FCG Prodacapo DRG Server 3.1

User's Manual

Document version 9

Contents

Introduction

FCG Prodacapo DRG Server

FCG Prodacapo DRG Server enables IT systems to access DRG grouper functionality using TCP/IP sockets. A centralized, single DRG Server can serve all patient information systems in a hospital (or larger area), and is easy to maintain and update.

While using standard interfaces to exchange patient information between utilizing systems and DRG Server, it is easy to connect new systems to DRG Server regardless of their operating system.

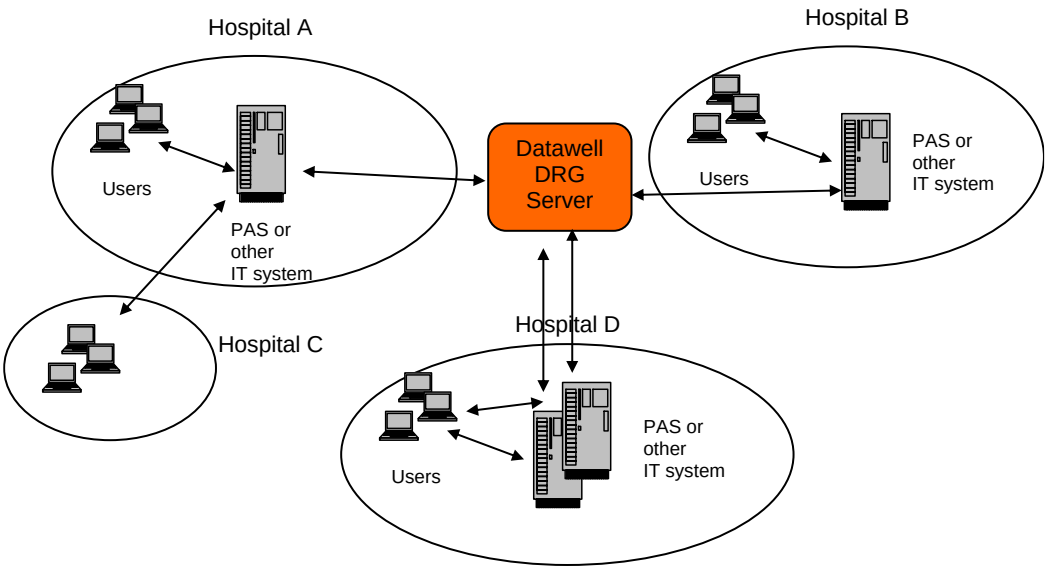

Figure 1. Example of FCG Prodacapo DRG Server implementation

FCG Prodacapo DRG Server extension, FCG Prodacapo *DRG Grouper Service*, includes web service interface for easy integration using SOA tools. The installation of the web service interface is described in a separate document. This document contains the instructions to install the core DRG Server providing a direct TCP/IP socket interface.

GDPR Compliance

FCG Prodacapo DRG Server doesn’t contain and store personal information, see appendix A.

About FCG Prodacapo Groupers

FCG Prodacapo DRG groupers are based on YAGG technology, which is owned, developed and maintained by FCG Prodacapo. FCG Prodacapo grouper products are designed to run different DRG specifications in Windows and Linux operating systems. Different versions use the same grouper core, YAGG Engine, which ensures the same grouping results over different platforms and interfaces.

The grouping specification is separated from the grouper program, which makes DRG specification updates easy. Usually, annual updates of the grouper specification do not require reinstallation or update to the grouper program itself, just installing the new specification is required.

FCG Prodacapo DRG Server

Overview

DRG Server

DRG Server runs as a Windows service or a Linux background process. When started, DRG Server reads the current configuration. The configuration defines which grouper instances to start (e.g. DRG v2015 and DRG v2016), and which TCP ports are attached to these versions. Client software connecting to a version port directly always uses this version regardless of date information in the patient record. DRG Server also opens a "multi-version" TCP port. Patient records sent to this port are assigned to a grouper instance based on the discharge date in the patient record or the version identifier directly if provided with the patient data. For instance, patient discharged 2015-12-20 is assigned to grouper "DRG v2015" and patient discharged 2016-01-02 is assigned to grouper "DRG v2016". See *Configuring the system* for more details.

DRG Server also exchanges information with grouper instances and external commands using TCP sockets. These ports are defined in the configuration file, but default values need not usually be changed.

DRG Server writes notes and errors to the main log file. This log file *drgsrv.log* should be monitored for system errors.

Grouper instances

Grouper instances get grouping requests from DRG Server. Every instance is bound to a specific grouping definition, for example DRG v2014 and DRG v2015 run in different grouper instances. They all have separate log files which should be checked, if there are problems when starting the grouper instance.

