User Documentation for the RUG-III Version 5.20 DLL

Iowa Foundation for Medical Care

August 9, 2005

This is user documentation for the DLL provided in the RUG-III Version 5.20 Grouper package.

Disclaimer

To the maximum extent permitted by applicable law, CMS and its contractors/distributors make no representations about the suitability, for any purpose or use, of the software and documents. THE SOFTWARE AND DOCUMENTS ARE PROVIDED "AS IS" AND WITHOUT WARRANTY OF ANY KIND. ANY WARRANTIES, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR USE, TITLE, AND NON-INFRINGEMENT ARE HEREBY DISCLAIMED. Recipient shall not, in any event, be entitled to, and CMS or contractors/distributors shall not be liable for indirect, special, incidental, or consequential damages of any nature including, without being limited to, loss of use, data, or profit irrespective of the nature of the claim, whether in contract, tort, warranty, or otherwise.

Introduction

The DLL provided with the RUG-III Version 5.20 Grouper package is a 32-bit DLL based on C++ code and is named ***RUG520.DLL***. The DLL contains an integer function[1] called RugCalc() that performs RUG-III classification for the 34-group, the 44-group, and the 53-group model.

Return Value. The RugCalc( ) function return value is 0 if there are no MDS RUG-III items with out of range errors. If out of range errors occur, then the function return value is 1.

Each call to the function returns both a hierarchical RUG-III classification and an index maximized classification. The input and output parameters for this function are described in detail in the document "Grouper Doc.pdf" and are repeated in Table 1 below. This DLL differs from the prior DLL for RUG-III Version 5.12 in the following ways:

- The Version 5.20 DLL is written in C++ while the Version 5.12 DLL was written in Visual Basic. The new 5.20 DLL has been developed in C++ to allow use with a wider range of platforms and has been tested with C++, Visual Basic 6 and Visual Basic .NET.
- Unlike, the Version 5.12 DLL, the new 5.20 version does not need to be registered under Windows (in fact, it ***cannot*** be registered with Windows). Your calling program should indicate the path in which the DLL is located (the default path is the application directory – no path need be indicated if the DLL is located in the application directory).

[1] The data type of the return value is Integer in C++ and Visual Basic .NET programs calling the Grouper and Long in Visual Basic 6 programs calling the Grouper.

- The prior Version 5.12 classification module was an object class, while the new Version 5.20 classification module is a function.

The C++ source code for the 5.20 DLL is provided in the grouper package in a file named ***RUG520.CPP***. This source code uses variables with the standard parameter names in Table 1 below. This source code is provided to enable users to better understand RUG-III Version 5.20 classification and as an aid in developing code in other languages.

Standard Parameters

Table 1 describes the standard parameters for the RugCalc( ) function.

Table 1. Standard Input and Output Parameters for the 5.20 DLL

Parameter Name	Parameter Type	Data Type	Description
sMdsRecord	Input	String	A string containing the MDS record in standard format (length=1812 bytes).
sRehabType	Input	String	A string containing one of the following two values designating the type of rehab classification to perform: • "Mcare" = special Medicare rehabilitation classification (using ordered therapies in Section T on 5-day and readmission/return assessments) • "Other" = rehabilitation classification does not use ordered therapies in Section T Upper, lower, or mixed case is acceptable when supplying this parameter.
sModel	Input	String	A string containing one of the following three values designating the RUG-III model to use: • "53" = 53-group model • "44" = 44-group model • "34" = 34-group model
iQuarterlyFlag	Input	Integer or Long[1]	The quarterly flag is an integer value indicating whether RUG-III is to be calculated for quarterly assessments. 0 = RUG-III not calculated for quarterlies 1 = RUG-III is calculated for quarterlies The grouper will automatically determine whether the MDS record corresponds to a quarterly assessment.

[1] The data type of this parameter is Integer in C++, Visual Basic .NET, and SAS programs calling the Grouper and Long in Visual Basic 6 programs calling the Grouper.

