our software.

CMS Bulletin

In brief, CMS released an update to handle the vaping diagnosis and the Covid-19 diagnosis. To be slightly less brief, here is CMS's own short bulletin on what they call "37.1 R1":

`www.cms.gov/files/document/icd-10-ms-drgs-version-371-effective-april-1-2020.pdf`

Lowering AIX Support

Since we do not sell enough of them on-line to warrant putting them on-line every year, we are dropping the AIX version of DRGFilt. We are still able to produce the product, so we will be making it an on-demand item for at least the coming year. If you want to purchase the AIX DRGFilt, you will have to send us an email, probably through our tech blog (see below).

Our Tech Blog

As always, for the most recent news on our products, please visit our tech blog:

`drggrouperstechblog.blogspot.com/`

For details on this particular release, see the linked posts on our tech blog; link by the label "v37".

Appendix B: Return Codes

The following is a list of valid return codes for DRGGroupers groupers, as of 2008. This list is provided to give you an idea of what is returned. Please consult our on-line documentation for an up-to-date list.

`drggrouperstechblog.blogspot.com/2015/10/dae-return-codes-error-reporting.html`

1. means EITHER that there were no diagnosis codes given or that the given principal dx code does not have an MDC (ie is not valid as a principal dx).

2. means no DRG could be found to match MDC and principal dx.

3. means that the given age was not between 0-124.

4. means that the given sex was not 1=male or 2=female

5. is no longer used; Discharge Status is defaulted depending on various factors (disposition status coding system, eg)

6. depends on various factors; for example, for some DRG versions this means that birthweight was not between 200 and 9000 grammes.

7. means that the principal dx code was not a valid choice as a primary diagnosis; the code may be valid, but it is not an allowed starting point for assigning a DRG.

8. means that the DRG masks file could not be found or initialized by the run-time environment.

9. means that the DRG to be labelled is too high for the specified DRG version or that there was a HAC violation

10. means that the DRG masks file has internal structural errors.

Appendix C: Discharge Status

The following is a partial list of Discharge Statuses, coded using the UB92 standard.

For a more complete list, please see the CMS web site:

`www.cms.gov/Outreach-and-Education/Medicare-Learning-Network-MLN/MLNMattersArticles/downloads/SE0801.pdf`

01 = Home, Self Care

02 = Short Term Hospital

03 = Skilled Nursing Facility

04 = ICF

05 = Other Facility

06 = Home Health Service

07 = Against Medical Advice

08 = Home IV Service

20 = Expired

30 = Still a patient

Appendix D: Software Licenses

There are the standard licenses DRGGroupers grants through our parent company, M+H Consulting, LLC.

Reseller License

If you want to embed our grouper in a product you will be selling on the market, then you need to negotiate a custom license agreement. Please contact us for details.

Non-standard Licenses

Our licensing is negotiable and we have granted custom license in past. If you have an arrangement you would like us to consider, contact us.

Contact Information

You can find our up-to-date contact information on-line:

`www.drggroupers.com/contact.html`

Standard Client Software License

M+H Consulting, LLC grants to the purchaser a single CPU license for the accompanying software. The software may not be used on more than one computer at the same time. The purchaser shall not make copies of the software except for use as a backup.

The accompanying software is the property of M+H Consulting, LLC and may not be distributed. The software may not be distributed as part of an application or external website without the express, written consent of M+H Consulting, LLC.

Standard Server Software License

M+H Consulting, LLC grants to the purchaser a single CPU license for the accompanying software. The software may not be used on more than one computer at the same time. The purchaser shall not make copies of the software except for use as a backup.

The software may run on only one CPU, "the server," at a time, but no restriction is placed on how many client CPU's can access the server at a time. However, the purchaser may not put the software on a server to which there is network access outside of the purchaser's organization. In this context, "organization" refers to parent companies and their wholly-owned subsidiaries and business units.

The software is the property of M+H Consulting, LLC and may not be distributed. The program may not be distributed as part of an application or external website without the express, written consent of M+H Consulting, LLC.

Glossary

The following is a glossary of terms used in this book which may not be familiar to the reader, especially the programmer who is trying to embed a grouper in their software.

The definitions here are not intended to be complete; instead, they are intended to make our descriptions and directions clearer.

Classification

DRGs are classified as either Medical or Surgical because some conditions have both a surgical and non-surgical treatment path and the expected resource consumption of the two paths are quite different.

CPT Codes

The American Medical Association has defined a scheme for coding procedures, which they call CPT (Current Procedural Terminology). For more information, go on-line:

`www.ama-assn.org/ama/pub/category/3113.html`

DRG

DRG stands for Diagnosis Related Group. Inpatient medical records can be classified into separate groups. Each group is identified by a number. This number is the DRG. Each DRG has various attributes associated with it: the description, the Major Diagnostic Category (MDC), the expected Length Of Stay (LOS), the weight (a measure of how resource-intensive it is), a low trim point and a high trim point.

DRG Grouper

Software that assigns a DRG is called a Grouper. Groupers take coded medical data elements as input and give a DRG as output. The inputs are: age-on-admission, sex, discharge disposition, diagnosis codes (before v33, in ICD9cm; after v32, in ICD10) and procedure codes (before v33, in ICD9cm; after v32, in ICD10).

Up to ten diagnoses are considered and the diagnosis codes are assumed to be in order of significance.