Starting and stopping the system

In Windows operating systems, you can use *net start DRGServer* and *net stop DRGServer,* or Windows Control Panel/Services to start/stop DRG Server. In Linux systems, start the process *drgsrv* in background (&) from the Linux shell and stop it using the *kill* command.

Requirements

There are no special requirements for the hardware; any modern, standard computer can run FCG Prodacapo DRG Server.

Installation and configuration

Installation

Windows

A complete FCG Prodacapo DRG Server consists of the grouper software and DRG specifications. To install FCG Prodacapo DRG Server, right-click *setup.exe* and *Run as administrator*. Follow the instructions. The installation program will create a directory for the software, copy the program and register DRG Server as a Windows service.

NOTE: The default installation directory is C:\DRGServer (not *Program Files*).

After successfully running the installation program, FCG Prodacapo DRG Server must be configured before use and DRG specifications must be copied to the system.

Linux

To install FCG Prodacapo DRG Server in Linux environment, extract the file *drgserver.tar.Z* to appropriate location (for example to the directory: /opt/drgsrv), then add environment variable DRGSERVER_HOME to contain the installation path. In the above example the variable would look like this: DRGSERVER_HOME=/opt/drgsrv.

Note that FCG Prodacapo DRG Server must be configured before use and DRG specifications must be copied to the system.

Configuring the system

Configuration file drgsrv.ini

File *drgsrv.ini* includes information about DRG versions used, interfaces for the client software, and technical parameters (such as timeouts). Pre-configured file is provided within the installation, but some parameters may also need manual adjusting.

File *drgsrv.ini* consists of sections (e.g. `[CLIENT]`) and parameter-value pairs (e.g. `MAX_CLIENTS=200`). The file is read during DRG Server startup, so DRG Server must be rebooted after changes to the file.

Section *MULTIVERSION* is used, when a single TCP socket is used by clients to contact several DRG versions, and dynamic DRG version allocation is performed by the DRG Server. The dynamic allocation of the patient data to DRG version is based on 1) validity dates of DRG versions and the date field in the patient data, or 2) version identifier in the patient data matching the version identifier defined in file *drgsrv.ini*. When only one DRG version is used, or several DRG versions are used, but they are contacted directly using the TCP ports defined in the *VERSIONS* section, *MULTIVERSION* section may be omitted.

NOTE: The installation kit contains sample files *drgsrv_single.ini* and *drgsrv_multi.ini*. If you need only one grouper instance at a time to run, you should rename *drgsrv_single.ini* as *drgsrv.ini*. If you need several DRG versions to run simultaneously, rename *drgsrv_multi.ini* to *drgsrv.ini*.

The following table contains the parameters in *drgsrv.ini*. Parameters with default values may be omitted in the file, others are obligatory.

Section	Parameter	Description	Example Value	Def-ault
CLIENT	TIMEOUT	Timeout in seconds while reading data from client.	10	5
CLIENT	IDLE_TIMEOUT	Disconnect client, if idle for more than IDLE_TIMEOUT seconds (0= never disconnect).	3600	0
CLIENT	MAX_CLIENTS	Maximum number of simultaneous client connections.	100	-
MULTIVERSION	DEFAULT_VERSION	Version to be used if the date value in the dataset doesn't match to any grouper validity range or if the date is missing.	SCA2015PR1F	-
MULTIVERSION	CLIENT_PORT	Port number to connect to select the grouper version using patient discharge date / version id.	7180	-
MULTIVERSION	RECORD_FORMAT	Format of the dataset to be grouped: FIXED or FREE. FIXED format dataset contains fields in fixed positions; FREE format dataset contains fields separated by separator characters.	FREE	-
MULTIVERSION	FIELD_SEPARATOR	Only used in FREE format: character separating fields.	,	-
MULTIVERSION	FIELD_SURROUNDER	Only used in FREE format: character at the beginning and at the end of fields.	"	-
MULTIVERSION	VERSIONID_POSITION	Column position of version information affecting the selection of the grouper version. FREE format: (serial) number of the column; FIXED format: column position in the dataset.		
MULTIVERSION	VERSIONID_LENGTH	ONLY USED WHEN RECORD_ FORMAT=FIXED. The length of version identifier.		
MULTIVERSION	DATE_POSITION	ONLY USED WHEN VERSIONID_POSITION is not present. Column position of date information affecting the selection of the grouper version. FREE format: (serial) number of the column; FIXED format: column position in the dataset.	1	-