Table 1. Standard Input and Output Parameters for the 5.20 DLL

Parameter Name	Parameter Type	Data Type	Description
nCmiArray	Input	Double precision array	nCmiArray is a double-precision array containing case mix indices (CMIs). This array affects the index maximized RUG-III classification returned by the grouper. The array must be supplied even if you are not interested in the index maximized RUG-III classification. If you are not interested in index maximized results, you can initialize the array with zeroes and the index maximized RUG-III will equal the hierarchical RUG-III classification. For C++ and Visual Basic programs calling the grouper DLL, the array must be dimensioned from 0 to 58. Put the 58 CMI indices in elements 1 to 58 (element 0 can be initialized to 0.0). For SAS programs using the SAS grouper module, the array should be dimensioned with 58 elements and there is no "0" element. The RUG-III groups corresponding to the 58 elements in nCmiArray are listed in Section 4 of this document. The standard CMI sets available with Version 5.20 are available in Section 9 of "Grouper Doc.pdf".
sRugHier	Output	String	The grouper will return a 3-byte character code (e.g., "RUC") for the hierarchical RUG-III group. When you call the grouper, you should initialize this variable to a blank string. This code will be the default RUG-III group (BC1) if any of the 108 RUG-III MDS items are out of range.
sRugMax	Output	String	The grouper will return a 3-byte character code (e.g., "RUC") for the index maximized RUG-III group. When you call the grouper, you should initialize this variable to a blank string. This code will be the default RUG-III group (BC1) if any of the 108 RUG-III MDS items are out of range.
nRugHier	Output	Integer or Long[1]	The grouper will return the numeric code for the hierarchical RUG-III group. This is the numeric position of the group in the standard group order in Section 4. When you call the grouper, you should initialize this variable to a 0 value. Such numeric values have proved useful to researchers studying RUG-III. This numeric code will be that for the default RUG-III group (58) if any of the 108 RUG-IIIMDS items are out of range.

[1] The data type of this parameter is Integer in C++, Visual Basic .NET, and SAS programs calling the Grouper and Long in Visual Basic 6 programs calling the Grouper.

Table 1. Standard Input and Output Parameters for the 5.20 DLL

Parameter Name	Parameter Type	Data Type	Description
nRugMax	Output	Integer or Long[1]	The grouper will return the numeric code for the index maximized RUG-III group. This is the numeric position of the group in the standard group order in Section 4. When you call the grouper, you should initialize this variable to a 0 value. Such numeric values have proved useful to researchers studying RUG-III. This numeric code will be that for the default RUG-III group (58) if any of the 108 RUG-III MDS items are out of range.
nCmiValueHier	Output	Double precision	nCmiValueHier is a double-precision variable which on return will contain the case mix index value (from nCmiArray) corresponding to the resulting hierarchical RUG-III value. When you call the grouper, you should initialize this variable to a 0.0 value.
nCmiValueMax	Output	Double precision	nCmiValueMax is a double-precision variable which on return will contain the case mix index value (from nCmiArray) corresponding to the resulting index maximized RUG-III value. When you call the grouper, you should initialize this variable to a 0.0 value.
iAdlSum	Output	Integer or Long[1]	iAdlSum is an integer variable which on return will contain the RUG-III ADL scale score (range of 4 to 18). When you call the grouper, you should initialize this variable to a 0 value.
iCpsCode	Output	Integer or Long[1]	iCpsCode is an integer variable which on return will contain the Cognitive Performance Scale (CPS) score (range of 0 to 6). When you call the grouper, you should initialize this variable to a 0 value.
sRugsVersion	Output	String	sRugsVersion is a string variable which returns the version of the RUG-III classification logic which was used. This will be a value of “07” for the 44-group model, “08” for the 34-group model, and “09” for the 53-group model. When you call the grouper, you should initialize this variable to a blank string.

[1] The data type of this parameter is Integer in C++, Visual Basic .NET, and SAS programs calling the Grouper and Long in Visual Basic 6 programs calling the Grouper.

