An Axiometrix Solutions Brand

imc FAMOS Database Kit

Manual

Doc. Rev.: 1.1 - 2022-06-20

imc Test & Measurement GmbH • Voltastr. 5 • 13355 Berlin • Germany

Table of Contents

1 General introduction

1.1 Before you start

Dear user.

1. The software you have obtained, as well as the associated manual are directed toward competent and instructed users. If you notice any discrepancies, we request that you contact our Hotline 4.
2. Updates during software development can cause portions of the manual to become outdated. If you notice any discrepancies, we request that you contact our Hotline.
3. Please contact our Hotline if you find descriptions in the manual which you believe could be misunderstood and thereby lead to personal injury.
4. Read the enclosed license agreement 5. By using the software, you agree to the terms and conditions of the license agreement.

1.2 imc Customer Support / Hotline

If you have problems or questions, please contact our Customer Support/Hotline:

imc Test & Measurement GmbH

Hotline	(Germany):	**+49 30 467090-26**
E-Mail:	hotline@imc-tm.de	
Internet:	www.imc-tm.com	

International partners

For our international partners see www.imc-tm.com/distributors/.

Tip for ensuring quick processing of your questions:

If you contact us **you would help us**, if you know the **serial number of your devices** and the **version info of the software**. This documentation should also be on hand.

- The device's serial number appears on the nameplate.
- The program version designation is available in the About-Dialog.

Product Improvement and change requests

Please help us to improve our documentation and products:

- Have you found any errors in the software, or would you suggest any changes?
- Would any change to the mechanical structure improve the operation of the device?
- Are there any terms or explanations in the manual or the technical data which are confusing?
- What amendments or enhancements would you suggest?

Our Customer Support 4 will be happy to receive your feedback.

1.3 Legal notices

Quality Management

imc Test & Measurement GmbH holds DIN-EN-ISO-9001 certification since May 1995. You can download the CE Certification, current certificates and information about the imc quality system on our website: www.imc-tm.com/quality-assurance/.

imc Warranty

Subject to the general terms and conditions of imc Test & Measurement GmbH.

Liability restrictions

All specifications and notes in this document are subject to applicable standards and regulations, and reflect the state of the art well as accumulated years of knowledge and experience. The contents of this document have been carefully checked for consistency with the hardware and the software systems described. Nevertheless, it is impossible to completely rule out inconsistencies, so that we decline to offer any guarantee of total conformity. We reserve the right to make technical modifications of the systems.

The manufacturer declines any liability for damage arising from:

- failure to comply with the provided documentation,
- inappropriate use of the equipment.

1.4 imc Software License Agreement

imc Test & Measurement GmbH
Voltastr. 5
13355 Berlin
Commercial register: Berlin-Charlottenburg HRB 28778
Managing director: Kai Gilbert, Michael John Flaherty

imc Test & Measurement GmbH
Terms and Conditions
Governing the Use of imc Test & Measurement GmbH Software
As of: June 9, 2022

§ 1 Objects of the Agreement

(1) In addition to the "General Terms and Conditions Governing imc Test & Measurement GmbH Deliveries and Services to Customers", these terms and conditions apply to all contracts concluded with imc Test & Measurement GmbH (hereinafter referred to as "imc") which involve the transfer of rights of use to any software developed by imc (standard software, software created or adjusted specifically for the Customer, which is recorded on the machine-decodable data carriers such as data files, databases and database material, updates, upgrades, releases, etc., including corresponding documentation, information and materials, hereinafter referred to as "Software").

(2) The Software is provided to the Customer as an executable object program on machine-decodable data carriers specified in the "Objects of the Agreement". The Software's product documentation is also supplied to the Customer either in print or on a machine-decodable data carrier. Unless otherwise expressly agreed in writing, the Customer is not issued the source code of the Software.