MULTIVERSION	DATE_FORMAT	ONLY USED WHEN VERSIONID_POSITION is not present. Following formats are allowed: YMD8, DMY8, YMD10, DMY10 and Y4 (examples are in this order).	20140514, 14052014, 2014-05-14, 14.05.2014, 2014	-
VERSIONS	VERSION1	Version identification of the grouper specification 1.	SCA2015PR1F	-
VERSIONS	FILENAME1	Definition file name of the grouper specification 1.	ndrgbase.gg	-
VERSIONS	CLIENT_PORT1	Client port of the grouper specification 1. If this port is used by the client, grouper version is not selected dynamically using date information. Instead, this version is always used.	7174	-
VERSIONS	VALIDITY1	ONLY USED WHEN DATE_POSITION IN MULTIVERSION IS PRESENT. Beginning and expiration dates of the grouper specification 1 (must be formatted YYYYMMDD-YYYYMMDD).	20150101-20151231	-
VERSIONS	VERSION*N*	Version identification of the grouper specification *N*.		
VERSIONS	FILENAME*N*	Definition file name of the grouper specification *N*.		
VERSIONS	CLIENT_PORT*N*	Client port of the grouper specification *N*.		
VERSIONS	VALIDITY*N*	ONLY USED WHEN DATE_POSITION IN MULTIVERSION IS PRESENT. Beginning and expiration dates of the grouper specification *N*.		
COMMAND	TIMEOUT	Command socket timeout in seconds	10	5
COMMAND	PORT	Port number, where commands are connecting	7172	-
GROUPER	TIMEOUT	Grouper instance timeout in seconds (how long the Controller is waiting for the grouping results)	10	5
GROUPER	PORT	Port number where grouper instances are connecting.	7173	-

Adding new grouper versions

If you are running a single DRG version at a time, following steps are needed to update or install the initial DRG specification:

- Copy the DRG specification file to the subdirectory *singleversion*, which is located in the installation directory of DRG Server.
- When installing at the first time: Open the input/output definition file (usually named *ndrgbase.gg*). Check that the input and output formats and the coding of patient sex and discharge codes are compatible with your patient data. See *Grouper definition files* for details.

If you are running multiple DRG versions simultaneously, following steps are needed to update or install the initial DRG specification:

- Create a new grouper definition directory named after the grouper *version id* under DRG Server installation directory. The grouper version id is delivered with the grouper specification file.
  (e.g., create directory C:\DRGServer\SWE2015PR1F). If previous grouper versions (directories) exist, you can have a look/copy the directory as a template for the new version. Check that the input and output formats and the coding of patient sex and discharge codes are compatible with

your patient data. See *Grouper definition files* for details.

NOTE: Since version 3.1.0, the installation directory name of the grouper version and the version id of the grouper may differ. For instance, you may install grouper version *SWE2015PR1F* to subdirectory named S*WE2015*. This makes it easier to update the grouper, if new grouper versions ("bug fixes") are released during the year.

- Modify *drgsrv.ini* file. New version is added by adding following lines to [VERSIONS]:

  VERSION3=<*the name of the version folder, for example SWE2010PR1F*>
  CLIENT_PORT3=<*port number, for example 7183*>
  FILENAME3=<*name of the grouper definition, for example ndrgbase.gg*>
  VALIDITY3= <*date range that version is valid, for example 20140101-20141231*>

  In this example version number **3** is used, replace it with the appropriate value (VERSION**3**, CLIENT_PORT**3**, FILENAME**3**).

  The FILENAME parameter refers to a file that specifies the input/output interface for the grouper. The file also includes a reference to the grouper definition file.

  The VALIDITY parameter is only needed, if dynamic version allocation using date in the patient data is used. See *Configuration file drgsrv.ini* for details.

  If you want to define this newly added version as default, update DEFAULT_VERSION parameter under [MULTIVERSION].

  For more information about the grouper definition files and configuration, have a look at sections *Configuring the system* and *Grouper definition files*.

For both single version and multiversion installations, the service must be (re)started before the changes take effect:

- If DRG Server is running, stop the service. In Windows, stop service DRGServer using Control Panel/Services. In Linux, use *kill <DRG Server ProcessId>* to stop the service.
- Start DRG Server. In Windows, use Control Panel/Services, in Linux, start DRG Server as a background process.

Testing the system

After installing FCG Prodacapo DRG Server you can test the installation locally with the *drgclient* included in the installation kit. See file *drgclient_readme.txt* for details.

Maintenance

Directories and files

The main installation directory (e.g. *C.\Program Files\DRGServer*) contains the following files:

Windows	**Linux**	
drgsrv.exe	drgsrv	DRG Server executable
drgsrv.ini	drgsrv.ini	configuration file
drgsrv.log	drgsrv.log	main log file
yagg.exe	yagg	grouper engine executable, used to start grouper instances

Grouper versions are in subdirectories named by their grouper ID, (e.g. *C.\DRGServer\DRG2015 in Windows or /opt/DRGServer/DRG2015 in Linux*).