Table 1. Standard Input and Output Parameters for the 5.20 DLL

Parameter Name	Parameter Type	Data Type	Description
sDllVersion	Output	String	sDllVersion is a string which returns a value of "1.00", the version number (within RUG-III Grouper Version 5.20) for the present DLL or SAS grouper module. When you call the grouper, you should initialize this variable to a blank string.
iError	Output	Integer or Long[1]	The grouper will return an integer error code. The error code will have one of the following values: 0 = No grouper calling error, RUG-III was calculated or set to the default value if out-of-range values were found for any RUG-III item 1 = sRehabType parameter was invalid 2 = sModel parameter was invalid 3 = iQuarterlyFlag parameter was invalid 4 = RUG-III not calculated for this type of MDS record. This code will always be returned for an MDS batch header record, MDS batch trailer record, discharge record, reentry record, or inactivation record. It will also be returned for a quarterly assessment if the user has indicated that RUG-III is not to be calculated for quarterly assessments (iQuarterlyFlag = 0). Finally, this error code will also be returned if the reasons for assessment codes (MDS items AA8a and AA8b) are invalid (e.g., out of range). 5 = Invalid CMI value used. No CMI value may be less than or equal to –9999.

The RugCalc( ) function must be called with the parameters corresponding to their position and type in Table 1. Details about how to declare and call the function from C++, VB6, and VB .NET are presented later in this document.

Case Mix Indices

Table 2 describes the elements of the nCmiArray standard input parameter. The file "Cmi520.xls" presents the standard CMI sets available with Version 5.20 in a format that will allow ease of coding.

[1] The data type of this parameter is Integer in C++, Visual Basic .NET, and SAS programs calling the Grouper and Long in Visual Basic 6 programs calling the Grouper.

Table 2. Elements in nCmiArray[ ]

CMI Array Element	Corresponding RUG-III Group
0	Required dummy element--set to 0
Rehabilitation/Extensive Groups for 53-group model	
1	RUX: Rehabilitation Ultra High Plus Extensive / ADL 16-18
2	RUL: Rehabilitation Ultra High Plus Extensive / ADL 7 – 15
3	RVX: Rehabilitation Very High Plus Extensive / ADL 16 – 18
4	RVL: Rehabilitation Very High Plus Extensive / ADL 7 - 15
5	RHX: Rehabilitation High Plus Extensive / ADL 13 - 18
6	RHL: Rehabilitation High Plus Extensive / ADL 7 – 12
7	RMX: Rehabilitation Medium Plus Extensive / ADL 15 – 18
8	RML: Rehabilitation Medium Plus Extensive / ADL 7 - 14
9	RLX: Rehabilitation Low Plus Extensive / ADL 7 - 18
44-Group Rehabilitation Groups for the 53-group and 44-group models	
10	RUC: Rehabilitation Ultra High / ADL 16 – 18
11	RUB: Rehabilitation Ultra High / ADL 9 – 15
12	RUA: Rehabilitation Ultra High / ADL 4 - 8
13	RVC: Rehabilitation Very High / ADL 16 – 18
14	RVB: Rehabilitation Very High / ADL 9 – 15
15	RVA: Rehabilitation Very High / ADL 4 - 8
16	RHC: Rehabilitation High / ADL 13 – 18
17	RHB: Rehabilitation High / ADL 8 – 12
18	RHA: Rehabilitation High / ADL 4 - 7
19	RMC: Rehabilitation Medium / ADL 15 – 18
20	RMB: Rehabilitation Medium / ADL 8 – 14
21	RMA: Rehabilitation Medium / ADL 4 - 7
22	RLB: Rehabilitation Low / ADL 14 – 18
23	RLA: Rehabilitation Low / ADL 4 – 13
Extensive Groups for all models (53-, 44-, and 34-groups)	

Table 2. Elements in nCmiArray[ ]

