



Gesellschaft für medizinische
Bildverarbeitung mbH

MedCom Client/Server Database

Technical Documentation

Author	Roland Ohl
Revision	0.3
Date	7.10.02
Filename	DBTechnicalDocu.doc
Pages	42

Document History

Revision	Date	Change	Name
0.1	07.11.01	Initial Revision Taken from ScanNT 3.3	M. Grimm
0.2	26.11.01	Revision for Exomio 1.1	R. Ohl
0.3	7.10.02	Revision for TeleConsult 2.5	R. Ohl
		Signature	
		Released	

Table of contents

1	DOCUMENT PURPOSE	5
2	CLIENT/SERVER-ARCHITECTURE OVERVIEW	6
3	MEDCOM DATABASE CLIENT	8
3.1	Database data handling	8
3.2	Defined data file types	8
3.3	Database Tables	9
3.3.1	Table <i>Aerzte</i>	10
3.3.2	Table <i>Patienten</i>	11
3.3.3	Table <i>PatientStudys</i>	11
3.3.4	Table <i>Daten</i>	12
3.3.5	Table <i>Funktionen</i>	12
3.3.6	Table <i>Drives</i>	12
3.3.7	Table <i>Discs</i>	13
3.3.8	Tables <i>OptionsFeld1</i> , <i>OptionsFeld2</i> and <i>OptionsFeld3</i>	13
3.3.9	Table <i>Information</i>	13
3.3.10	Table <i>Messages</i>	13
3.3.11	Table <i>StudyTypes</i>	14
3.4	Database Registry settings	14
3.4.1	Sub-Key Directories	14
3.4.2	Sub-Key ExChange	14
3.4.3	Sub-Key <i>Files</i>	16
3.4.4	Sub-Key Service -> Settings	16
3.4.5	Sub-Key Settings	16
3.5	Outsourcing data	18
3.5.1	Outsourcing to MO	18
4	INTERFACE WITH THE MAIN APPLICATION	20
4.1	Windows messages	20
4.2	Information exchange via Registry	21
4.3	Architecture of the Client	22
4.3.1	<i>C...</i> -classes	23
4.3.2	<i>TDialog...</i> -classes	23
4.3.3	<i>R...Set</i> -classes	24
4.3.4	<i>T...</i> -classes	24
5	DATABASE SERVER FUNCTIONALITY	26
5.1	Client Server Communication	26
5.2	Server Registry Entries	26
5.3	List of Client Server Network Commands	26
5.4	Internal Software Architecture	33

MedCom Client Server Database

5.4.1	C...-Classes.....	34
5.4.2	TDialog...-Classes	34
5.4.3	R...Set-Classes	34
5.4.4	T...-Classes	34
6	MEDCOM DICOM STORESCP	35
6.1	Dicom Communication.....	35
6.2	Connection to the Database.....	37
6.3	Registry Entries	39
6.4	Code Structure.....	40
7	VERIFICATION / TEST CASES	41
7.1	Hazard: Error in Saving	41
7.2	Hazard: Error in loading data.....	41
7.3	Hazard: Data security	42

1 Document Purpose

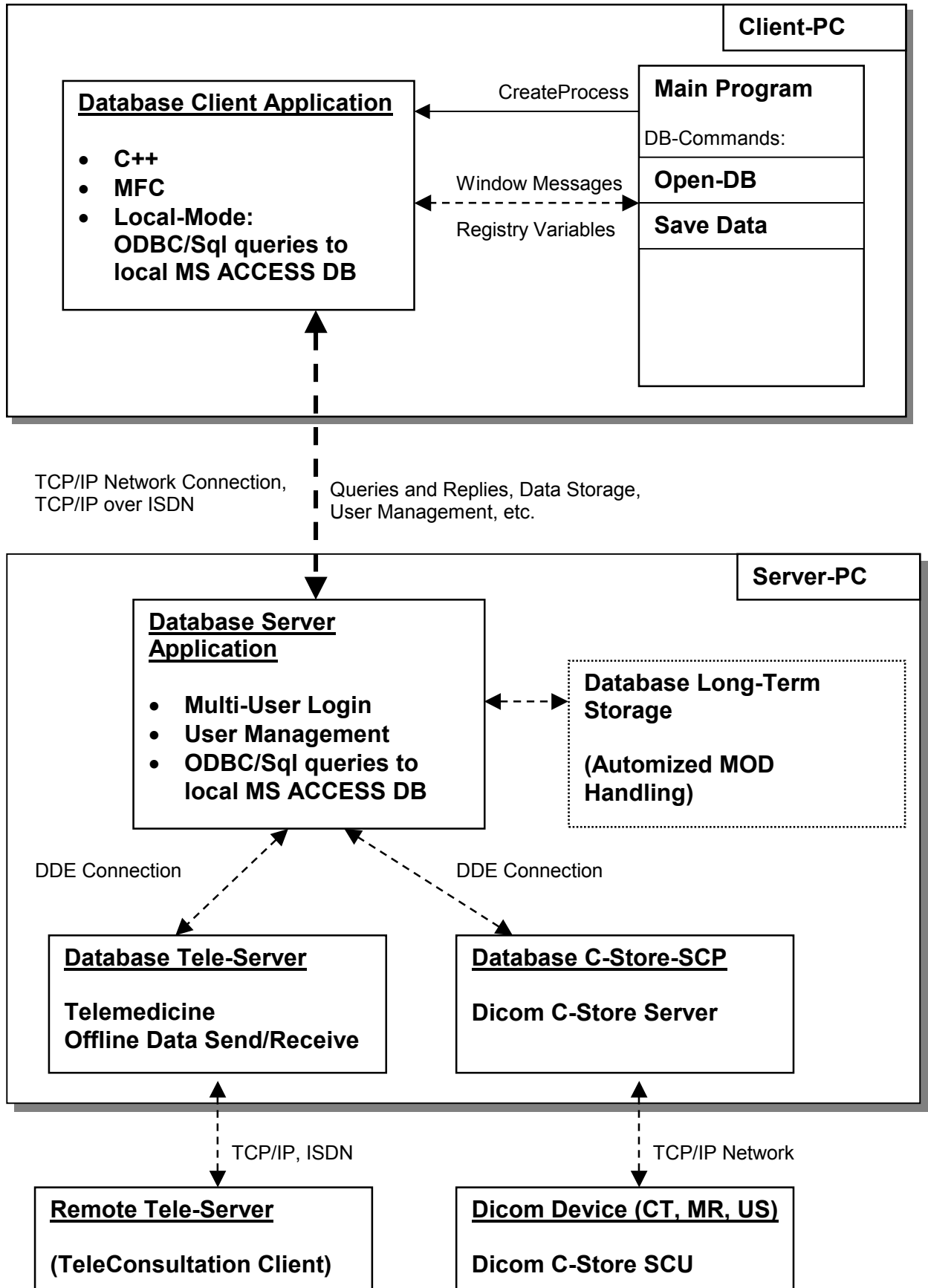
This document describes the architecture of the MedCom Client/Server database. The database consists of a Client Executable and a Server Application. The Main Program (ScanNT, Exomio, TeleConsult, etc.) connects to the database client via registry data exchange, the client connects to the server over different kind of connection types or opens a local database directly.

Beside the dataflow between the Client and Main Application the Database serves as a Dicom C-Store service provider.

Since the Client Application implements also a local database access (i.e. without connecting to a server) both application use exactly the same set of database tables, query/retrieve statements, etc.. The term 'Database' in this document refers to both applications if not noted separately.

It should be noted that the Database Client may run in non-server mode – in this mode the Application doesn't display any server related user-interface elements. This mode is either defined at compile time or (more common) via dongle licence bit at start-up time.

2 Client/Server-Architecture Overview



3 MedCom Database Client

The MedCom-Database Client is a C++ object oriented MFC (Microsoft Foundation Classes) ODBC (Open Database Connectivity) Application. The database itself is designed with Microsoft® Access 97. The main application will call this program at the first access. The Communication between the main application and the database client application is realised through windows messages and registry entries.

The database application supports several users with different access rights. Consequently, every user has to login for the first time and can only access the patients he examined, except he has the right to see all patients.

These datasets are assigned to patients and studies that can be stored by the MedCom Database. Studies are used to store datasets produced by the main application which are somehow related together. The duration of a study is defined as the time between the first and the last stored dataset.

The data is stored at the harddisc. To handle an “harddisc out of space” problem the user can move datasets to a removable magneto optical drive or other drives on the PC or in a local area network.

3.1 Database data handling

The database application physically stores temporary generated files in a specified path on the harddisc. The naming of these files follows the convention:

<unique number.type identifier>

unique number : a database created unique value
type identifier : a shortcut identifying the data type

Additional information which describes the files, the studies, the patients and the users of the database are stored in a relational database, which structure is described in chapter 3.3.

3.2 Defined data file types

The MedCom Database supports currently 24 different dataset formats. The internal type identifier of these types is equal to the ending of the physically stored file. The table below describes the different types and their IDs. Moreover you can store every file-type under the format MCF, which contains files which are added to the database and now are in a compressed format.

Data type	Description
TYP_BILD	2D-images
TYP_FRAMES	Ultrasound frame sequences
TYP_KONFIGURATION	Configuration files, which can contain different kind of configurations for Frames, Volumes, Dicom images etc.
TYP_VIDEO	Videos
TYP_VOLUMENMITFRAMES	Volumes (3D-datasets) which are assigned to Frames
TYP_VOLUMENOHNEFRAMES	Volumes (3D-datasets) which are not assigned to Frames
TYP_DICOMSLICE	Sequences of DICOM 2D-images

TYP_RTPLAN	DICOM RT-Plans
TYP_AUDIO	MedCom defined Audio-files
TYP_PANMITFRAMES	MedCom defined Panorama-files assigned to Frames
TYP_PANOHNEFRAMES	MedCom defined Panorama files not assigned to Frames
TYP_SLIDIRECT	Same as TYP_DICOMSLICE, but will be stored directly from the Main application
TYP_RTSTRUCT	DICOM RT-Struct
TYP_RTIMAGE	DICOM RT-Image
TYP_RTDOSE	DICOM RT-Dose
TYP_SECONDARYCAPTURE	DICOM Secondary Capture images
TYP_CHA	Special files for the Charisma application
TYP_MSK	3D-Mask files for Volumes or Configurations.
TYP_PRV	Files showing a Preview for the Images
TYP_MCA	Files which will be attached to a message in the Telemedicine-version
TYP_MCD	MedCom format used to import and export data from the database.
TYP_TPL	Templates used in the Charisma-Application
TYP_MCF	Files which will be added directly from the database to a study will be stored under this format.
TYP_TUT	Files, which contain Tutorials for the Ultrasound Simulator

Some files are stored in subject to the data from which they are produced. That means, the database is unable to store that kind of (child) data without their corresponding parent entry. Other types are only used in addition to other existing files. These additional files will be named in the following way:

<unique number of the master file.type identifier of the master file. type identifier of the slave file>

unique number of the master file : a database created unique value for the master file

type identifier of the master file : a shortcut identifying the data type of the master file

type identifier of the slave file: a shortcut identifying the data type of the slave file

For example you could have a volume with the identifier 123, which is stored as 123.VOL. For this volume you will find an entry in the database table *Daten*. To this volume you can have an additional 3D-Mask-File which will be stored under the name 123.VOL.MSK, with no entry in the database.

3.3 Database Tables

The MedCom Database is based on a relational structure of database tables. The current version consists of 13 tables.

Figure 1 shows the tables and there relationships. Relations with assigned with **1-∞** have referential integrity. Other relations are without referential integrity.