Exact file names are dependent on the grouper specification. Log file for the grouper version is named *yagg.log*. This file should be checked if there are problems starting the grouper instance, or if the grouping fails for some reason.

Starting and stopping

Windows

Windows Services can be managed by *Control Panel*'s *Services* application. After installation, DRGServer is listed in the service list, and can be started and stopped with *Start* and *Stop* buttons.

By double clicking DRG Server in the list, you get a property dialog, where you can control the service more precisely. In normal use, you probably want to start the service automatically in system startup. This can be set by choosing *Automatic* for the *Startup type*.

It is also possible to start DRG Server by starting the Windows command prompt and entering: *net start DRGServer*. DRG Server is stopped by entering: *net stop DRGServer.*

Linux

In Linux environment DRGServer is started by executing following command: *drgsrv &*. DRGServer can be stopped by executing command *kill* <PID>. PID for DRGServer can be found using *ps* command.

Daily maintenance

Event log

Windows version of DRG Server uses Windows Event Log to output messages related to installing, starting and stopping the service. You can see these messages by starting Windows' *Event Viewer*. Log Messages are stored in the *Application Log*, which can be chosen in the *Log* menu.

Linux versions use *stderr* for these errors.

After the service has been successfully started, log messages are written to the main log file *drgsrv.log*.

Log messages

The main log file *drgsrv.log* should be monitored to find errors in the system. Messages in the log are started with date and time, followed by:

- DEBUG: debug-only messages
- ERROR: error messages which need attention
- WARNING: not fatal to the system, but may be indicators of other problems
- empty space in normal status messages

Examples of different messages:

```
20141218 114339:DEBUG   Grouper connection accepted
20141218 112635:ERROR   Cannot open client listener for grouper (index=3)
20141218 114339:WARNING Grouper version 2014 already connected, cannot add grouper (index=2)
20141218 112635:        Grouper connection closed (index=3)
```

Troubleshooting

Debug mode

In DEBUG mode, DRG Server produces more information to log file *drgsrv.ini*, and may help detecting problems in the system. It can be switched on by adding the following lines to the configuration file drgsrv.ini:

[DEBUG]

DEBUG=1

By setting DEBUG=0 or removing the section, the debug mode is disabled.

There is also alternative way to set the debug mode:

Windows: Debugging mode can be activated by setting Windows registry key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\DRGServer\Parameters\Debug

to value 1 and restarting DRG Server. Debugging can be disabled by setting the key back to 0 and restarting DRG Server.

Linux: debugging can be activated by starting DRG Server with command line switch *–d*: *drgsrv –d&*.

Error messages

Some common error messages and their explanations follow:

Cannot open x	Configuration file is not found and DRG Server cannot be started. 'x' contains the complete path to *drgsrv.ini*, check that the directory is correct and the file exists.
Parameter x missing in section x in file x	Obligatory parameter is missing in *drgsrv.ini* and DRG Server cannot be started.
Parameter x missing or out of range (x-x) in section x in file x	Obligatory numeric parameter is missing in *drgsrv.ini* and DRG Server cannot be started. It is also possible that the numeric value is too low or high, valid range is given in parenthesis.
Cannot open client listener in port x	TCP port cannot be opened for listening. Check that the port number defined in parameter CLIENT_PORT in section [MULTIVERSION] is free. (You can list reserved ports by running command line tool netstat –a).
"Parameter RECORD_FORMAT: unknown file format x	Parameter RECORD_FORMAT in section [MULTIVERSION] must be either "FREE" of "FIXED".
Parameter DATE_FORMAT is not any of valid format strings: ymd8,dmy8,ymd10,dmy10 or y4	Parameter value of DATE_FORMAT in section [MULTIVERSION] must be one of the valid formats in the list.