§ 2 Rights of Use, Scope

With regard to any transfer of rights of use to Software created by imc, the following provisions apply:

(1) Basic provisions

a) The Customer is granted a non-exclusive and – subject to the terms and conditions governing the use of Software by third parties, resale and leasing – non-transferrable right of use to the Software for its own purposes. "Use" signifies running the programs and editing the data records.

b) Until each due fee is paid in full, the Customer is entitled to use the Software solely on a revocable basis. If the Customer is in default with regard to the payment of fees, imc is entitled to revoke the use of the respective services for the duration of the default. The Customer is granted the permanent right to use copyright protected services, in particular the Software, only upon full payment of the agreed fee.

c) The Customer agrees to undertake appropriate precautionary measures to prevent unauthorized access by third parties to the Software. The original data carriers and the data carries used to make copies as per the agreement, as well as the documentation, are to be stored in a secure location. Employees are to be notified that the production of copies beyond the scope of the agreement is not permitted.

d) If the right of use is revoked or expires due to another reason, the Customer is obligated to return to imc the Software, the copies made by the Customer and the documentation. Provided that a physical return of the Software and the copies is not possible due to technical reasons, the Customer is obligated to delete such and confirm deletion to imc in writing.

(2) Reproduction

a) The Customer is entitled to make copies of the Software only if copies are necessary to use the Software in accordance with the contract. The following are considered cases in which reproduction is necessary: installation of the Software from the original data carrier onto the hard disk drive of the hardware used, as well as loading the Software into the computer memory.

b) The Customer is entitled to create a backup copy if such is necessary to safeguard future use. Copies may only be made for other purposes after prior written consent has been issued by imc.

c) The Customer is not allowed to make any reproductions other than those expressly permitted under the provisions of this agreement.

(3) Use of the Software by Third Parties, Resale and Leasing

a) The Software may be used for the purposes stipulated in this contract, in particular for the Customer's business operations. Access to the Software may also be provided to parties which rely on using the Software as instructed by the Customer. In particular, the Customer is entitled to operate the Software or allow the Software to be operated on data processing devices, which are located on the premises of and are directly owned by a third party company (outsourcing). The prohibition against multiple use remains unaffected.

b) The Customer may permanently sell or give the Software to third parties provided that the Customer is granted permanent use of the Software. In the context of its period of use, the Customer may temporarily transfer the Software to third parties for a fee or free of charge. The prohibition against multiple use remains unaffected. The Customer is expressly notified that transfer to third parties is not permitted and use by third parties is technically not possible if an individual license must be acquired or an individual activation is required for third party usage, such as in the case of runtime licenses.

c) With regard to the valid use of Software by a third party, the Customer is obliged to ensure that the third party acknowledges the provisions of this agreement governing the rights of use as binding for such third party. The Customer may not transfer Software and documentation to third parties if there are grounds to suspect that the third party may infringe upon the provisions of this agreement governing the rights of use, in particular with regard to the unauthorized production of copies.

d) Subject to the provisions stipulated in § 4 Paragraphs 1 and 2 or a deviating express agreement in writing, the Customer may not use the Software while the Software is being used by a third party (prohibition against multiple use); in the event that the Software is transferred to the third party, the customer is obliged to surrender to imc all Software copies including, if applicable, all existing backup copies, or to destroy copies not surrendered.

(4) Decompilation

The reverse translation of the provided program code into other code forms (decompilation), disassembling and other forms of reverse engineering of the various production phases of the Software is not permitted. If interface information is required to achieve the interoperability of a separately created computer program, such may be requested from imc, or a third party to be named by imc, for a minor fee. Section 69 e of the German Copyright Act ("UrhG") remains unaffected by this provision.

(5) Changes by imc

If imc conducts adjustments, changes or enhances the Software on behalf and on account of the Customer, the Customer thus acquires the corresponding rights of use to the changes or enhancements of the Software to which he is entitled according to the stipulations of this agreement.

(6) Exceptional Usage Requests by the Customer

If the Customer requests to use the Software according to terms which deviate from the requirements stipulated in Paragraphs 2 through 5, this exceptional use of the Software must be agreed in writing by imc. In such an instance, the Customer agrees to provide imc with information about the desired scope of use, the pertinent field of application, etc. If imc subsequently grants a license covering the Customer's special intended use, the parties agree that a new license fee is owed by the Customer, which is independent of payments made by the Customer for the previously existing license.

§ 3 Copyright, Protection of the Software

(1) The intellectual property, in particular the copyright as well as all industrial property rights and trade secrets, are retained by imc and are not transferred to the Customer. The Customer's ownership of the machine-decodable data carries and data processing units remains unaffected.