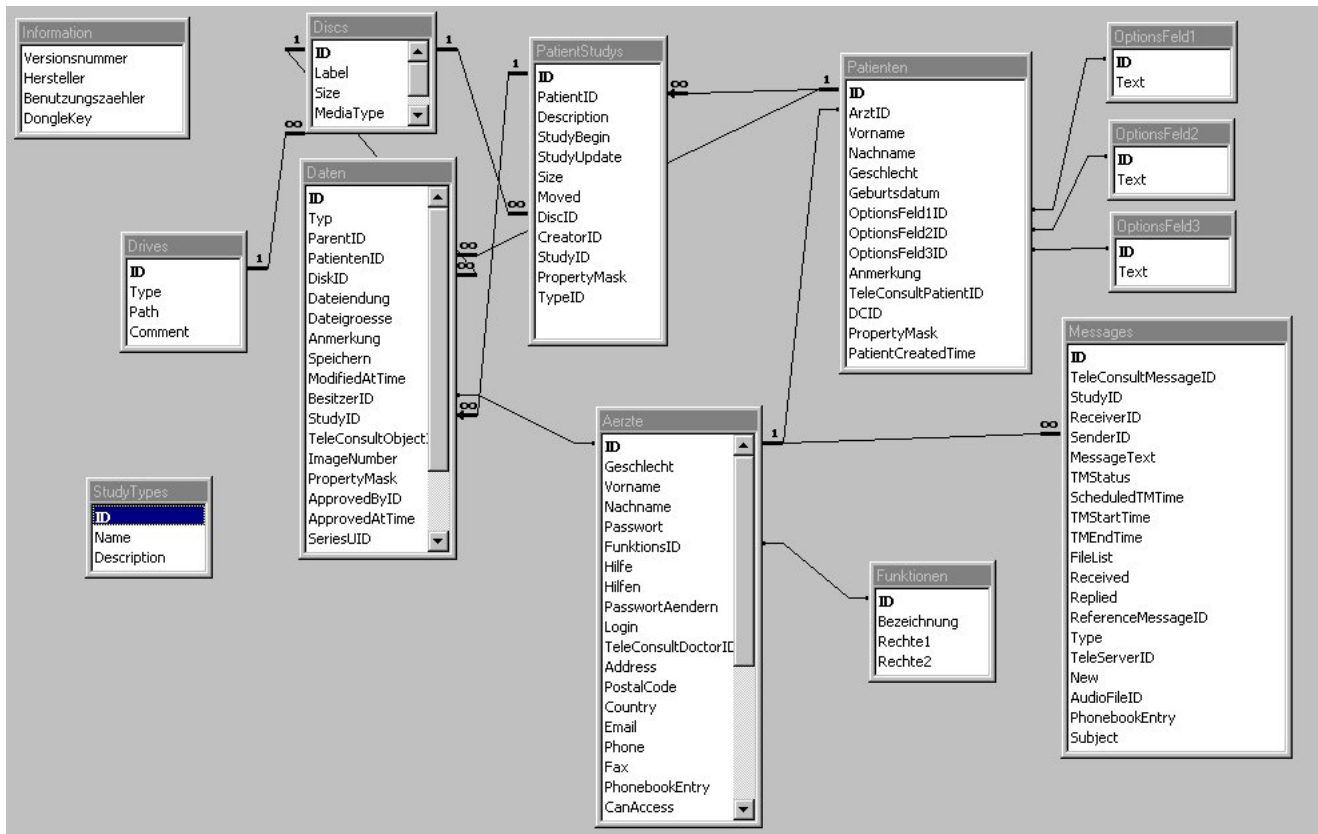


Figure 1: Relationships between database tables

In the following the database tables and their relations will be described. (In the description screenshots with German terms are used, which have the following meaning: *Feldname* = Fieldname, *Felldatentyp* = Datatype, *Beschreibung* = Description, *Ja/Nein* = Yes/No, *Zahl* = Number)

3.3.1 Table *Aerzte*

The table *Aerzte* describes the user, respectively the doctors using the application. For the telemedicine-version there are also communication partners stored as doctors. The table consists of the following fields.

MedCom Client Server Database

Feldname	Felddatentyp	Beschreibung
ID	AutoWert	
Geschlecht	Ja/Nein	Sex
Vorname	Text	Firstname
Nachname	Text	Lastname
Passwort	Text	Password
FunktionsID	Zahl	ID of the Function
Hilfe	Ja/Nein	Sollen Hilfen (Assistenten) angezeigt werden?
Hilfen	Zahl	Welche Hilfen sollen angezeigt werden.
PasswortAendern	Ja/Nein	Benutzer muß Passwort bei der nächsten Anmeldung ändern.
Login	Text	Login-Name für Datenbank-Zugriff über einen Client
TeleConsultDoctorID	Text	D(für Doktor)DongleKey (5 Zeichen) - Zeit hexadezimal (4) - ID (5) insgesamt 21 Zeichen
Address	Text	Adresse (Strasse und Hausnummer des Arztes)
PostalCode	Text	Postleitzahl
Country	Text	Land
Email	Text	E-mail-Adresse
Phone	Text	Telefon-Nummer des Arztes
Fax	Text	Fax-Nummer des Arztes
PhonebookEntry	Text	Name des Phonebook-Eintrages(Zentrum) über den der Arzt erreichbar ist.
CanAccess	Ja/Nein	Legt fest ob dieser Arzt Zugriff auf die lokale Datenbank hat oder ob er nur als Empfänger von Nachrichten gespeichert ist.
City	Text	Stadt
ZeigeBeiAnmeldung	Ja/Nein	
MobilePhone	Text	
DataPhone	Text	Number of the phone over which the data shall be transferred
PropertyMask	Zahl	Mask for different properties of a doctor, e.g. receiving broadcasts
FilterFirstName	Text	FirstName-Field for the patient-filter
FilterLastName	Text	LastName-Field for the patient-filter
FilterID	Text	Patient-Dicom-ID-Field for patient-filter
FilterType	Zahl	0 = all, 1= last month, 2= last three month, 3 = last six month
CurPatientID	Zahl	ID of the last selected Patient in the patient-view
CurStudyID	Zahl	ID of the last selected Study in the patient-view
FilterStudyType	Zahl	ID of the Study Type which shall be shown in the patient list

Figure 2: Table Aerzte

3.3.2 Table Patienten

In the table *Patienten* the description of the patients is stored.

Feldname	Felddatentyp	Beschreibung
ID	AutoWert	
ArztID	Zahl	ID of the doctor, who entered the patient in the database
Vorname	Text	Prename of the patient
Nachname	Text	Lastname of the patient
Geschlecht	Zahl	0 = masculine, 1 = feminine, 2 = none
Geburtsdatum	Zahl	Birthday
OptionsFeld1ID	Zahl	ID of the assigned value in the OptionsFeld1-table
OptionsFeld2ID	Zahl	ID of the assigned value in the OptionsFeld2-table
OptionsFeld3ID	Zahl	ID of the assigned value in the OptionsFeld3-table
Anmerkung	Text	Medical history and/or comments to the patient
TeleConsultPatientID	Text	Special ID for the patient used in the telemedicine version
DCID	Text	Dicom-ID, respectively clinical patientID of the patient
PropertyMask	Zahl	PropertyMask for patients
PatientCreatedTime	Datum/Uhrzeit	The time when the patient has been created.

Figure 3: Table Patienten

3.3.3 Table PatientStudys

In this table the values for a study is stored. A study is defined of a period of interrelated examinations.

MedCom Client Server Database

Feldname	Felddatentyp	Beschreibung
ID	AutoWert	ID of the Study
PatientID	Zahl	ID of the patient to whom this study is assigned
Description	Memo	Description of the patient-study
StudyBegin	Datum/Uhrzeit	Begin of the study
StudyUpdate	Datum/Uhrzeit	Update of the study
Size	Zahl	Size of the study
Moved	Ja/Nein	Yes, if the study is on an external media, otherwise No
DiscID	Zahl	ID of the disc, on which the study is stored
CreatorID	Text	TeleConsultDoctorID of the person which created the study
StudyID	Zahl	ID of the study in the database of the creator, identifies the study together with theTeleConsultPatientID
PropertyMask	Zahl	PropertyMask for Studys
TypeID	Zahl	ID of the Type of the Study

Figure 4: Table PatientStudys

3.3.4 Table Daten

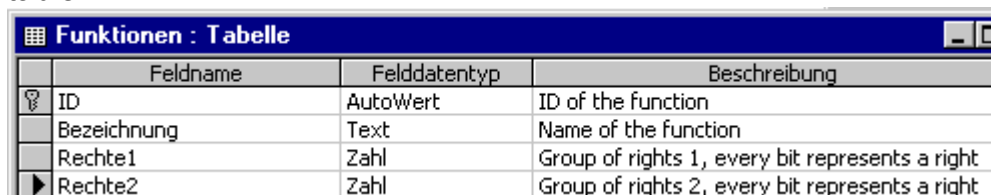
In this table additional information to an image file is stored.

Feldname	Felddatentyp	Beschreibung
ID	AutoWert	
Typ	Zahl	Type of the file, e.g. 1=Frames, etc.
ParentID	Zahl	ID of the file, this file depends on
PatientenID	Zahl	Patient to whom this file is assigned to
DiskID	Zahl	ID of the disc or media, where this file is located
Dateiendung	Text	Filename-ending, e.g. FRM for frames
Dateigroesse	Zahl	Size of the file
Anmerkung	Memo	Comments to the file
Speichern	Ja/Nein	Used for the calculation of the loading time of the file
ModifiedAtTime	Datum/Uhrzeit	Time of the creation of the file
BesitzerID	Zahl	ID of the owner and creator of this object
StudyID	Zahl	ID of the study, this file belongs to
TeleConsultObjectID	Text	Special ID for the telemedicine version, not important for this version
ImageNumber	Zahl	Number of images, which belong to an image sequence
PropertyMask	Zahl	Property Mask for different kind of flags like lock-mechanism
ApprovedByID	Zahl	This file has been approved by the doctor with that id
ApprovedAtTime	Datum/Uhrzeit	The time when the file was approved
SeriesUID	Text	Dicom Series UID of the file, if available
ModalityType	Text	Type of the Modality to which the Series belongs, e.g. CT, MR etc.
PatientPosition	Text	Position of the patient in the series, e.g. HFP, HFS, HFDR, HFDL, FFP, FFS, FFDR, FFDL
ModalityModelName	Text	Name of the Manufacturers Modality Model
TechnicalCommentary	Memo	Technical Commentary for a file, used by Exomio

Figure 5: Table Daten

3.3.5 Table Funktionen

To every user a function with usage rights is assigned. These functions are stored in this table.



Feldname	Felddatentyp	Beschreibung
ID	AutoWert	ID of the function
Bezeichnung	Text	Name of the function
Rechte1	Zahl	Group of rights 1, every bit represents a right
Rechte2	Zahl	Group of rights 2, every bit represents a right

Figure 6: Table Funktionen

3.3.6 Table Drives

In the table drives, locations for data storage are stored.

Feldname	Felddatentyp	Beschreibung
ID	AutoWert	ID , primary key
Type	Zahl	Type of the drive, could be for example 0 - temp drive 1-MO 2-Network 3-CD-Rom, etc.
Path	Text	path on the drive to the data, for example f:\InvivoDB\
Comment	Text	Comment to the drive, for example network mounted, etc.

Figure 7: Table Drives

3.3.7 Table Discs

Discs on which the data will be swapped out are stored in this table.

Feldname	Felddatentyp	Beschreibung
ID	AutoWert	ID of the disc
Label	Text	Label of the disc
Size	Zahl	size
MediaType	Text	Harddisc, MO, etc.
DriveID	Zahl	ID of the drive the disc is assigned to

Figure 8: Table Discs

3.3.8 Tables OptionsFeld1, OptionsFeld2 and OptionsFeld3

For the database three optional fields can be defined. For example profession, etc. the values of each optional field are stored in the tables *OptionsFeld1*, *OptionsFeld2* and *OptionsFeld3*.

Feldname	Felddatentyp	Beschreibung
ID	AutoWert	ID of the OptionField Val
Text	Text	Text to the value

Figure 9: Structure of the tables for the optional fields

3.3.9 Table Information

In the table Information, some information about the state of the database-file is stored.

Feldname	Felddatentyp	Beschreibung
Versionsnummer	Zahl	Versionnumber of the database-file
Hersteller	Text	Manufactor of the database, normally MedCom GmbH
Benutzungszaehler	Zahl	How often has the database application been started
DongleKey	Zahl	Identifier of the dongle

Figure 10: Table Information

3.3.10 Table Messages

In this table, which is only used in the telemedicine-version of the database, the messages used in the telemedicine-context are stored.