Error opening grouper listener socket x	TCP port cannot be opened for listening. Check that the port number defined in parameter PORT in section [GROUPER] is free. (You can list reserved ports by running command line tool netstat –a).
Error opening command listener socket x	TCP port cannot be opened for listening. Check that the port number defined in parameter PORT in section [COMMAND] is free. (You can list reserved ports by running command line tool netstat –a).
Error in parameter name x: must be of form x, where n=1-x	Parameter VERSION in section [VERSIONS] must contain the version index (e.g. VERSION2), where the integer must be between 1-MAXGROUPERS.
Missing or out of range (x-x) parameter x, cannot start grouper version x	Numeric parameter in section [VERSIONS] is not valid, check the range and format.
Missing or invalid parameter x, cannot start grouper version x	Parameter in section [VERSIONS] is not valid, check the format.
Cannot start grouper version x	Error while trying to start the grouper instance (yagg.exe).
Default grouper x is not defined in section [MULTIVERSION]	You should define which grouper version is used for patient records where dynamic allocation using discharge date fails.
Section x not found, cannot start groupers	Section is not found in the configuration file, add the section.
Failed to accept new grouper, re-initializing grouper listener	System call to accept new connection has failed. This should never happen, contact FCG Prodacapo.
Cannot read grouper version, terminating grouper	Grouper instance doesn't give information about the grouper version during start-up.
Unknown grouper version x, not connected	Version ID given by the grouper instance doesn't match the ID in *drgsrv.ini*. Version ID is fixed in the grouper version, and you should always use the same ID in *drgsrv.ini* and subdirectory name. For instance, version DRG2014 should be located in subdirectory "DRG2014", and it should print "INFO: Version DRG2014" into DRG2014\yagg.log.
No client port defined for grouper (index=x)	Every grouper should have a valid client port. This should never happen, contact FCG Prodacapo.
Cannot open client listener for grouper (index=x)	Every grouper instance must have a TCP port where clients can connect. Check that CLIENT_PORTx defined in [VERSIONS] is not occupied. (You can list reserved ports by running command line tool netstat –a).
Maximum number of clients exceeded, cannot accept new connection	Maximum number of clients is defined in drgsrv.ini, section [CLIENTS], parameter MAX_CLIENTS. You should probably increase this value and reboot DRG Server.

Error accepting new connection in port x	System call to accept new connection has failed. This should never happen, contact FCG Prodacapo.
Error reading client (index=x): Missing SB. Closing client connection	Error while reading data from a client. The message is invalid, *start block* character is not found in the beginning of the message.
Error reading client (index=x): Unexpected SB. Closing client connection	Error while reading data from a client. The message is invalid, *start block* character is found when expecting message data or end block character.
Error reading client (index=x): Line too long. Closing client connection	Client is sending a message that is too long for DRG Server. Check that the client is sending a valid message to DRG Server.
Error reading client (index=x, error code=x). Closing client connection	Other error while reading message from client. Check that the client is sending a valid message to DRG Server. Also network errors are possible.
Grouper error (grouper index=x). Restarting grouper	Grouper instance is not responding to the grouping request. Check file *yagg.log* in the grouper subdirectory for more information.
Client sending data to a grouper which is not up (client index=x, grouper index=x)	Grouper instance has been terminated and client tries to use that instance, or grouper instance is starting but not ready yet.
Error writing to client (index=x). Closing client connection	Client socket is not accepting messages.
Socket error in grouper connection (index=x), restarting grouper	Internal socket error. This should never happen, contact FCG Prodacapo.
Rejecting command connection from x	Only connections from localhost are allowed to the command port. Someone is trying to connect from other host.
Grouper process start failed, version=x, index=x	Grouper instance cannot be started, error executing yagg.exe.
Grouper home directory name too long	The version identification should not be this long.
Invalid grouper index (x)	Internal error. This should never happen, contact FCG Prodacapo.
Error writing to grouper (index=x)	Grouper instance is not accepting the grouping request. Check file *yagg.log* in the grouper subdirectory for more information.
Error reading from grouper (index=x)	Grouper instance is not responding to the grouping request. Check file *yagg.log* in the grouper subdirectory for more information.

FCG.

Building Clients

Interface

DRG Server provides a TCP/IP connection with multiple client support. The interface is technically implemented as a TCP (server) socket, which is connected by clients. Clients need to know the IP address of the server, and the TCP port to connect.

There are two different ways to connect: connect directly to the port dedicated to a specific grouper version, or connect to a port which allocates the correct grouper based on the discharge date in the patient record (multi-version port). Technically these connections are identical from the client's point of view.

Client programs must complete following steps to connect to the server:

1. (Windows clients only) Initialize Winsock with **WSAStartup** call
2. Create a stream socket with **socket** call
3. Connect the socket to the server with **connect** call

These system calls have small differences in different operating systems. Windows operating systems use Windows Sockets, which must be initialized before using other calls. In Linux environments, **socket** can be called directly. See example later in this chapter to see an actual implementation in Windows.

To terminate the connection, client must complete following steps:

1. Close the socket with **closesocket** call (Windows clients) or **close** (Linux clients)
2. (Windows clients only) Terminate Winsock with **WSACleanup** call

Clients can group several cases during a connection, or they can connect and disconnect every time they group a patient case. Connecting causes a minor delay, and if grouping is intensive, connection should not be broken in the middle of grouping actions. However, proper error handling is necessary, since network connections can be broken during long sessions.

Message structure

Description

TCP/IP provides a reliable stream connection, where packets are guaranteed to arrive in the right order. However, depending on the physical network structure and TCP/IP implementation, packets can be divided into smaller packets during transfer. This means, that when client sends a packet, the server may receive it in two pieces. This is one reason why enveloping is usually performed during TCP/IP transfer.