(2) Copyright notices, serial numbers as well as designations and reservations of rights which serve as program identification or a protective right may not be removed or changed. The Customer is obliged to transfer the existing protective right notices to all copies. In particular, backup copies of the Software must be expressly designated as such.

§ 4 License Types, Multiple Use

(1) In the case of a Single-User License, the Software may be activated and run on only one data processing unit. "Activation" refers to the process of transferring the license to the data processing unit.

If the technical specifications for the Software permit a second activation, then the Customer may additionally activate the Software on a second data processing unit. However, the Software may only run on one data processing unit at any one time, not on both simultaneously.

(2) With a Network License, the Software may be run on as many data processing units as the amount of licenses obtained. In this case a central data processing unit acts as the license server for which the activation process is performed.

If the technical specifications for the Software permit a second activation, then the Customer may additionally activate and run the Software on as many data processing units as the amount of licenses obtained. However, these additional data processing units must be used by the same users who operate the Software via the license server.

(3) Subject to the provisions in Paragraphs 1 and 2 or a deviating express agreement in writing regarding network use, multiple use of the Software is not permitted.

(4) If the data processing unit is changed, the Customer is obliged to delete the Software from the hard disk drive of the previously used hardware.

§ 5 Software-Subscription

If the Software used is a software-subscription, the following additional restrictions apply:

(1) The right of use is valid for a limited time period. The start and end of the time period are specified. After the end of the time period, the right of use is expired.

(2) If the Customer wishes to continue using the Software after elapse of the specified time period, the subscription must be renewed.

§ 6 Trial Version

If the Software used is a free trial version, then the following additional limitations apply:

(1) The trial version only entitles the user to test the Software. In particular, commercially productive utilization is not permitted.

(2) The rights of use granted expire after the elapse of a period stated in the product description.

§ 7 License Key

(1) Upon delivery of the Software the Customer receives a License Key. Using this License Key, the Customer is able to activate the Software purchased. By means of this License Key the Customer can also view his license status and order updates and upgrades.

(2) The License Key is to be protected against access by third parties in order to prevent misuse. If, however, a third party gains unlawful access to the Key, the Customer is obliged to notify imc immediately via telephone, as well as in writing, so that the previous License Key may be suspended and a new one issued.

§ 8 Conclusion

(1) The law of the Federal Republic of Germany shall apply under exclusion of private international law. The provisions of the UN Convention on Contracts for the International Sale of Goods (CISG) do not apply.

(2) The place of performance for all obligations arising from this agreement is imc's registered seat. Insofar as the Customer is a merchant as defined by the German Commercial Code (HGB), a legal entity under public law, or a special asset under public law, the exclusive place of jurisdiction for all disputes directly or indirectly arising from the contractual relationship is agreed as imc's registered seat. The same applies to persons who have no general place of jurisdiction in Germany, as well as to persons who have moved their place of residence or usual whereabouts abroad since conclusion of the contract, or whose place of residence or usual whereabouts is unknown at the time the action is filed. In addition, imc is entitled to file suit at the statutory venue.

(3) Oral side-agreements are not valid. Deviating or supplementary conditions as well as modifications of this contract, including this written requirement clause, are only valid if agreed in writing and expressly marked as a modification or supplement.

(4) If certain provisions of this contract are inoperative or unfeasible, this does not prejudice other provisions of the contract. The contracting parties agree to contractually substitute an operable provision which approximates the commercial intention of the contract as closely as possible for any inoperable one.

2 imc FAMOS Database Kit

2.1 Overview

This Kit provides functions for accessing databases. The data can be transferred either from the database to FAMOS, or vice-versa from FAMOS to the database. The Kit provides access to:

- Oracle 10g, 11g, 12c
- MS SQL Server 2005, 2008 or higher
- MySQL 5.5, 5.6
- Microsoft SQL Server Compact Edition 4.0
- Database systems about ODBC

The prerequisite is imc FAMOS 7.0 or a higher version.

Multithreading: All functions of the database kit may only be called in the standard-execution thread. A call within a BEGIN_PARALLEL block (i.e. within sequence functions executed in a separate thread) is not allowed.

2.2 Registration in imc FAMOS

To be able to use Database-Kit in FAMOS, it must be registed for use. In a normally proceeding installation, this registration is performed automatically.