Messages : Tabelle			
	Feldname	Felddatentyp	Beschreibung
	ID	AutoWert	ID of the message
	TeleConsultMessageID	Text	Identifying ID of the message in the TeleConsult-Network
	StudyID	Zahl	ID of the Study in the local database this message is referring to, if any
	ReceiverID	Zahl	ID of the receiver in the local database this message should be sent to
	SenderID	Zahl	ID of the sender of the message in the database
	MessageText	Memo	Message text
	TMStatus	Text	Transmission State of the message
	ScheduledTMTime	Datum/Uhrzeit	Planned time for transmission
	TMStartTime	Datum/Uhrzeit	Real transmission start time
	TMEndTime	Datum/Uhrzeit	Real transmission end time
	FileList	Text	List of files which belong to the message
	Received	Ja/Nein	yes - the message has been received, no - the message has been or should be sent
	Replied	Ja/Nein	yes = an answer arrived for the message, no - no answer is available in the moment
	ReferenceMessageID	Zahl	ID of the message in the database this message refers to, if any
	Type	Text	N = Normal Message, Q = Query Message, R = Reply Message
	TeleServerID	Zahl	ID of the message in the TeleServer
	New	Ja/Nein	For received messages this flag marks if a message has been shown or not
	AudioFileID	Zahl	ID of the corresponding audio file, 0 if no audio file belongs to the message
	PhonebookEntry	Text	Phonebook-Entry, address to which the message should be sent, if no receiver is specified
	Subject	Text	Subject of the message

Figure 11: Table Messages

3.3.11 Table StudyTypes

Studys can have different types. These types can be defined through the user. The different study types are stored in this table.

	Feldname	Felddatentyp	
	ID	AutoWert	
	Name	Text	Name of the StudyType
	Description	Text	Description of the StudyType

Figure 12: Table StudyTypes

3.4 Database Registry settings

The Client uses the key `HKEY_LOCAL_MACHINE\Software\MedCom\Database` for saving and retrieving informations of different kind. This information is used to store information for the database application and to exchange information between the database and the main application. In the following the Client registry sub-keys and values will be described.

3.4.1 Sub-Key Directories

In this Sub-Key, paths to directories for the storage of several files are stored:

- *Database-Path* – Directory where the mdb-file for the relational database is stored.
- *DataPath* – Directory where the image-files are stored.
- *LogFilePath* – Directory where the log-files are stored.
- *Phonebook-Path* – Directory where the phonebook with the connection properties is stored.
- *Tmp-Path* – Directory for the temporary storage of files.

3.4.2 Sub-Key Exchange

In this Sub-Key variables for the exchange between the main-application and the database client are stored.

- *Charisma Group 1* – Used for Charisma Application to group doctors.

- *Charisma Group 2* – Same than above.
- *Charisma Group 3*- Same than above.
- *Data PatientID* – ID of the last patient for whom data was loaded.
- *Database action successful* – Says if the last database action was successful.
- *DB window handle* – Handle of the database UI – window.
- *Examinationdate* – Date of the examination of a loaded file.
- *FileComment* – Comment to a file.
- *FileCreationDate* – Date, a file has been created.
- *FileStorageDate* – Date, a file has been stored.
- *FromDBName* – Name of the file to load from the database.
- *FromDBPath* – Path to the file to load from the database.
- *FromDBType* – Type of the file to load from the database.
- *Last Contour ID* – ID of the last loaded contour.
- *Last FrameID* – ID of the last loaded frame.
- *Last Plan ID* – ID of the last Plan ID.
- *Last VolumenID* – ID of the last loaded volume.
- *LastDataTCID* – TeleConsult ID of the last used Data.
- *LastLoadedBirthdate* – Birthdate of the last loaded Patient.
- *LastLoadedDicomID* – Dicom ID of the last loaded Patient.
- *LastLoadedStudyID* – ID of the last used Study.
- *PatFirstName* – First name of the last accessed patient.
- *PatientBirthdate* – Birthdate of a patient.
- *PatientDicomID* – DicomID of the last accessed patient.
- *PatientName* – Complete name of the last accessed patient.
- *PatientSex* – Sex of patient.
- *PatLastName* – Last name of the last accessed patient.
- *PatSelected* – Indicates, if a patient has been selected or not.
- *PictureSelected* – Indicates if a picture has been selected or not.
- *Plantype* – Type of a plan in the Charisma application.
- *SavePossible* – Indicates, if a save is possible or not.
- *SimulationPlanApproved* – Indicates, if a simulation plan is approved or not.
- *ToDBBirthDate* – Birthdate, which shall be written to the database.
- *ToDBComment*- Comment, which shall be written to the database.
- *ToDBDataTCID* – Teleconsult ID, which shall be used through the database.
- *ToDBDCMID* – Dicom ID, which shall be written to the database.
- *ToDBFName* – Firstname, which shall be written to the database.
- *ToDBLName*- Lastname, which shall be written to the database.
- *ToDBModalityModelName* – Modality Model name, which shall be written to the database.
- *ToDBModalityType* – Modality type, which shall be written to the database.
- *ToDBName* – Name of a file which shall be written to the database.
- *ToDBNumFrames* – Number of frames in a file which shall be written to the database.
- *ToDBPath* – Path to a file which shall be written to the database.
- *ToDBPatientPosition* – Patient position, which shall be written to the database.
- *ToDBSeriesUID* – SeriesUID, which shall be written to the database.
- *ToDBSex* – Sex, which shall be written to the database.
- *ToDBStudyDate* – Study-Date, which shall be written to the database.
- *ToDBTechnicalCommentary* – Technical commentary, which shall be written to the database.
- *ToDBType* – Type of a file which shall be written to the database.

- *Volume with Frames*- Indicates if a volume depends on frames or not.

3.4.3 Sub-Key Files

In this sub-key paths to different kind of files used by the application are stored.

- *ClientExe* – Path to the database client executable.
- *DataBase* – Path to the mdb-file.
- *DICOM store* – Path to the DICOM- storescu – service file.
- *Phonebook* – Path to the phonebook defined by MedCom for server access and telemedicine communication.
- *SysPhonebook* – Path to the windows phonebook.
- *UpdateFile* – Path to the executable which is used for updates.

3.4.4 Sub-Key Service -> Settings

Has only the value *Database Data-Path* which stores the path to files on the magneto optical discs.

3.4.5 Sub-Key Settings

Under this sub-key different settings for the database client application are stored. It has three sub-keys which are described below. The values stored directly under this key are:

- *Actual Patient ID* – Stores the ID of the currently in the patient manager or patient-view selected patient.
- *AudioQuality* – Stores the selected quality for audio recording.
- *Datadisc drive* – Stores the ID of the MO-drive.
- *DebugLevel* – Stores the debug level, which influences the output of error information.
- *DebugWindow* – Debug-Window “on” or “off”.
- *Last patient* – Stores the position of the last selected patient in the patient-tree.
- *Last user login* – Stores the time of the last use of the database.
- *Last UserID* – Stores the ID of the last user of the database.
- *New user-login after minutes* – Stores the time after which a new login is necessary, when the user logged out for a short time.
- *Options-Feld 1 – Bezeichnung* – Stores the name for the first optional field.
- *Options-Feld 2 – Bezeichnung* – Stores the name for the second optional field.
- *Options-Feld 3 – Bezeichnung* – Stores the name for the third optional field.
- *Removable drive speed* – Stores the speed of the MO drive.
- *Temp drive speed* – Stores the speed for the drive where the data is stored by default.
- *Use Card Reader* – Shall a card reader be used or not ? Not used in the current version.

3.4.5.1 Sub-Key Settings->MessagesFilter->In

This Key is used to store the settings for the filter of the list with the received messages. This key is only relevant for the telemedicine version.

- *FirstName* – Filter for the senders first name.
- *LastName* – Filter for the senders last name.
- *Meflag* – Indicates if only messages directly sent to the user shall be shown.
- *MessageID* – Filter for the TeleConsultMessageID.
- *Properties* – Indicates which areas of the filter-dialog shall be used for filtering.
- *SenderID* – Stores the filter for the TeleConsultID of the sender.
- *TimeFrom* – First date from which messages shall be shown.
- *TimeTo* – Last date from which messages shall be shown.
- *TimeType* – Stores the time-type for the filter.

- *Type*- Stores which kind of messages shall be shown.

3.4.5.2 Sub-Key Settings->MessagesFilter->Out

This Key is used to store the settings for the filter of the list with the sent messages. This key is only relevant for the telemedicine version.

- *FirstName* – Filter for the receivers first name.
- *LastName* – Filter for the receivers last name.
- *Meflag* – Indicates if only messages sent from the current user shall be shown.
- *MessageID* – Stores the filter for the TeleConsultMessageID.
- *PhonebookEntry* – Filter for the phonebook-entry of the receiver.
- *Properties* – Indicates which areas of the filter-dialog shall be used for filtering.
- *ReceiverID* – Stores the filter for the TeleConsultID of the receiver.
- *State* – Stores the filter for the state of the messages which shall be shown.
- *TimeFrom* – First date from which messages shall be shown.
- *TimeTo* – Last date from which messages shall be shown.
- *TimeType* – Stores the time type for the filter.
- *Type* – Stores which kind of messages shall be shown.

3.4.5.3 Sub-Key Settings->Statusbar

This key stores the settings for the control of the announcements of the status-bar.

- *Error when size less then MB* – Stores when an error message depending on the free space on the temp drive shall be shown.
- *Warn when size less then MB* – Stores when an warning depending on the free space on the temp drive shall be shown.

3.4.5.4 Sub-Key Settings->Tooltips

Stores the settings for the tooltips in the application.

- *Show comment* – Stores if a comment on the data shall be shown.
- *Show expected loading time* – Stores if the expected loading time for a file shall be shown.
- *Show tooltips for data* – Stores if generally tooltips shall be shown.

3.4.5.5 Sub-Key Sonstiges

In this key a various of other settings which are not masked through the other keys are covered.

- *ActPatient ExamID* – Stores the ID of the last in the patient view selected study.
- *Actual ExamID* – Stores the ID of the last selected data item.
- *Actual Patient ID*- Stores the ID of the last selected patient in the patient view.
- *Dicom Timeout* – Stores the timeout after which a DICOM operation shall be aborted.
- *General Timeout* – Stores the timeout for general network operations.
- *Last Doctor* – Stores the login-name of the last doctor who connected to the database.
- *Last FrameID* – Stores the ID of the last selected frames.
- *Last VolumenID* - Stores the ID of the last selected volumen.
- *LastPbkEntry* – Stores the ID of the last selected phonebook entry in the login window.
- *Little Endian* – Stores, if for DICOM little endian encoding shall be used or not.
- *Login Timeout* – Timeout after which the login process shall be aborted.
- *Logout Timeout* – Timeout after which a automatic logout should take place.
- *Own AE* – Configuration setting for the DICOM sending.
- *PC-IP-Adress* – Stores the default IP-address for DICOM sending.
- *PC-Port-Adress* – Stores the default port for DICOM sending.
- *SortInbox* – Stores the sort-order of the inbox.

- *SortOutbox*- Stores the sort order of the outbox.
- *Swap out to* – Stores the drive which has been used for the last swap out action.
- *Target AE* – Configuration setting for the DICOM sending.

3.5 Outsourcing data

The database supports outsourcing of data to another harddisc, a network drive or removable, magneto-optical discs. For each location where data is swapped out to, an entry in the database table will be made. For the harddisc-path where data is stored per default there is a special entry with the ID 1. All data, which is not stored in this directory, is handled by the program as *swapped out*. In the table *Drives* you find the type of the media and the path to the storage place. Moreover, for every storage media you will find an entry in the table *discs*. For data which is stored on the harddisc or on a network drive there will be a dummy entry in this table for each path. For MO's and CD-Roms there will be an entry for every disc on which data is stored.

3.5.1 Outsourcing to MO

The table below shows the supported magneto optical media:

Media size [Mbyte]	Media format [Zoll]	Sector size [Bytes]	Number of sides
230	3 ½	512	1
640	3 ½	2.048	1
1.300	3 ½	2.048	1
1.200	5 ¼	512	2
2.600	5 ¼	1.024	2
5.200	5 ¼	2.048	2

Figure 13: Specification of the supported media

Data currently stored on the hard disc (temporarily) is marked with a flag in *Daten.Speichern*. When transferring the data to the MO the corresponding flag is reset and the *Daten.DiskID* changes from 1 (means the main storage directory) to a number greater 1, identifying the place (Disc or other drive) where the data is stored. The *Discs.Label* field contains the label for the disc or path. It starts with "INVIVO" followed by a number starting with one. (*Daten.DiskID* +1).