DRG Server uses a very simple enveloping, where only two additional bytes per message are needed. Combined with the properties of TCP/IP standard, this is sufficient to transfer messages in a safe and reliable way.

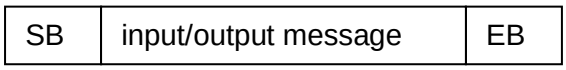

Figure 2. Message format

The message is started with a *start of block* (SB) byte. Following bytes are considered to be part of the message, until an *end of block* (EB) byte is read. The message format between SB and EB is defined in the grouper version definition file, which is given as an argument while initializing the grouper. More details about different data formats can be found in *Appendix A*.

Values for SB and EB are 0x01 (binary 00000001) and 0x02 (binary 00000010), respectively.

NordDRG example

The grouper definition file describes an input message to contain a case identification, five ICD-10 diagnosis pairs, 10 procedures, age, gender, discharge status and length of stay. These fields are separated by commas. Output record is defined to contain the case identifier, RTC, MDC and DRG codes.

With these definitions, input record could look like this:

```
<SB>PSISKL19573,N64.9,,C50.4,,,,,,,,HAE10,HAD30,,,,,,,,,46,2,08,
4<EB>
```

The message starts with SB byte, followed by the actual message. There are actually two diagnosis pairs and two procedure codes, the rest of the codes are empty. Age is 46, gender 2 and discharge code 08. Gender and discharge coding is defined in the definition file and can vary with the grouper version. Message is terminated by the EB byte. Note the typical arrangement of diagnoses in NordDRG grouping: Diagnosis pairs contain only one diagnosis code and the other one is empty.

Output record could look like this:

```
<SB>PSISKL19573,0,9,258<EB>
```

There is the case identifier followed by fields produced by the grouper. The second field (0) is the RTC code, which is the technical return code of the grouper. This is zero, when the grouping is successful. Other codes indicate missing, erroneous or incoherent input information. Third and fourth fields contain MDC and DRG.

Client example

A Simple DRG Server client

This client is written in C programming language. Any standard C/C++ development tool can be used. The client is easy to modify to other platforms such as Linux.

```
/*
  A Windows command line utility to test DRG Server.

  It reads data from "indata.txt" and writes grouping results
  to standard output.

  Parameters: IP address of DRG Server.

  Copyright 1999-2018 FCG Prodacapo Oy
  Version 1.0

*/

#include <stdio.h>
#include <winsock.h>
#include "gtcpio.h"
#define MAXLEN 1024
#define SERVERPORT 7171
WSADATA ws;

void main(int argc, char **argv)
{
    FILE *infile;
    SOCKET client_sd;
    int ret;
    char msgbuf[MAXLEN+1];
    char *ptr;
    struct sockaddr_in srv_addr;

    if (argc != 2)
    {
        fprintf(stderr, "Usage: ggclient server_ip_address\n");
        exit(1);
    }
    /* initialize winsock */
    if (WSAStartup(0x0101, &ws) != 0)
    {
        fprintf(stderr, "Failed to initialize winsock\n");
        exit(1);
    }
    /* set server address and port */
    memset((char *) &srv_addr, 0, sizeof(srv_addr));
    srv_addr.sin_addr.s_addr = inet_addr(argv[1]);
```

```
    srv_addr.sin_port = htons(SERVERPORT);
    srv_addr.sin_family = AF_INET;
    /* create socket */
    if ((client_sd = socket(PF_INET, SOCK_STREAM, 0))
         == INVALID_SOCKET)
    {
        fprintf(stderr, "Error creating socket\n");
        exit(1);
    }
    /* make connection */
    if (connect(client_sd, (struct sockaddr *) &srv_addr,
         sizeof(srv_addr)) == SOCKET_ERROR)
    {
        fprintf(stderr, "Error connecting DRG Server\n");
        exit(1);
    }
    /* open file to be grouped */
    if ((infile=fopen("indata.txt", "r")) == NULL)
    {
        fprintf(stderr, "Error opening indata.txt\n");
        exit(1);
    }
    /* group each record in the file */
    while (fgets(msgbuf, MAXLEN, infile) != NULL)
    {
        /* remove CR+LF */
        if ((ptr=strchr(msgbuf, '\n')) != NULL)
            *ptr='\0';
        printf("writing:%s\n", msgbuf);
        /* Call external WriteMessage to make a message
           envelope and send message */
        if ((ret=WriteMessage(client_sd, msgbuf)) != MSGIO_OK)
        {
            printf("Error writing socket (ERR:%d)\n", ret);
            exit(1);
        }
        /* Call external ReadMessage to receive message and
           strip envelope */
        if ((ret=ReadMessage(client_sd, msgbuf, MAXLEN, 5))
            != READ_MSG)
        {
            printf("Error reading socket (ERR:%d)\n", ret);
            exit(1);
        }
        printf("read:%s\n", msgbuf);
    }
    closesocket(client_sd);
    /* terminate winsock */
    WSACleanup();
}
```