Up to fifteen procedures are considered and the procedure codes are assumed to be in order of significance.

DRG Version

A new definition of DRGs is released every October 1st. Each new definition is a new version, requiring a new release of the software and a new masks file.

US Federal (aka CMS or HCFA) DRG versions are released in October of every year. Version 3 (the first version we support) was released in October of 1985. So if you are using calendar years, 1/1/2001 - 9/30/2001 would be covered by version 18 and 10/1/2001 through 9/30/2002 would be covered by version 19. And so on.

The official release is on October 1st every year and we usually release our implementation of the algorithm by October 15th every year.

Please specify which version you need for any of our products.

For our software, version numbers can be preceeded by an optional letter 'f' for federal (in the ICD9cm-based versions) and by an optional letter 'v' for version (starting with v33).

Federal DRGs

The original DRGs were defined by HCFA (now CMS), a part of the United State federal government. We call these DRG definitions "the federal DRGs" For more information, see

`en.wikipedia.org/wiki/Diagnosis_Related_Group`

HAC (Hospital Acquired Complication)

This is a set of rules which change the DRG assigned for a non-exempt institutation based on any POA flags which are present. This was introduced in US federal DRG assigment with version 26, released in October of 2008.

ICD9cm Codes

The official federal grouper, version 2 through 32 inclusive, only accepts ICD9cm codes (International Committee on Diseases, version 9, Clinical Modifications) for both diagnoses and procedures. There are two lists of codes: a list of diagnoses and a separate list of procedures. Procedures were originally often called "surgeries". The ICD9cm codes are defined by organ system and have a major and minor component, often written

major.minor

ICD10 Codes

The official federal grouper, starting with v33, only accepts ICD10 codes (International Committee on Diseases, version 10) for both diagnoses and procedures. There are two lists of codes: a list of diagnoses and a separate list of procedures. Procedures were originally often called "surgeries". The ICD10 codes of either kind are alpha-numeric and have a maximum length of seven characters.

Major Diagnostic Category (MDC)

The MDC is a kind of pre-DRG, a looser category from which the appropriate DRG is chosen. Some analyses are based on this looser category in order to provide a higher-level result. See Appendix A for a listing of MDCs as they were in 2001.

Masks File

The canonical US government grouper, which we call "the federal grouper", applies logic and masks to its input to determine the DRG. The masks are code-specific bit masks representing various conditions. Thus, in order to assign a DRG you need both grouper software, which contains the logic, and the appropriate masks file, which contains the masks.

For ICD9cm-based versions (2-32), we expect masks files to be named

```
drgmasks.fN
```

where N is the version to which they correspond. So the masks file name for version 8 would be

```
drgmasks.f8
```

and the masks file name for version 21 would be

```
drgmasks.f21
```

For ICD10-based versions (33 and above), we expect masks files to be named

```
drgmasks.vN
```

where N is the version to which they correspond. So the masks file name for version 33 would be

```
drgmasks.v33
```

POA (Present on Admission)

In order to avoid paying for medical mistakes, diagnoses are now flagged as having been present on admission (POA).

The values for the POA flag are not, as one might expect, simply Y or N; rather the following options are defined:

- Y for Yes -N for No -U for Unspecified -W for clinically undetermined -1 for unreported / not used / exempt from reporting

Return Code

Our groupers always return a code which indicate success (a value of zero) or some kind of failure (a non-zero value). For details on the possible kinds of failure, you can refer to the list in Appendix B, or our tech support page on our web site:

Significant Code Bit String (dflg and sflg)

Not all the codes, either diagnosis or procedure, are significant in determining the DRG. Our groupers will tell you which codes were used in the DRG assignement and which were not.

This information is conveyed by a string of either ones or zeros. For historical reasons, this string of characters is referred to as "the bit string."

There are ten input Diagnosis codes, and so there are ten digits in the Diagnosis bit string. There are fifteen input Procedure codes, so there are fifteen digits in the Procedure bit string.

If a given digit is zero, then the corresponding code was not used; if a given digit is one, then the corresponding code was used.

`drggrouperstechblog.blogspot.com/2011/08/question-hi-tech-support-pdf-is-very.html`

Index

A
AP-DRGs 12
API 37, 42, 46, 66

C
Comma Separated Value file (see CSV)
Control File 27, 28, 32
CSV 27, 28, 32, 33, 34

D
Discharge Disposition (see Discharge Status)
Discharge Status 48, 71, 72
DRG Properties 15
DRGFilt Control File (see Control File)

E
Exempt 18, 39, 49

F
Federal DRGs 12, 77

H
HAC 15, 16, 18, 37, 42, 44, 48, 49, 54, 57, 61, 65, 67, 71, 77
Hospital Acquired Complication (see HAC)

M

M & H Consulting (see M+H Consulting)
M and H Consulting (see M+H Consulting)
M+H Consulting 17, 53, 58, 62, 63, 66, 73, 74
Major Diagnostic Category (see MDC)
mhdrg.dll 44, 46, 67
mhicd.dll 46, 54, 67
mhicdvb.dll 45

P

POA 15, 16, 18, 37, 39, 42, 44, 48, 49, 51, 54, 57, 59, 61, 63, 65, 67, 77, 79
Present On Admission (see POA)

R

RDRGs 12, 13

S

SDRGs 13
SWIG 54, 57, 59, 61, 63, 67

U

UHDDS 14, 18