The application checks the available memory before copying the data. If the MO has no more free space, the MO will be ejected and the user has to insert a new one. If the media is not formatted, the application will do this. The application does not support any other format than FAT.

If a new disk is inserted, the database file "InViVoDB.mdb" will be written to the MO first. This allows recovering of the database in case of a hard disc crash. The MO-handling is done in *TMoVerwaltung* providing the following functions:

- Testing if the desired drive supports removable media
- Testing if a MO is inserted at all.
- Testing if the media writeable.
- Testing if the inserted MO has a FAT file system.
- Format media
- Prevent and allow media removal.

MedCom Client Server Database

- Calculating free disc space.
- Reading volume name.
- Reading drive parameters for optimising file copy.
- Testing if any other application is accessing the MO, because this prevents a media removal.
- Ejecting the MO.

4 Interface with the Main Application

The Main Application (ScanNT, TeleConsult, Exomio, etc.) starts the database during the first access (path of database executable is given in the registry database) by creating a process. If the user leaves the database interface, the window will be hidden until the next access. The communication between the two applications is realized by Windows messages. When the user closes ScanNT a standard window message (WM_CLOSE) is send to the database to close it as well.

The database operates after its initialization in two different modes. The first mode is a loading mode. This means the user wants to load some previously acquired data to the Main Application. So the user has the possibility to select a special dataset. This mode also allows moving data to MO and doing administrative stuff, like adding a new user.

The other mode is called saving mode. In this case the user can store a single dataset to the database. Depending on the kind of data the user has to enter a new patient name or select one out of a list. In this mode the data will be moved to the database and the database application will return control to the Main Application.

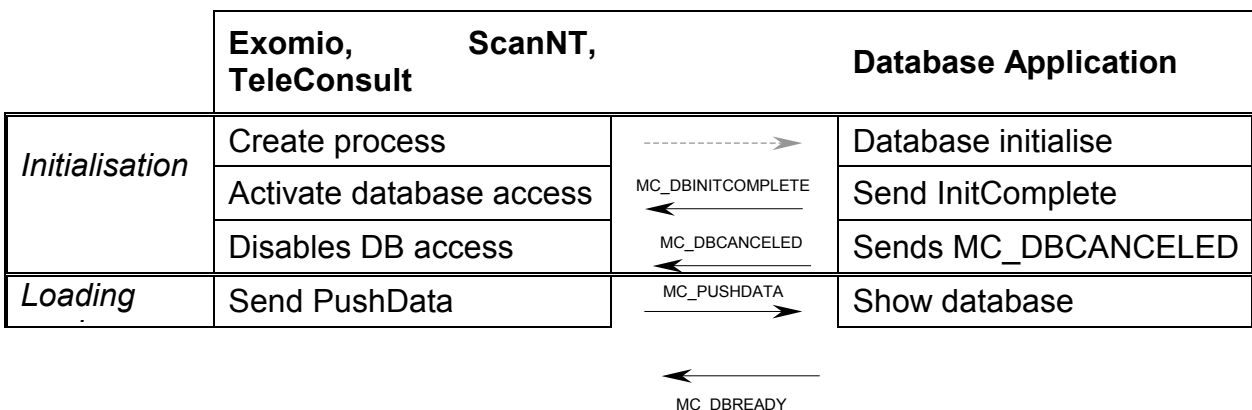
For the physical data transfer the database applications store the selected dataset to a temporary file on hard disc and write the required information (i.e. file selected, filename, etc.) to the registry.

4.1 Windows messages

The messages are defined in both applications. We distinguish now between:

	used messages	defined in module (Exomio, ScanNT, TeleConsult)	defined in class (Database Application)
<i>Initialisation</i>	MC_DBINITCOMPLETE	FILE-IO fileio_init.c	CInViVoTestDoc
<i>User Canceled the login screen</i>	MC_DBCANCELED	FILE-IO fileio_init.c	CInViVoTestDoc
<i>Loading mode</i>	MC_PUSHDATA	FILE-IO fileio_init.c	CMainFrame
	MC_DBREADY	FILE-IO fileio_init.c	CInViVoTestDoc
<i>Saving mode</i>	MC_SAVEDATA	FILE-IO fileio_init.c	CMainFrame
	MC_DBREADY	FILE-IO fileio_init.c	CInViVoTestDoc
<i>Hide Database UI</i>	MC_HIDEDATA	FILE-IO fileio_init.c	CMainFrame
<i>Close applications</i>	WM_CLOSE	Standard windows message	

The communication flow is shown in the figure below:



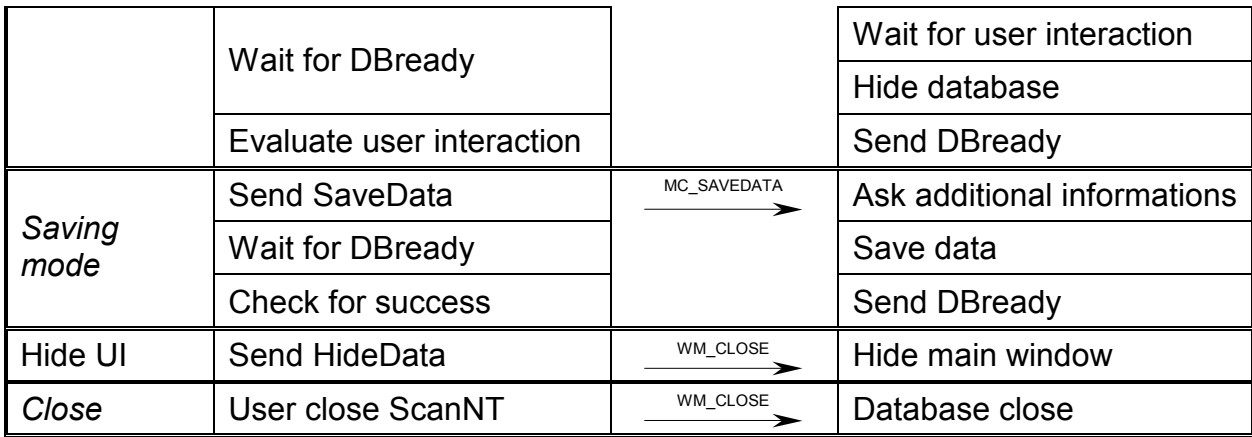


Figure 14: Message flow between the Main Application and the Database Application

During the initialisation, the database writes its main window handler to the registry. By reading this entry the main application is able to send messages directly to the database. This has the advantage that the main application can determine whether the database is still running by checking the access of sending a message. The respond messages are realised as broadcasts.

4.2 Information exchange via Registry

Moreover registry entries are used to exchange information between the database client and the main application (Exomio, ScanNT, TeleConsult, etc.). These information can be found in the registry sub-key *ExChange* of the database which is described in chapter -> 3.4.2.

4.3 Architecture of the Client

The client bases on an MFC (Microsoft Foundation Classes) – Single Document Application for 32Bit-Windows. The classes used in the program are mostly MFC or MFC-derived classes. In one case there is used ActiveX-Control. Where it seems useful normal C-Code is included as well. The following class-tree gives an overview over the classes used in the program together with there MFC-base-classes.

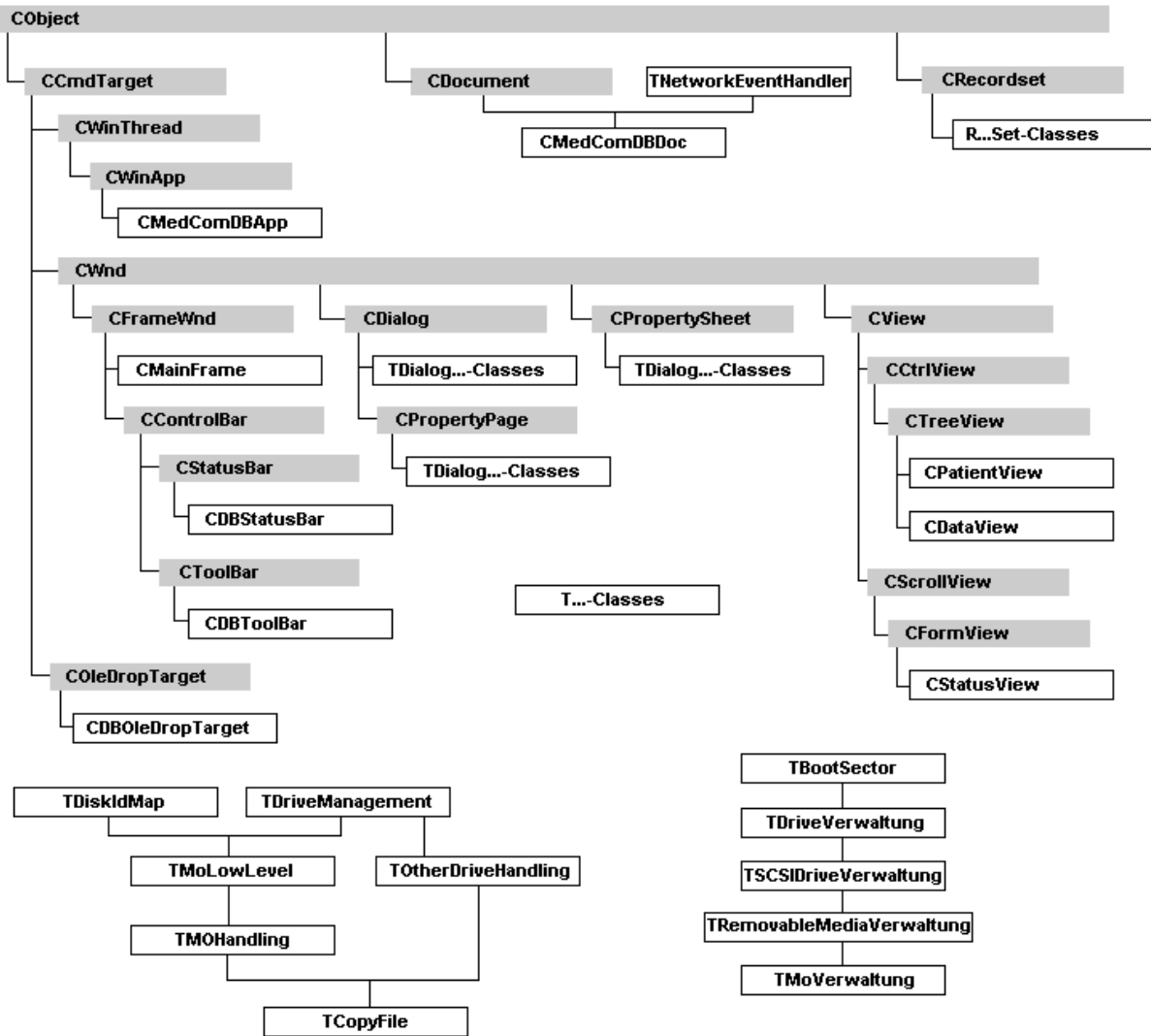


Figure 15: Class-Structure of the Database Client Application

In this figure Classes with a grey background are MFC-Classes. Classes with a white background are MedCom-defined classes. The MedCom-classes can be divided in four main groups of classes:

- Classes named with C... are derived from MFC-classes and are mainly used to implement the GUI of the main window of the database and the application fundament.
- Classes named with TDialog... are derived from the MFC-classes CDialog, CPropertySheet and CPropertyPage and are used to implement dialogs which are occasionally used by the main application.

- Classes named with *R...Set* are used to implement the database access via ODBC-connectivity. These classes are derived from the MFC-class CRecordSet.
- Classes named with *T...* are not derived from an MFC-class and are used to implement different tasks in the background of the application. They are handling for example the access to the MO-drive, the communication with the Client, etc.

4.3.1 C...-classes

The C...-classes defined by MedCom are the following:

- CMedComDBApp
- CMedComDBDoc
- CMainFrame
- CDBStatusBar
- CDBToolBar
- CDBOleDropTarget
- CPatientView
- CDataView
- CstatusView
- Etc.