Grouper definition files

Overview

Every grouper version must have a main definition file specifying the grouper input and output records and the grouping logic itself. The general structure of the definition file is:

```
// Constants
define MALE…
define FEMALE…
define TRANSFER…
define DEATH…
```

Definition of codes for patient sex and discharge status. These MUST be edited to be compatible with the patient record.

```
// Input file definition
Input {
```

Input fields describing the patient record sent by the client. These MUST be edited to be compatible with the patient record. The number of diagnoses and procedures can be altered.

```
}
```

```
Calcfield {
```

Diagnosis pair definitions. If you change the number of diagnosis in the input section, you must also have corresponding diagnosis pair definitions here.

```
}
```

```
// Diagnoses and procedure list definitions
define ANYDG…
define COMPDG…
define ANYPR…
```

These definitions define which diagnoses and procedures are actually used in the grouping.

```
Output {
```

Group fields RTC, MDC and DRG are defined here.

```
}
```

```
// include NordDRG definition
include crypted <logicfile>.ggc";
```

Typically input, output and code definitions are not changed very often and remain the same in different (year) versions. If changes are made, however, you should note that the changes in input record also affect dynamic allocation of the grouper version based on patient discharge date.

Input Section

The structure of the fields of the input record (name, type, place and length), are defined in the `Input` section.

The format of the input record is defined by one of the following expressions:

```
Format Fixed;

Format Free;

Format Free Separator 'x' Surrounder 'y';
```

The formats are interpreted as follows:

Fixed format

The fields are fixed length and in a fixed place as informed in the definition expression of the field.

Free format

The order of the fields is defined, but their length might vary.

`Separator 'x'` command defines the field separator character, which is placed in a record of free format between two fields.

`Surrounder 'y'` command sets the character, which is at the beginning and in the end of each field.

If `Format` is missing, it is assumed that the format of the input data is fixed.

The definition of the fields depends on the format. Therefore fixed and free formats are presented here separately.

Fixed format

The fields of the record are defined by following expressions

```
Field (symbol) Is (type) Length (length) At (place) ;

Field (symbol) Is (type) Length (length);
```

These expressions define the *´symbol´* field, the length of which is *´length´*. The place of the first character in the record is *´place´*. The places in the record are calculated starting from 1. If the place is not given, the field is placed to begin after the last field of the record. *'type'* defines the data type of the field. The value of this parameter is *'integer'* or *'string'*. For example:

```
Field DG1A Is String Length 5;

Field AGE Is Integer Length 5 At 6;
```

Free format

The fields of the record are defined by the following expression

```
Field (symbol) Is (type) ;
```

When defining the fields in free format it is not necessary to present either the place or the length of the field. The type of the field is given as in fixed format. For example:

```
Field DG1A Is String;

Field AGE Is Integer;
```

Diagnosis codes

NordDRG uses ICD-10 diagnosis classification. The principal diagnosis (DG1A and DG1D) has a special effect in the grouping; the order of other diagnoses is not significant to the resulting DRG.

The manifestation diagnosis (asterisk code) and the etiological diagnosis (dagger code) must be separated into different columns. Thus every diagnosis pair has two columns.

Examples

Following table gives examples how to place diagnoses into columns in the NordDRG grouper:

Diagnosis	DG1A	DG1D	DG2A	DG2D
dg1=N291*A181	N291	A181		
dg1=A181+N291	N291	A181		
dg1=J450, dg2=L272	J450		L272	

row 1: diagnosis pair separated into two columns

row 2: same as row1, but the dagger code is first

row 3: two different diagnoses, not a pair

Diagnoses are defined in the input section using identifiers DG1A, DG1D, DG2A, DG2D etc. In addition, diagnosis pairs are created in *calcfield* section:

```
CalcField
{
  String DG1C=DG1A+DG1D;
  String DG2C=DG2A+DG2D;
  String DG3C=DG3A+DG3D;

… for each diagnosis

}
```

All diagnoses MUST also be listed in ANYDG macro, for instance:

```
define ANYDG DG1A,DG1D,DG1C,DG2C,DG2A,DG2D,DG3C,DG3A,DG3D;
```

This defines 3 diagnoses to be used in the grouping. All but the principal diagnosis are defined in the COMPDG macro, for instance:

```
define COMPDG DG2C,DG2A,DG2D,DG3C,DG3A,DG3D;
```

This matches with the previous example with 3 diagnoses.