CMI Array Element	Corresponding RUG-III Group
24	SE3: Extensive Services 3 / ADL > 6
25	SE2: Extensive Services 2 / ADL > 6
26	SE1: Extensive Services 1 / ADL > 6
Rehabilitation Groups for the 34-group model	
27	RAD: Rehabilitation All Levels / ADL 17 - 18
28	RAC: Rehabilitation All Levels / ADL 14 – 16
29	RAB: Rehabilitation All Levels / ADL 9 - 13
30	RAA: Rehabilitation All Levels / ADL 4 - 8
Remaining Groups for all models (53-, 44-, and 34-groups)	
31	SSC: Special Care / ADL 17 – 18
32	SSB: Special Care / ADL 15 – 16
33	SSA: Special Care / ADL 4 – 14
34	CC2: Clinically Complex with Depression / ADL 17 - 18
35	CC1: Clinically Complex / ADL 17 – 18
36	CB2: Clinically Complex with Depression / ADL 12 - 16
37	CB1: Clinically Complex / ADL 12 – 16
38	CA2: Clinically Complex with Depression / ADL 4 - 11
39	CA1: Clinically Complex / ADL 4 – 11
40	IB2: Cog. Impairment with Nursing Rehab / ADL 6 - 10
41	IB1: Cognitive Impairment / ADL 6 – 10
42	IA2: Cog. Impairment with Nursing Rehab / ADL 4 - 5
43	IA1: Cognitive Impairment / ADL 4 - 5
44	BB2: Behavior Problem with Nursing Rehab / ADL 6 - 10
45	BB1: Behavior Problem / ADL 6 – 10
46	BA2: Behavior Problem with Nursing Rehab / ADL 4 - 5
47	BA1: Behavior Problem / ADL 4 - 5
48	PE2: Physical Function with Nursing Rehab / ADL 16 - 18

Table 2. Elements in nCmiArray[ ]

CMI Array Element	Corresponding RUG-III Group
49	PE1: Physical Function / ADL 16 – 18
50	PD2: Physical Function with Nursing Rehab / ADL 11 - 15
51	PD1: Physical Function / ADL 11 – 15
52	PC2: Physical Function with Nursing Rehab / ADL 9 - 10
53	PC1: Physical Function / ADL 9 – 10
54	PB2: Physical Function with Nursing Rehab / ADL 6 - 8
55	PB1: Physical Function / ADL 6 - 8
56	PA2: Physical Function with Nursing Rehab / ADL 4 - 5
57	PA1: Physical Function / ADL 4 - 5
58	BC1: RUG-III group not calculated due to data errors

Calling the DLL from C++

In the 5.20 Grouper package, there are two example C++ programs that call the 5.20 version DLL to perform RUG-III classification. The first example program is in ***Demo520_CPP1.ZIP*** with the C++ source code calling the DLL named ***Demo1.CPP*** and the corresponding executable code named ***Demo1.exe***. The second example program is in ***Demo520_CPP2.ZIP*** with the C++ source code calling the DLL named ***Demo2.CPP*** and the corresponding executable code named ***Demo2.exe***. The two programs work exactly the same, the only difference being in the method of accessing the DLL. ***Demo1.CPP*** links to a special RUG520.lib that has been created with the RUG520.H header file. ***Demo2.CPP*** uses LoadLibrary("RUG520.DLL") to access the DLL. Both programs use the test data file ***Test520a_max.txt***. To execute either program, place the executable, the DLL (***RUG520.DLL)*** and this test data file in the same directory and double click the executable.

Both programs use the DLL to perform 34-group RUG-III classification with the standard D01 CMI set and with non-Medicare rehabilitation classification (ordered therapies are ignored). Each program reads each record in one of the standard test files (***Test520a_max.txt***) and calls the RugCalc( ) function in the DLL. If there is a problem (record is inappropriate for RUG-III classification or out of range value detected for any MDS RUG-III item), then information concerning the record is written to an error file (ErrorRecords.txt). If there are no problems for a record, the program writes the calculated 34-group RUG-III hierarchical code, CMI associated with that code, ADL sum, CPS code, and RUG-III version code for that record, along with the corresponding known values present in the data record (see ***Test520a_max Doc.pdf***).