4.3.2 TDialog...-classes

The *TDialog...*-classes defined by MedCom are the following:

- TDialogAddDrive
- TDialogAnmeldung
- TDialogBitteWarten
- TDialogDicomImport
- TDialogDrivesList
- TDialogEingabeText
- TDialogEinstellungen
- TDialogEinstellungenAnmeldung
- TDialogEinstellungenAnsicht
- TDialogEinstellungenGrund
- TDialogEinstellungenStatusBar
- TDialogEinstellungenViewPat
- TDialogFileIO
- TDialogFilterInbox
- TDialogManagerBenutzer
- TDialogManagerInfoVerwaltung
- TDialogManagerPatienten
- TDialogMessageWindow
- TDialogModifyDrive
- TDialogNetzwerkEinstellungen
- TDialogNeuAnmerkung
- TDialogNeuArzt
- TDialogNeuFunktion
- TDialogNeuPatient
- TDialogNeuUntersuchung
- TDialogNewStudy
- TDialogNewTCMessage
- TDialogNewTCQueryMessage

- TDialogOptionsfeld1
- TDialogOptionsfeld2
- TDialogOptionsfeld3
- TDialogOptionsfelder
- TDialogOutgoingMessages
- TDialogPasswortAendern
- TDialogPatFilterDemographic
- TDialogPatFilterReceiver
- TDialogPatFilterSender
- TDialogScanDisk
- TDialogSelectData
- TDialogSelectDrive
- TDialogServiceEinstellungen
- TDialogShowIngoingMessage
- TDialogShowIngoingQuery
- TDialogShowOutgoingMessage
- TDialogShowOutgoingQuery
- TDialogVerbindungHerstellen
- TDialogWait
- TDialogWaitDicom
- Etc.

4.3.3 R...Set-classes

The R...Set.-classes defined by MedCom are the following:

- RAerzteSet
- RDatenDynaSet
- RDatenSet
- RDiscSet
- RDrivesSet
- RFunktionenSet
- RInformationSet
- RMessagesSet
- ROptionsfeld1Set
- ROptionsfeld2Set
- ROptionsfeld3Set
- RPatientDynaSet
- RPatientenSet
- RQuerySet
- RtestSet
- Etc.

4.3.4 T...-classes

The T...-classes defined by MedCom are the following:

- TAquisitionsManagerVerwaltung
- TBenutzer
- TBootSector
- TCopyFile
- TDatenbankVerwaltung

- TDatenDiscVerwaltung
- TDiskIdMap
- TDriveManagement
- TDriveVerwaltung
- TListCtrlVerwaltung
- TMOHandling
- TMoLowLevel
- TMoVerwaltung
- TNetwork
- TNetworkEventHandler
- TOtherDriveHandling
- TPasswortVerwaltung
- TPatientenVerwaltung
- TRegistryVerwaltung
- TRemovableMedia
- TSCSIDriveVerwaltung
- TStatusVerwaltung
- TStoreScplInterface
- TSwapping
- TTeleServerInterface
- TTimerVerwaltung
- TTreeCtrlVerwaltung
- TUntersuchungsVerwaltung
- TzeitVerwaltung
- Etc.

5 Database Server Functionality

Generally the database application works in two modes: the local mode and the client/server-mode. In the local mode the application accesses the data in the mdb-file directly via ODBC-commands. Therefore a database file need to be installed on the local PC where the program runs. In the client/server-mode the database application works as a client which accesses the *MedCom Database Server* over a connection based on the TCP/IP protocol. Then the data is stored central on the PC where the server is located.

The Database Server implements virtually the same functionality as the Database Client but without any user-interface elements. The server is controlled and responds via dedicated Network Commands between him and the connected clients. The number of connected clients is theoretically limited only by the number of available lines in the network subsystem, however internally we currently limit the number of open lines to 25.

5.1 Client Server Communication

The Protocol (i.e. commands) used to drive the communication between Client and Server is internally defined by MedCom. These commands are mainly:

- Commands to Query the Patient Table, Study Table, Data Table, etc.
- Commands to Download files
- Commands to Maintain the User accounts
- Commands to query the dicom Store-SCP server
- Login Mechanism
- Global Maintenance: Version Checking, Software Update handling, etc.

5.2 Server Registry Entries

The Server uses the key : **HEY_LOCAL_MACHINE\Software\MedCom\DBServer**

Below we will describe the registry values for the Server:

- *AutomaticClientUpdates* – Flag, which indicates that the patient and the file information of the clients shall automatically be updated, if there are changes made through other users.
- *DcmAutomatic* – Indicates, if there shall be an automatic DICOM Import or not.
- *DebugLevel* – Stores the debug level, which influences the output of error information.
- *DebugWindow* – Debug-Window “on” or “off”.
- *Dicom Timeout* – Timeout for DICOM operations.
- *General Timeout* – Timeout for other operations between the client and the server.
- *ListenPort* – Port to listen for connections.
- *Max Connections* – Maximal number of parallel connections.
- *PhonebookFile* – Path to the system phonebook.
- *PhonebookTimeout* – Time a user on a client has to change the phonebook until the operation is aborted.
- *UpdateFile* – If there are different version of Client and Server, an update file can automatically be downloaded. This entry stores the path to the update-file.

5.3 List of Client Server Network Commands

The following commands are used for the communication between the database client and the server:

C-Constant Name	Value	Direction	Comment
GET_PATIENTS	20	C->S	Send back list with patient entries

MedCom Client Server Database

SEND_PATIENT_BLOCKS	21	S->C	Number of blocks for patient list
NO_PATIENTS	22	S->C	No patients available
GET_STUDYS	25	C->S	Send back the studies related to a patient ID
SEND_STUDYS	26	S->C	Respond: Study List
GET_DATA	27	C->S	Get information about all data that is related to a certain patient
SEND_DATA	28	S->C	Reply to GET_DATA
NO_DATA	29	S->C	Reply to GET_DATA
GET_PATIENTSTATUS	30	C->S	Requests all information that is related to a patient ID like name, birthdate, etc.
SEND_PATIENTSTATUS	31	S->C	Reply
NO_PATIENTSTATUS	32	S->C	Error: No data found for this Patient ID
GET_FILESTATUS	33	C->S	Get All data that is related to a file ID
SEND_FILESTATUS	34	S->C	Reply
NO_FILESTATUS	37	S->C	Error: File ID not found
GET_FILEREMARK	36	C->S	Get comment for a file ID
SEND_FILEREMARK	35	S->C	Reply
NO_FILEREMARK	38	S->C	Error: File ID not found
GET_FILE	39	C->S	Client requests to download one file from the server (Used inside of GET_FILES loop)
SEND_FILECOUNT	40	S->C	Reply: Number of files that will actually be transmitted (Tree structure)
SEND_FILESIZE	41	S->C	Number of blocks a file consists
SEND_FILEBLOCK	42	S->C	Send Datablock
NO_FILE	43	S->C	Error: File-ID unknown
GET_FILES	44	C->S	Clients requests to receive all files that are related to a file ID (i.e. volumes+plan)
GET_FILEBLOCK	45	C->S	Client is ready to receive next block
LOGIN	46	C->S	Login attempt with password
LOGIN_ANSWER	47	S->C	Answer to login attempt (OK, Unknown, Wrong Passord, etc.)
GET_FILETYPE	48	C->S	Requests the filetype related to a table entry
SEND_FILETYPE	49	S->C	Reply
SEND_FILEDATA	50	C->S	Sends file data from client to server
SEND_FILEID	51	S->C	Reply: New file ID in data table
NO_MORE_FILES	52	C->S	No more files to be transferred from client to server
NEXT_FILE	53	S->C	Server requests next file in a row
GET_OFIELDDATA	54	C->S	Clients requests information about the optional fields
SEND_OFIELDSIZE	55	S->C	Number of blocks for optional fields data
SEND_OFIELDEEMPTY	56	S->C	Reply: This optinal field is empty
GET_OFIELDBLOCK	57	C->S	Requests next block with field data
SEND_OFIELDBLOCK	58	S->C	This is the next block with field data
CREATE_PATIENT	59	C->S	Client sends data to create new entry in patient table
PATIENT_CREATED	60	S->C	Reply: Patient created
UPDATE_PATIENT	61	C->S	Change data of a known patient
UPDATE_SUCCESSFUL	62	S->C	Reply: OK
PATIENT_DONT_EXISTS	63	S->C	Reply: Error patient-ID not found
DELETE_PATIENT	64	C->S	Client requests to delete a patient, with all data
PATIENT_DELETED	65	S->C	Reply: No error
CAN_ACCESS	66		?? (Obsolete)
CAN_ACCESS_TRUE	67		Obsolete
CAN_ACCESS_FALSE	68		Obsolete

MedCom Client Server Database

GET_FILESIZE_IN_BYTES			Obsolete
SEND_FILESIZE_IN_BYTES			Obsolete
GET_COMMENT	71	C->S	Get comment related to a file
SEND_COMMENT	72	S->C	Reply
DELETE_STUDY	73	C->S	Delete a study
STUDY_DELETED	74	S->C	Reply: OK, deleted
STUDY_NOT_FOUND	75	S->C	Reply: Study not found
GET_COMMENT_TO_CHANGE	77	C->S	Get a comment in order to change that
SEND_COMMENT_TO_CHANGE	78	S->C	Reply
CHANGE_COMMENT	79	C->S	Data to change the comment of a file
COMMENT_NOT_CHANGED	80	S->C	Could not change the comment
DELETE_FILE	81	C->S	Delete a file and all related files
FILE_DELETED	82	S->C	Reply: OK
FILE_DOESNT_EXIST	83	S->C	Reply: File not found
GET_DOCTORS	84	C->S	Request: Get list with Accounts on the database
SEND_DOCTORS	85	S->C	One block with account data
NO_DOCTORS_FOUND	86	S->C	Error: No accounts (Except the internal SuperUser, hopefully)
GET_DICOM_PATIENTS	87	C->S	Request: Get list with dicom patients
SEND_DICOM_PATIENT_NUMBER	88	S->C	Number of blocks with patient data
DICOM_IMPORT_ERROR	89	S->C	Error while retrieving dicom data
GET_NEXT_DICOM_PATIENT_BLOCK	90	C->S	Client is ready to receive next Block
SEND_DICOM_PATIENT_BLOCK	91	S->C	Dicom block with patient data
GET_DICOM_STUDYS	92	C->S	Client requests Dicom Studies related to a patient
SEND_DICOM_STUDY_NUMBER	93	S->C	Reply: Number of Studies related to this patient
GET_NEXT_DICOM_STUDY_BLOCK	94	C->S	Client requests a Study Block (in a transfer loop)
SEND_DICOM_STUDY_BLOCK	95	S->C	Study Block data
GET_DICOM_SERIES	96	C->S	Client requests all series related to a dicom study
SEND_DICOM_SERIES	97	S->C	Reply:List with series data (Key and Modality)
GET_DICOM_PATIENT_STATISTIC	98	C->S	Client requests detailed data about a dicom patient
SEND_DICOM_PATIENT_STATISTIC	99	S->C	Reply
CONNECT_ACCEPTED	100	S->C	Respond to a connect attempt: Client Connection accepted
CONNECT_REFUSED	101	S->C	Respond to a connect attempt: Client Connection refused (all lines busy)
DICOM_IMPORT_PATIENT	102	C->S	Command: Move this Patientdata from Dicom Buffer into Database
DICOM_IMPORT_STUDY	103	C->S	Command: Move this study from Dicom Buffer into Database
DICOM_IMPORT_SERIE	104	C->S	Command: Move this series from Dicom Buffer into database
DICOM_IMPORT_COMPLETE	105	S->C	Last Dicom Import request successfully processed
DICOM_IMPORT_NOT_COMPLETE	106	S->C	Error while importing dicom data
DICOM_DATA_REFRESHED	107	S->C	Dicom data has been re-loaded from Store-SCP server
REFRESH_DICOM_DATA	108	C->S	Client requests to re-load dicom stuff
GET_DICOM_STUDY_STATISTIC	109	C->S	Client requests study details (i.e. number of