Procedure codes

NordDRG uses NCSP procedure classification. The order of the procedures given to the grouper is not significant. All "extra codes" should be given, including "Z" codes.

Procedures are defined in the input section using identifiers PR1, PR2, PR3 etc. All procedures MUST also be listed in the ANYPR macro, for instance:

```
define ANYPR PR1,PR2,PR3,PR4,PR5,PR6,PR7,PR8,PR9,PR10;
```

This defines 10 procedures to be used in the grouping.

Other input variables

Age is given as days (also for adults) in admission.

The **length of hospital stay** is given as the number of calendar days (discharge date – admission date + 1). Note that if the admission date equals to discharge date, the length of stay is given value 1. (In addition, the Swedish SK-OP version uses value 0 for outpatient cases that are not admitted into ward).

Macros FEMALE and MALE define the codes used for **patient sex**, for example:

define MALE "1","ML";

define FEMALE "2","FM";

Note that several codes can be given. Code values must be given in quotation marks (""), even when numeric. If several codes are used, they must be separated with a comma (,).

The **discharge status** has 2 specific macros DEATH and TRANSFER, for example:

define DEATH "E";

define TRANSFER "R";

The death of the patient is significant in the grouping logic; also the condition that the patient has *not* died during the treatment is checked in some groups (e.g. short therapy).

TRANSFER indicates that the patient has been transferred to another unit. Other discharge codes are not specified to the grouper and do not affect the grouping logic.

Summary of Input Section

A summary of expressions to be used in the Input section (* = only in free format, ** = only in fixed format):

```
Format Fixed; (**)

Format Free Separator 'x' Surrounder 'y'; (*)

Format Free; (*)

Field (symbol) Is Integer|String Length (length) At (place) ;
(**)
```

```
Field (symbol) Is Integer|String Length (length) ; (**)
Field (symbol) Is Integer|String ; (*)
```

Following fields must be defined in input section:

1. DG1A (manifestation/asterisk diagnosis 1)
2. DG1D (etiological/dagger diagnosis 1)
3. DG2A (manifestation/asterisk diagnosis 2)
4. DG2D (etiologically/dagger diagnosis 2)
5. all other diagnosis pairs DG***N***A and DG***N***D (up to 20 diagnosis pairs)
6. PR1 (procedure code 1)
7. PR2 (procedure code 2)
8. all other procedure codes PR***N*** (up to 30 procedures)
9. SEX (patient gender)
10. AGE (patient age in days)
11. DISCHRG (discharge status)
12. DURATION (length of hospital stay)

Output Section

The format of the output record, as well as the input record, is defined by one of the following expressions.

```
Format Fixed;
Format Free;
Format Free Separator 'x' Surrounder 'y';
```

See Input Section for details, how these expressions are interpreted.

If "Format" is missing, it is assumed that the format used in output is fixed.

Data fields (Field)

It is possible to add fields into the input record in two ways. The whole input record can be added with command:

```
AppendFields;
```

In this case the input record is set at the beginning of the output record and other fields are appended after it.

Another option is to define all the input data fields separately by the following expressions:

fixed format: `Field (symbol) Length (length) At (place);`

free format: `Field (symbol);`

NOTE! If `AppendFields` expression is used, the numbering of the positions of other fields starts from 1. For example:

```
AppendFields;
```

```
Group DRG is String Length 4 At 1;
```

This expression defines the output record, which has the complete input record in the beginning, followed by the group field `DRG`.

Group fields (Group)

The definition of group fields starts with the word `Group` and has the same format as the definitions of the data fields of the `Input` section.

Fixed format:

```
Group (symbol) Is (type) Length (length) At (place) ;
```

```
Group (symbol) Is (type) Length (length) ;
```

Free format:

```
Group (symbol) Is (type) ;
```

Group fields in NordDRG are:

DRG The main results of the grouping.

MDC The main diagnosis category of the DRG.

RTC The technical return code of the grouping. Zero (0) if the grouping was completed technically without errors.

Appendix A: GDPR Compliance of FCG Prodacapo DRGServer

FCG Prodacapo DRG Server doesn't contain and store personal information and thus it's not covered by GDPR (General Data Protection Regulation, (EU) 2016/679).

(In new versions data contents and security by design is taken as a corner stone and future versions implements security as planned).

Grouping is a function that can use arguments that are derived from personal information. Groupers do not implement any storage. Thus, groupers do not contain any patient information. The other systems utilizing and integrating with Prodacapo Grouper (Prodacapo Regional, Prodacapo Costing, other customer systems including but not limited to PAS systems) document and implement their GDPR compliance in their product context depending on their data storage, security, logging etc. solutions in use.