When either program is executed with this test data set, 108 error record entries should be produced as follows:

- Record # 702 is an inappropriate record for classification (iError = 4), since this record is an "unknown" type of record with the AA8b reason for assessment item having a bad value of 9.
- Records 703 through 750 each have an out of range MDS RUG-III item.
- Record 752 through 810 each have an out of range MDS RUG-III item.

When either program is executed, the output file should list all other records and the calculated 34-group hierarchical results should match the known values in the record.

Features of the C++ code to note are:

- The CMI array is dimensioned with 59 elements. The 0 element is ignored in the RUG-III classification logic. Only elements 1 through 58 are used.
- The return value for the RugCalc( ) function is an integer value equaling 0 if no RUG MDS items were found to be out of range and equaling 1 if any RUG MDS item was out of range.

Calling the DLL from Visual Basic 6

The Grouper package includes a VB6 demo program in the ZIP file named ***Demo520_vb6.ZIP***. The source code calling the DLL to perform RUG-III classification is named ***Demo520_vb6.frm*** and the executable is ***Demo520_vb6.exe***. This demo program processes all of the records in the test data set ***Test520a_max.txt*** and performs 53-group RUG-III classification. To execute the demo program, place the executable, the DLL (***RUG520.DLL)***, and this test data file in the same directory and double click the executable. For each record, the program calls the DLL to compute RUG-III and it reads expected values that are stored in the test record. It then writes an output file called ***Demo520_vb6.txt*** which lists the calculated and expected values for each record. When the program is finished, it will display a message which should state that 1004 records were processed and that no errors were encountered. An error would occur if the calculated RUG-III group (either index hierarchical or index maximized) did not match the expected values.

Features of the VB6 code to note are:

- The CMI array is dimensioned with 59 elements. The 0 element is ignored in the RUG-III classification logic. Only elements 1 through 58 are used.
- The return value for the RugCalc( ) function is an integer value equaling 0 if no RUG MDS items were found to be out of range and equaling 1 if any RUG MDS item was out of range.

Special care must be taken in declaring and using the DLL from VB6. Three points should be noted:

- In the function declaration, some parameters are passed ByVal while others are passed ByRef. These references must be followed ***exactly*** as in ***Demo520_vb6.frm*** or program errors may occur.
- Note that the function declaration contains an ALIAS clause. This clause is required.
- All parameters (including the return value of the function) that are treated as integer in C++ ***must*** be declared as ***long*** in VB6 (exactly as shown in ***Demo520_vb6.frm)***. Again, failure to follow this guideline may result in program errors.

If you wish to call the DLL from VB6, please study the demo program carefully and follow it closely. The methods displayed for declaring and calling the RugCalc() function must be followed exactly to insure that your program works correctly.

Calling the DLL from Visual Basic .NET

In the 5.20 Grouper package, there is an example VB .NET program in the ZIP file named ***Demo520_NET.ZIP***. The source code that calls the 5.20 version DLL to perform RUG-III classification is named ***Module1.vb*** and the executable is ***ConsoleApplication1.exe***. This program passes an example MDS record to the RugCalc() function and displays selected results calculated for the 34-group model.

Features of the VB .NET code to note are:

- The CMI array is dimensioned with 59 elements. The 0 element is ignored in the RUG-III classification logic. Only elements 1 through 58 are used.
- The return value for the RugCalc( ) function is an integer value equaling 0 if no RUG MDS items were found to be out of range and equaling 1 if any RUG MDS item was out of range.

Special care must be taken in declaring and using the DLL from VB .NET. Three points should be noted:

- In the function declaration, some parameters are passed ByVal while others are passed ByRef. These references must be followed ***exactly as in Module1.vb*** or program errors may occur.
- Note that the function declaration contains an ALIAS clause. This clause is required.
- As noted above, all parameters (including the return value of the function) that are treated as integer in C++ must be declared as long in VB6. However, this ***does not apply to VB .NET***. ***Integer parameters should be declared as integers in your VB .NET program (exactly as shown in Module1.vb).***

If you wish to call the DLL from VB .NET, please study the demo program carefully and follow it closely. The methods displayed for declaring and calling the RugCalc() function must be followed exactly to insure that your program works correctly.