MedCom Client Server Database

			series)
SEND_DICOM_STUDY_STATISTIC	110	S->C	Reply
GET_DICOM_SERIE_STATISTIC	111	C->S	Client request details about a serie
SEND_DICOM_SERIE_STATISTIC	112	S->C	Reply
DICOM_START	113	C->S	Client request temporary ownership to dicom (Note: This will be removed soon)
DICOM_STOP	114	C->S	Client releases ownership of dicom
DICOM_BUSY	115	S->C	Dicom not available: Allready in use
DICOM_NOT_BUSY	116	S->C	OK, dicom is owned now by this client
GET_OPTION_FIELDS	117	C->S	Get names of the 3 optional fields
SEND_OPTION_FIELDS	118	S->C	Reply
SERVER_DOWN	119	S->C	The line is about to be closed
PREVIEW_DICOM	120	C->S	Client requests to receive a file for dicom preview functionality
DICOM_PREVIEW_ERROR	121	S->C	Reply: An error occurred while processing the dicom preview
DICOM_PREVIEW_FILE_READY	122	S->C	The server is now ready with dicom preview preparation
GET_DICOM_FILEBLOCK	124	C->S	Client requests dicom file block
SEND_DICOM_FILEBLOCK	123	S->C	Server sends dicom file block
CREATE_NEW_OPTION	125	C->S	Create new entry in optional fields
NEW_OPTION_CREATED	126	S->C	Reply: OK, no error
CHANGE_OPTION	127	C->S	Change the value of an optional field
OPTION_CHANGED	128	S->C	Reply: Ok, no error
DELETE_OPTION	129	C->S	Delete an optional field entry
OPTION_DELETED	130	S->C	Reply: OK, no error
SET_OPTION_FIELDS	131	C->S	Change names of optional field table
OPTION_FIELDS_SETTED	132	S->C	Reply: Ok, no error
GET_FUNCTIONS	133	C->S	Get list with user functions
SEND_FUNCTIONS	134	S->C	Reply: List with functions
NO_FUNCTIONS_FOUND	135	S->C	No functions found
CREATE_NEW_FUNCTION	136	C->S	Create a new user function in table
NEW_FUNCTION_CREATED	137	S->C	New function has been created
DELETE_FUNCTION	138	C->S	Delete a function
FUNCTION_DELETED	139	S->C	Function deleted
GET_FUNCTION_DATA	140	C->S	Get data related to a function entry
SEND_FUNCTION_DATA	141	S->C	Reply
NO_FUNCTION_DATA_FOUND	142	S->C	Reply: No function data found
CHANGE_FUNCTION	143	C->S	Client request to change an function entry
FUNCTION_CHANGED	144	S->C	Reply: Ok, no error
FUNCTION_NOT_CHANGED	145	S->C	Reply: Failed to change this function
DELETE_DOCTOR	146	C->S	Delete an user entry
DOCTOR_NOT_DELETED	147	S->C	Failed to delete user account
DOCTOR_DELETED	148	S->C	OK, account deleted
GET_DOCTOR	149	C->S	Client requests data of an account
SEND_DOCTOR	150	S->C	Reply (User data)
DOCTOR_NOT_FOUND	151	S->C	Reply: User entry not found
CHANGE_DOCTOR	152	C->S	Client asks to change an account
DOCTOR_CHANGED	153	S->C	Reply: Entry has been changed
DOCTOR_NOT_CHANGED	154	S->C	Reply: Account not found
NEW_DOCTOR	155	C->S	Clients requests to create a new user entry
NEW_DOCTOR_CREATED	156	S->C	Reply: OK no error
CHANGE_PASSWORD	157	C->S	Change the password of an account
PASSWORD_CHANGED	158	S->C	Reply: Password changed
PASSWORD_NOT_CHANGED	159	S->C	Reply: Failed to change password

MedCom Client Server Database

FUNCTION_NOT_DELETED	160	S->C	Error message in function handling
FILE_NOT_FOUND	161	S->C	Error message in file download
DATA_ON_MO	162	S->C	The requested file is on MOD disk
CHECK_FOR_LOGIN	163	C->S	Check if a login name is allready occupied.
CHECK_FOR_LOGIN_ANSWER	164	S->C	Reply
SEND_STATISTIC	165	S->C	Server statistic
DICOM_STEP	166	S->C	Dicom Import progress step (10 %)
PATIENT_NOT_DELETED	167	S->C	The patient could not be deleted
GET_PHONEBOOKENTRYS	168	C->S	Client requests the names of the phonebook-entrys from the server
SEND_PHONEBOOKENTRYS	169	S->C	Reply: List with phonebook entries
NO_PHONEBOOK_FOUND	170	S->C	Error: Phonebook not found on server side
CREATE_NEW_STUDY	171	C->S	Create a new study in the study table
NEW_STUDY_CREATED	172	S->C	Reply: Study created
STUDY_CREATION_FAILED	173	S->C	Reply: Failed to create study (Probably the patient doesn't exists)
NO_STUDY_DATA_FOUND	174	S->C	Reply to GET_STUDYS: No study found
STUDY_ON_MO	175	S->C	The study is swapped out
GET_STUDY_DESCRIPTION	176	C->S	Get the study comment from server
SEND_STUDY_DESCRIPTION	177	S->C	Reply
CHANGE_STUDY	178	C->S	Client requests to change a study
STUDY_CHANGED	179	S->C	Reply: No error
NO_STUDY_FOUND	180	S->C	Reply: Error – study not found
GET_DOCTORS_ADDRESSES	181	C->S	Tele: Get account address fields
SEND_DOCTORS_ADDRESSES	182	S->C	Reply: Send the address values of the doctors
NEW_MESSAGE	183	C->S	Tele: Creating of a new message
NEW_MESSAGE_ANSWER	184	S->C	Reply: Answer to the NEW_MESSAGE-request
GET_MESSAGES	185	C->S	Tele: Get a number of messages, which is defined through a filter
SEND_MESSAGES	186	S->C	Reply: Answer to GET_MESSAGES, sends the requested message data
SEND_MESSAGE_BLOCK	187	S->C	Sends a block of max. 100 messages
NO_MESSAGES	188	S->C	Reply: No messages for the specified filter found
GET_MESSAGE	189	C->S	Tele: Get the data of a message from the server
SEND_MESSAGE	190	S->C	Reply: Sends the requested message data
SEND_MESSAGE_ERROR	191	S->C	Reply: An error occurred during the reading of the message data
MESSAGE_READED	192	C->S	Tele: The message has been read, changes the new flag
GET_REFMESSAGE_ID	193	C->S	Tele: Get the ID of the referenced message
SEND_REFMESSAGE_ID	194	S->C	Reply: Sends the ID of the related message
CHANGE_MESSAGE	195	C->S	Tele: Change the attributes of an existing message
DELETE_MESSAGE	196	C->S	Tele: Delete a message from the database
MESSAGE_DELETED	197	S->C	Reply: Answer to the delete request
CANCEL_MESSAGE	198	C->S	Tele: Cancels the transfer of a message
MESSAGE_CANCELED	199	S->C	Reply: Answer to a cancel request
CHECK_MESSAGES	200	C->S	Tele: Checks if new messages have been arrived
CHECK_MESSAGES_ANSWER	201	S->C	Reply to the CHECK_MESSAGES-request
SEND_MESSAGE_AGAIN°	202	C->S	Tele: Request to reactivate a message
SEND_MESSAGE_AGAIN_ANSWER	203	S->C	Reply to the previous request
UPDATE_SENDED_MESSAGES	204	C->S	Tele: The TeleServer-Interface shall update

MedCom Client Server Database

			the send messages
UPDATE_RECEIVED_MESSAGES	205	C->S	Tele: The TeleServer-Interface shall update the received messages
MESSAGE_STATE_CHANGED	206	S->C	Tele: The state of a message has been changed
NEW_MESSAGE_RECEIVED	207	S->C	Tele: A new message has been received
SEND_NEW_AUDIO_FILE	208	C->S	Tele: Sending a new audio file for an existing ID
GET_NEW_AUDIO_FILE	209	S->C	Reply: Command to get the data of the new audio file
GET_TEMP_MESSAGE_SIZE	210	C->S	Tele: Gets the temporary message size, according to a given filelist
SEND_TEMP_MESSAGE_SIZE	211	S->C	Reply to the previous command
DICOM_PATIENT_CHECK	212	C->S	Checks if a patient with a given name or Dicom ID already exists
DICOM_PATIENT_RES	213	S->C	Reply to the previous command
GET_EXAMINATION_DATE	214	C->S	Gets the examination date of a given file
SEND_EXAMINATION_DATE	215	S->C	Reply to the previous command
DICOM_SEARCH_ID	216	C->S	Proves if a Dicom ID already exists in the database
DICOM_SEARCH_ID_RES	217	S->C	Reply to the previous command
DICOM_DELETE_ENTRY	218	C->S	Deleting a patient from the Dicom-Interface
DICOM_DELETE_FINISHED	219	S->C	Reply to the previous command
GET_PHONEBOOK	220	C->S	Gets the phonebook-file from the server
SEND_PHONEBOOK	221	C->S	Sending the phonebook file back to the server
PHONEBOOK_IN_USE	222	S->C	Another user is modifying the phonebook
PHONEBOOK_TIMED_OUT	223	S->C	The phonebook-usage-time has timed out, therefore the phonebook cannot be written back
PHONEBOOK_OKAY	224	S->C	The server is ready to receive the phonebook
STOP_SEND_FILES	225	C->S	The client cancels the file transfer process
UPDATE_SENDED_MESSAGES_FINI	226	S->C	Tele: The update send messages-command is finished
GET_STUDY_VALUES	227	C->S	Getting the values of a study with a specified ID
SEND_STUDY_VALUES	228	S->C	Reply to the previous command
GET_FILE_DATA	229	C->S	Get the properties of a file
SEND_FILE_DATA	230	S->C	Reply to the previous command
COMPARE_PATIENT	231	C->S	Seeks for a patient with specified properties in the database and gives a response, if a similar one exists
PATIENT_COMPARE_ANSWER	232	S->C	Reply to the previous command
INSERT_STUDY_FROM_LOCAL	233	C->S	Inserts a study from the database on the client into the database on the server
PROVE_FOR_FILE_EXISTS	234	C->S	Proves if a file with a given ParentID, StudyID and TeleConsultObjectID exists in the database
ANSWER_FILE_EXISTS	235	S->C	Reply to the previous command
ANSWER_FILE_EXISTS_ERROR	236	S->C	An error occurred during the PROVE_FOR_FILE_EXISTS-command
CASE_LIST_QUERY	237	C->S	Send back list of cases -- used by Simulator UI (a special main application)
CASE_LIST_NUM	238	S->C	Send number of items in the case list
CASE_LIST_ITEM	239	S->C	Sends an item of the case list
GET_UPDFILE	240	C->S	Requests updates for the installation of the database application
GET_UPDFILE_ANSWER	241	S->C	Reply from the server to the update request

MedCom Client Server Database

LICENCE_QUERY	250	MA->S	Licence query from the main application to the server
GET_PREVIEW_FILE	251	C->S	Client requests the preview file of an data entry
NO_PREVIEW_FILE	252	S->C	Server Answer: No preview file available
FILE_INCONSISTENT	253	S->C	A file which shall be sent has inconsistent file information
FILE_INCONSISTENT_ANSWER	254	C->S	Answer to the file inconsistent message
APPROVE_PLAN	255	C->S	Request to approve a plan
APPROVE_PLAN_SUCCEEDED	256	S->C	Approving plan succeeded
APPROVE_PLAN_FAILED	257	S->C	Approving of plan failed
UNAPPROVE_PLAN	258	C->S	Request to unapprove a plan
UNAPPROVE_PLAN_SUCCEEDED	259	S->C	Unapproving plan succeeded
UNAPPROVE_PLAN_FAILED	260	S->C	Unapproving of plan failed
GET_FILEID_BY_TCID	261	C->S	Query for File-ID with TeleConsultID
SEND_FILEID_BY_TCID	262	S->C	Send File-ID out of TCID
CHECK_SIMULATION_PLAN_APPROVED	263	C->S	checks if a simulation plan is approved or not
SIMPLAN_APPROVED_ANSWER	264	S->C	answer to the above request
BC_NEW_PATIENT	265	S->C	Broadcast: a new patient has been created
BC_CHANGE_PATIENT	266	S->C	Broadcast: a patient has been changed
BC_DELETE_PATIENT	267	S->C	Broadcast: a patient has been deleted
BC_NEW_STUDY	268	S->C	Broadcast: a study has been created
BC_CHANGE_STUDY	269	S->C	Broadcast: a study has been changed
BC_DELETE_STUDY	270	S->C	Broadcast: a study has been deleted
BC_NEW_FILE	271	S->C	Broadcast: a file has been created
BC_DELETE_FILE	272	S->C	Broadcast: a file has been deleted
GET_ATTACHMENT_FILE	273	S->C	sended from the server , when it is ready to receive attachment packages
SEND_ATTACHMENT_BLOCKNUMBER	274	C->S S->C	sending the number of attachment blocks
SEND_ATTACHMENT_BLOCK	275	C->S S->C	sending a block of the attachment file
ATTACHMENT_CREATED	276	S->C	the attachment has been successfully created on server-side
GET_ATTACHMENT	277	C->S	Get the attachment file from the server
BC_APPROVEPLAN	278	S->C	Broadcast: a plan has been approved
BC_UNAPPROVEPLAN	279	S->C	Broadcast: a plan has been unapproved
FILE_STORE_CONFIRMATION	280	S->C	Confirmation that a file has been stored through the server
DICOM_STUDY_CHECK	281	C->S	CHange the Name or ID of a patient and/or create a dicom-study for him
DICOM_STUDY_CHECK_SUCCESS	282	S->C	Study Check successful studyid returned
DICOM_STUDY_CHECK_ERROR	283	S->C	Study Check not successful errorNumber returned
NEW_STUDY_TYPE	284	C->S	Create a new Study Type
NEW_STUDY_TYPE_SUCCESS	285	S->C	Creation of new Study Type successful
NEW_STUDY_TYPE_ERROR	286	S->C	Failed to create a new Study Type
GET_STUDYTYPES	287	C->S	requests the list of study types from the server
SEND_STUDYTYPES_SUCCESS	288	S->C	Sends the list with the study types
SEND_STUDYTYPES_ERROR	289	S->C	an error occured during the study types request
GET_SINGLESTUDYTYPE	290	C->S	get the values for a single study type from the server
GET_SINGLESTUDYTYPE_SUCCESS	291	S->C	successful answer with the values for the study type

GET_SINGLESTUDYTYPE_ERROR	292	S->C	error during single study type request
MODIFY_STUDY_TYPE	293	C->S	Modify-Study-Type request to the server
MODIFY_STUDY_TYPE_SUCCESS	294	S->C	Successful modification of the study-type
MODIFY_STUDY_TYPE_ERROR	295	S->C	Error during the modification of the study-type
DELETE_STUDY_TYPE	296	C->S	Command to delete a study type
DELETE_STUDY_TYPE_SUCCESS	297	S->C	Successful deletion of a study type
DELETE_STUDY_TYPE_ERROR	298	S->C	Error during deletin of a study type
LICENCE_ANSWER	150	S->MA	Reply to the licence query, sent from the server directly to the main-application

5.4 Internal Software Architecture

The Server bases on an MFC (Microsoft Foundation Classes) – Dialog based Application for 32Bit-Windows. As in the Client the classes used in the program are mostly MFC or MFC-derived classes and where it seems useful normal C-Code is included as well. The following class-tree gives an overview over the classes used in the program together with there MFC-base-classes.

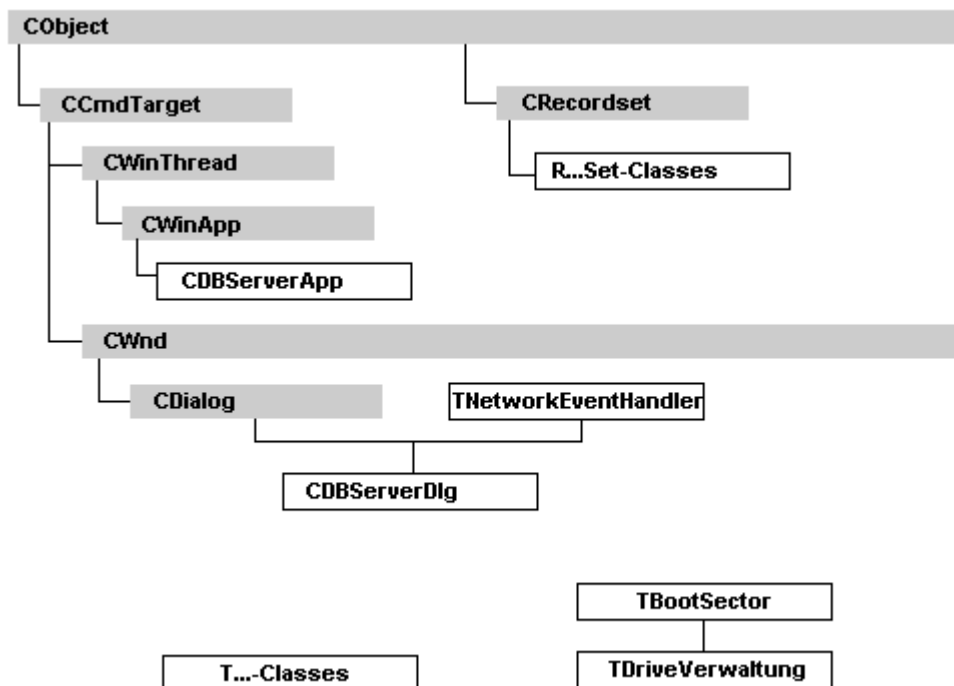


Figure 16: Class structure of the Database Server Application

The Server uses four kind of MedCom defined classes:

- Classes named with C... are derived from MFC classes and are used to implement the GUI of the main dialog and the application fundament.
- In the moment one class named with *TDialog...* (TDialogWait), which is derived from the class CDialog and implements an additional dialog during the database update.
- Classes named with *R...Set* are used to implement the database access via ODBC-connectivity. These classes are derived from the MFC-class CRecordSet.
- Classes named with *T...* are not derived from an MFC-class and are used to implement different tasks in the background of the application. They are handling for example the access to the MO-drive, the communication with the Client, the DICOM import, etc.

5.4.1 C...-Classes

The C...-classes defined by MedCom for the Server are the following:

- CDBServerApp
- CDBServerDlg
- Etc.

5.4.2 TDialog...-Classes

The TDialog...-classes defined by MedCom for the Server are the following:

- TDialogWait

5.4.3 R...Set-Classes

The R...Set-classes defined by MedCom for the Server are the following:

- RAerzteSet
- RDatenDynaSet
- RDatenSet
- RDiscSet
- RDrivesSet
- RFunktionenSet
- RInformationSet
- RMessagesSet
- ROptionsfeld1Set
- ROptionsfeld2Set
- ROptionsfeld3Set
- RPatientDynaSet
- RPatientenSet
- RPatientStudysSet
- RquerySet
- Etc.

5.4.4 T...-Classes

The T...-classes defined by MedCom for the Server are the following:

- TBenutzer
- TBootSector
- TConnectionField
- TDicomImport
- TDiskIdMap
- TDriveManagement
- TDriveVerwaltung
- TNetwork
- TNetworkEventHandler
- TPasswortVerwaltung
- TRegistryVerwaltung
- TStoreScpInterface
- TTeleServerInterface
- TTimerVerwaltung
- TUntersuchungsVerwaltung
- TzeitVerwaltung
- Etc.

6 MedCom Dicom StoreSCP

The MedCom StoreSCP (*Dicom C-Store Service Provider*) application is used to receive and process Dicom C-STORE requests on an open TCP/IP line. The Dicom C-STORE command is used by medical devices to send dicom data (like images, RT-Plan, etc.) to an external modality. The MedCom StoreSCP can receive the data and stores them into a file on the harddisk. The images are queried by dedicated DDE commands between the StoreSCP application and the MedCom Database Client (or Server).

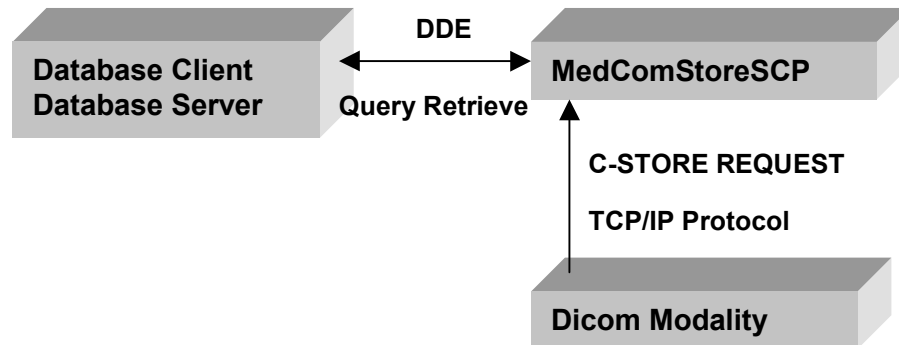


Figure 17: Dataflow between StoreSCP, Database and Dicom Modality

The StoreSCP application maintains an index list in order to buffer incoming slices until they are retrieved (and probably deleted) via DDE command.

6.1 Dicom Communication

The StoreSCP application opens an TCP port and waits for incoming connections. According to the Dicom Standard (please refer to the Dicom Standard for details) the communication is performed in three parts:

- **Dicom Association Request**
 During the association phase the two modalities negotiate the used transfer protocol and transfer syntaxes. The dicom standard defines the term "Abstract Syntax" and "Transfer Syntax". The Abstract Syntax defines Modalities (like CT, MR) and in particular *Information Models*. The current implementation of StoreSCP rejects any association requests with the following Abstract Syntax:
 - "1.2.840.10008.5.1.4.1.2.1.1" : *Patient Root Query/Retrieve Information Model FIND*
 - "1.2.840.10008.5.1.4.1.2.1.2" : *Patient Root Query/Retrieve Information Model MOVE*
 - "1.2.840.10008.5.1.4.1.2.1.3", *Patient Root Query/Retrieve Information Model GET*
 - "1.2.840.10008.5.1.4.1.2.2.1", *Study Root Query/Retrieve Information Model FIND*
 - "1.2.840.10008.5.1.4.1.2.2.2", *Study Root Query/Retrieve Information Model MOVE*
 - "1.2.840.10008.5.1.4.1.2.2.3", *Study Root Query/Retrieve Information Model GET*
 - "1.2.840.10008.5.1.4.1.2.3.1", *Patient/Study only Query/Retrieve Information Model FIND*
 - "1.2.840.10008.5.1.4.1.2.3.2", *Patient/Study only Query/Retrieve Information Model MOVE*
 - "1.2.840.10008.5.1.4.1.2.3.3", *Patient/Study only Query/Retrieve Information Model GET*
 - "1.2.840.10008.5.1.4.31", *Modality Worklist Information Model FIND*

Since the StoreSCP application doesn't parse or deals with the data directly the StoreSCP can accept virtually any proposed Dicom Transfer Syntax. However, since the Dicom Standard appears to rather difficult to implement for some Device Manufactures, it turns out to be reasonable to restrict the used Transfer Syntax to a dedicated value (for instance: Accept only Little Endian Transfer Syntax for CT Modality). This artificial restriction is invoked inside line configuration.

- **Dicom Message**
 After the Dicom Association is successfully established the StoreSCP application waits for any Dicom Message Request. Currently only two Dicom Messages are implemented:
 1. **C-STORE Request**
 On receive of such a Message the StoreSCP will create a new dicom file, create and store a Part10 Header in this file and stores all dicom data in this file. The StoreSCP application currently does not change or affect the used transfer syntax thus, the resulted dicom object contains the same dicom transfer syntax.
 2. **C-ECHO Request**
 This has been implemented for testing purposes – some devices use this Dicom Message to verify the Dicom Connection. StoreSCP just sends a Dicom C-ECHO-RSP message back.
- **Association Release**
 The dicom connection is usually closed after receive of an dicom Association Release command – StoreSCP responds with an `AS_RELEASE_RSP` command

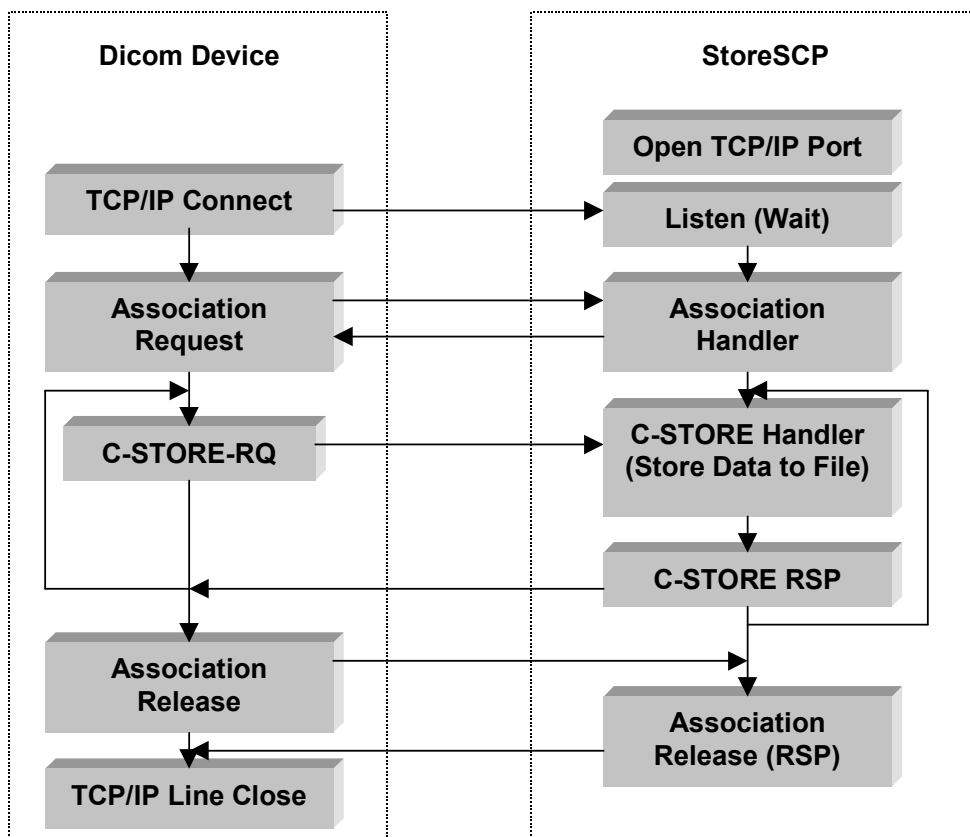


Figure 18: Dicom C-STORE Workflow

The Dicom UID used and defined by MedCom GmbH for the StoreSCP is defined as follows:

"1.2.276.0.55.0.0.1.0.6"

The last for numbers (0.1.0.6) represents the version number of StoreSCP (1.0.6)

6.2 Connection to the Database

The communication between the StoreSCP application and the MedCom database is done via DDE message exchange. The StoreSCP act in this case like a *DDE-Server*. The defined DDE Commands are described in table 18.

DDE Command (String)	Initial Direction	Description
"QPat"	C->SCP	Query on Patients: StoreSCP will respond with a list (See fig. 19) of all patients currently in the StoreSCP buffer.
"QStuNN"	C->SCP	Query on Study: StoreSCP will respond with a list of all studies related to this patient ID (NN).
"QSerNN"	C->SCP	Query on Series: StoreSCP will respond with a list of all series related to this study ID (NN).
"QImgNN"	C->SCP	Query on Images: StoreSCP will respond with a list of all images related to this series ID (NN).
"DPat" (+ ID)	C->SCP	Delete Patient with ID
"DStu" (+ ID)	C->SCP	Delete Study with ID
"DSer" (+ ID)	C->SCP	Delete Serie with ID
"DImg" (+ ID)	C->SCP	Delete Image with ID
"QREL"	C->SCP	Query Release: StoreSCP will respond with Release Code
"STAT"	SCP->C	StoreSCP informs about network state change: Idle vs. Busy

Figure 19: DDE Commands

The name of the DDE Server established by StoreSCP is "MedComSCP". The topic used in DDE commands is called "ScpQuery".

```

/** Patient Data used by MedCom Database */
/** These values are taken directly from */
/** Dicom Tags.          */
typedef struct _DB_PAT_DATA
{
    char    Name[64];
    char    DicomID[64];
    char    BirthDate[32];
    char    Sex[32];
    char    Comment[64];
}DB_PAT_DATA;

/** Study Data used by MedCom Database */
/** These values are taken directly from */
/** Dicom Tags.          */
typedef struct _DB_STUDY_DATA
{
    char    Date[32];
    char    Description[64];
}DB_STUDY_DATA;

/** Series Data used by MedCom Database */
/** These values are taken directly from */
/** Dicom Tags.          */
typedef struct _DB_SERIES_DATA
{
    char    Description[64];
    char    Modality[16];
}DB_SERIES_DATA;

/** Image Data used by MedCom Database */
/** These values are taken directly from */
/** Dicom Tags.          */
typedef struct _DB_IMAGE_DATA
{
    char    filename[MAX_PATH];
}DB_IMAGE_DATA;

/*****

typedef struct _ScpPatEntry
{
    unsigned long    key;          /** ID Inside StoreSCP server */
    DB_PAT_DATA      data;        /** Dicom Data          */
}ScpPatEntry;

typedef struct _ScpStudyEntry
{
    unsigned long    key;          /** ID Inside StoreSCP server */
    DB_STUDY_DATA    data;        /** Dicom Data          */
}ScpStudyEntry;

typedef struct _ScpSerieEntry
{
    unsigned long    key;          /** ID Inside StoreSCP server */
    DB_SERIES_DATA    data;        /** Dicom Data          */
}ScpSerieEntry;

typedef struct _ScpImageEntry
{
    unsigned long    key;          /** ID Inside StoreSCP server */
    DB_IMAGE_DATA    data;        /** Dicom Data          */
}ScpImageEntry;

```

Figure 20: Data Structures used in Query Commands

In order to load dicom slices the MedCom Database (Client or Server) sends a query request to the StoreSCP application. This query is responded with list of patient (in case that there is dicom data). In order to load a particular patient study (i.e. a number of single slices) a query is transferred to the StoreSCP application accordingly. After getting the filenames of an patient study the MedCom database compresses (lossless) the dicom slices in order to save disk space and generates a single file out of the slice sequence. When loading on to the main

application, the dicom file (SLI-Format) is decompressed into single dicom (original) files and loaded.

File Offset (bytes)	Contents	Description
0x0	String: "DCDB"	Identifier
0x4	String: "0002"	File Version number
0x8	Unsigned long	Number of Dicom files
0xC	Starting of Image Sequence: 0 Byte: Tag-Number 0 Byte: Datasize that follows N Bytes: Data Implemented Tags: 0x0004 : Compressed Dicom File 0x0006 : Uncompressed File 0xffff: End of Sequence Marker	Arbitrary number tags that can follow.

Figure 21: Internal Dicom SLI Format

If you import data from the StoreSCP to the MedCom Database the following DICOM-tags will be converted in Database entries:

DICOM-Field	Group	Element	MedComDB-Table	MedComDB-Entry
PATIENTS_NAME	0x0010	0x0010	Patienten	Vorname
PATIENTS_NAME	0x0010	0x0010	Patienten	Nachname
PATIENTS_SEX	0x0010	0x0040	Patienten	Geschlecht
PATIENTS_BIRTH_DATE	0x0010	0x0030	Patienten	Geburtsdatum
PATIENT_ID	0x0010	0x0020	Patienten	DCID
STUDY_DESCRIPTION	0x0008	0x1030	PatientStudys	Description
STUDY_DATE	0x0008	0x0020	PatientStudys	StudyBegin
STUDY_DATE	0x0008	0x0020	PatientStudys	StudyUpdate
SERIESDESCRIPTION	0x0008	0x103E	Daten	Anmerkung
MODALITY	0x0008	0x0060	Daten	Typ
MODALITY	0x0008	0x0060	Daten	Dateiendung
STUDY_DATE	0x0008	0x0020	Daten	AufnahmeDatum
STUDY_DATE	0x0008	0x0020	Daten	LetzterZugriffDatum

6.3 Registry Entries

The Main Registry Path used by the StoreSCP application is:

HKEY_LOCAL_MACHINE\SOFTWARE\MedCom\MCStoreSCP

In the following table all StoreSCP related registry entries are described:

Registry Entry	Type	Description
InstallPath	String	Contains the path of the installation
ImagePath	String	Folder where StoreSCP will write the image files
LogfilePath	String	Folder where StoreSCP will write logfiles.
DebugLevel	DWORD	Level Debug Output: Default: 0
LogWin	String	"on"/"off": Enable/Display of debug window
Line0Port	DWORD	TCP/IP Port for Line 0
Line0TSyntax	DWORD	Proposed Transfer Syntax (TS) during Association: 0 -> Take the first proposed TS 1 -> Restrict to <i>Implicit Little Endian</i> TS 2 -> Restrict to <i>Explicit Little Endian</i> TS 3 -> Restrict to <i>Explicit BIG Endian</i> TS 4 -> Restrict to <i>JPEG Baseline</i> TS 5 -> Restrict to <i>JPEG Extended</i> TS 6 -> Restrict to <i>RLE</i> TS
LineNPort	DWORD	TCP/IP Port for Line N (Max Lines: 5)
LineNTSyntax	DWORD	See above: TS for line N (Max Lines: 5)

Figure 22: Registry Entries

6.4 Code Structure

The StoreSCP is a Win32 Application written in standard C language. It consists of the following source code files:

1. **main.c**
Main Function: Initialisation, Main Event Loop, etc.
2. **dcm_tools.c**
Dicom Association Handler (Func: 'DicomAssociationHandler')
Dicom Message Handler (Func: 'DicomMessageReceiveHandler')
Dicom Part 10 Writer (Func: 'CreatePart10Header')
Dicom Parser to fill the internal patient list (Func: 'QueryDicomElements')
3. **net_tools.c**
Network Sub Routines (Sending, Receiving, Server-Part, Worker-Thread, etc.)
4. **reg_tools.c**
Registry Sub Routines.
5. **Fprintf.c**
Debug output handler

7 Verification / Test Cases

In this chapter we describe some potential risk related problems and especially procedures to verify the database implementation.

7.1 Hazard: Error in Saving

Cause: MO-Disc full

Datasets will not be stored if the free space available on the MO-Disc is smaller than the size of the dataset (plus the additional batch and or mask files, which may have to be stored also). Test by saving few data sets on MO-disc and leaving not enough space for an additional data set. Then try to save this data set. The MedCom database checks the free available space on MO-disc and compares it with the total file size of the data sets to be stored. If not enough space is available on the MO-Disc then it is ejected from the drive.

A unique Label number is produced and the user is asked to write this label on the etiquette of a new formatted MO-Disc and to insert it into the drive.

Cause: MO-Handling failed

The MOD handling should be checked in any case, by generating data in the database until the temporary space reaches the limit then use the database function to move the data to MOD. This should force at least two MODS (i.e. force to generate new numbers for MODS). Then it should be tested to load some data back from MOD using different MOD sources This test should be performed on Windows-NT and Windows-2000 since it involves some hardware related functions like media-removal, formatting, etc..

Cause: Wrong Patient selected

Data sets may be "lost" if they are assigned to wrong patients and/or studies.

The MedCom database asks the user before storing volumes or DICOM studies to select a patient. Some files can only be stored if the corresponding parent file already exists in the database.

Test:

1. Test by trying to save volumes without selecting a patient. Ignoring the upcoming window by selecting "next". This should not be possible.
2. Load any volume from the database, import a new volume directly in the main application and save this new one to the database. The database must ask for a new patient name.

Cause: Storing Simulation Plan on wrong patient name

The design of the database doesn't allow storing any simulation plan on a patient that is not the one from that the current 3D data has been loaded from. However, it is possible load 'foreign' simulation plan to the current patient.

A system test should include the try to store (function *Save Simulation Plan*) a simulation on a different patient.

Note: This test is only necessary, if you are using the MedCom database together with *Exomio*.

7.2 Hazard: Error in loading data

Cause: Programming Error

In a system test the usual pipeline should be performed:

1. Loading of data
2. Storing/Importing this data in the database
3. Generating Simulation Plan or Configurations

4. Storing this plan or configurations in the database (Must be stored on the same Patient)
5. Loading this plan or configurations should result in the same settings.

Cause: MO-Handling failed

See above.

7.3 Hazard: Data security

Cause: Database files destroyed:

If the database File *InViVo.mdb* is deleted then all the patient and study data will be lost. Additionally the links between the patient data and studies and the data sets stored in the temp drive and the MO-Discs will be lost. The MedCom database uses a partial backup. After a MO-Disc is full additional to the dataset also the database file *invivo.mdb* is stored on the new formatted MO-Disc.