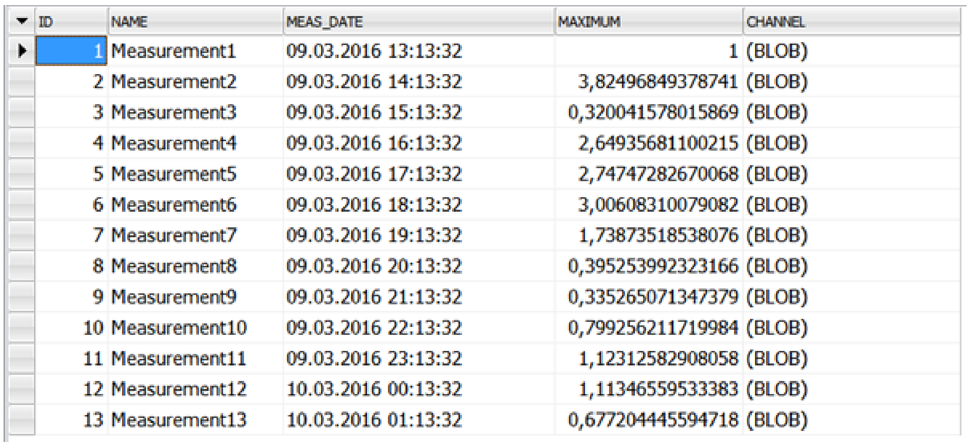
You can register the Kit in FAMOS by means of the menu function "Extra-Options...". In the dialog "Options", select in the tree diagram at left the entry "Register Kits". On the right-side portion, all available Kits are listed. This list should also contain a line for the Database-Kit [imc.Database.Kit.dll]. Put a check-mark in its box.

If this entry is not visible, see the the FAMOS documentation for more information on registering Kits.

2.3 General Remarks

The results of database queries are converted to FAMOS data sets and text arrays. The data sets and text arrays are assigned the names of the columns.

This example is intended to illustrate the conversion. The table Measurement contains the numerical columns ID and MAXIMUM, the text column NAME and the date column MEAS_DATE.

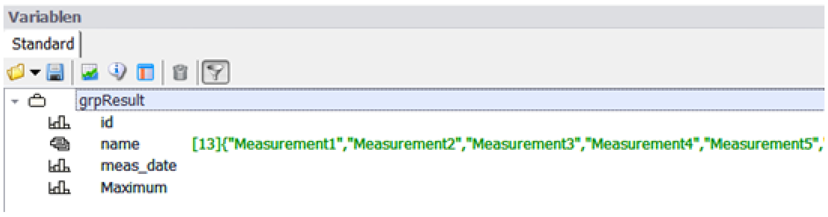


ID	NAME	MEAS_DATE	MAXIMUM	CHANNEL
1	Measurement1	09.03.2016 13:13:32	1	(BLOB)
2	Measurement2	09.03.2016 14:13:32	3,82496849378741	(BLOB)
3	Measurement3	09.03.2016 15:13:32	0,320041578015869	(BLOB)
4	Measurement4	09.03.2016 16:13:32	2,64935681100215	(BLOB)
5	Measurement5	09.03.2016 17:13:32	2,74747282670068	(BLOB)
6	Measurement6	09.03.2016 18:13:32	3,00608310079082	(BLOB)
7	Measurement7	09.03.2016 19:13:32	1,73873518538076	(BLOB)
8	Measurement8	09.03.2016 20:13:32	0,395253992323166	(BLOB)
9	Measurement9	09.03.2016 21:13:32	0,335265071347379	(BLOB)
10	Measurement10	09.03.2016 22:13:32	0,799256211719984	(BLOB)
11	Measurement11	09.03.2016 23:13:32	1,12312582908058	(BLOB)
12	Measurement12	10.03.2016 00:13:32	1,11346559533383	(BLOB)
13	Measurement13	10.03.2016 01:13:32	0,677204445594718	(BLOB)

The query:

```
grpResult =DbSelect(ConnectID,"SELECT Id, Name, Meas_Date, Maximum
FROM Measurement WHERE Id > 4","")
```

generates the group grpResult in FAMOS. The conversion to data sets and text arrays appears as follows in FAMOS' Variables list:


Variablen
Standard

- grpResult
  - id
  - name [13]{"Measurement1","Measurement2","Measurement3","Measurement4","Measurement5",
  - meas_date
  - Maximum

grpResult						
ID		Name		MEAS_DATE		MAXIMUM
5		Measurement5		1140682820.0000		34.8900
6		Measurement6		1140682860.0000		42.7700
7		Measurement7		1140683400.0000		51.0100
8		Measurement8		1140684361.0000		46.8900
9		Measurement9		1140684620.0000		47.0800
10		Measurement10		1140685260.0000		45.6000
11		Measurement11		1140685425.0000		50.0000
12		Measurement12		1140685521.0000		48.3000
13		Measurement13		1140685595.0000		46.9870

Data set Text array Data set Data set

The columns ID and MAXIMUM are each converted to a normal data set. The column MEAS_DATE is converted to a data set in the imc time format. A text array is derived from the column NAME. All group elements are joined together to a group. All group elements have the same size.

In the opposite procedure, meaning when inserting or updating data in the database table, the contents of a FAMOS group's group elements are converted to the data types of a database table's columns and written to the database. The conversion always is oriented to the data type of the database table's column.

2.4 Functions

2.4.1 Database connection

Functions for establishing and ending a database connection

DbInitialize 11	Establishes a defined initial kit condition
DbConnect 11	Sets up a connection to a database server
DbDisconnect 14	Ends a connection

2.4.1.1 DbInitialize

Initialization of the Database-Kit

Declaration:

DbInitialize () -> KitVersion

Parameter:

KitVersion	Kit version number

Description:

This function initializes the Database-Kit. A defined initial state of the kit is established. Any transactions are concluded with a rollback. All connection objects and the internal error memory are cleared. The function returns the kit's version number. This return value is a pure formality.

It is recommended to call this function at the start of every sequence.

Multithreading:

All functions of the database kit may only be called in the standard-execution thread. A call within a BEGIN_PARALLEL block (i.e. within sequence functions executed in a separate thread) is not allowed.

2.4.1.2 DbConnect

Connects with a database server

Declaration:

DbConnect (ServerType,TxServerName,TxDatabaseName,TxUserName,TxPassword, TxExtConnectionString) -> ConnectId

Parameter:

ServerType	Type of database system
	1: Microsoft SQL Server Compact Edition 4.0 by means of ADO.Net Provider
	2: Microsoft SQL Server (2005, 2008) by means of ADO.Net Provider
	3: Oracle (10g, 11g, 12c) by means of Oracle Data Provider for .NET
	4: MySQL Server (5.5, 5.6) by means of MySQL Connector/NET
	5: ODBC
TxServerName	Name of server
TxDatabaseName	Name of database
TxUserName	User name for logging on to server
TxPassword	Password for the specified user name
TxExtConnectionString	Extension of the standard connection string
ConnectId	Connection identifier
	>= 1 : Valid connection identifier
	< 0 : Error code

Description:

A connection to the specified database server is set up. The precondition is that an appropriate ADO.Net provider be installed on the PC.

Access to the database systems requires the following ADO.Net providers:

ServerType	ADO.Net Provider
1 (SQL Server Compact)	Microsoft SQL Server Compact Data Provider 4.0
2 (SQL Server)	.Net Framework Data Provider for SqlServer
3 (Oracle)	Oracle Data Provider for .NET (manufactured by Oracle)
4 (MySQL)	.Net Framework Data Provider for MySQL
5 (ODBC)	A suitable ODBC driver must be installed.

A connection string is constructed from the function's parameters. Then a connection object is created and a connection test is conducted. If the connection test is successful, a connection identifier is returned. This connection identifier represents a connection to a database system. It needs to be specified in all subsequent functions as the first parameter.

If an error occurs, it is possible to retrieve the error text using the function DbGetLastErrorText(). The call DbGetLastErrorText() must be made with the parameter ConnectId = 0.

- TxServerName specifies the name of the database servers.
- TxDatabaseName specifies the name of the database to use.
- The parameter TxUserName is the user name for logging onto the server. If TxUserName and TxPassword are empty, then the logging on is accomplished via the Windows authentication (at this time, only possible with the Microsoft SQL Server).
- The parameter TxPassword contains the password for the specified user name.
- Via TxExtConnectionString, it is possible to set additional elements in the connection string.

For the use of the ODBC interface a system data source must be created and established (choice of the ODBC driver, database server, database and other specific connection parameter). A connecting test should be carried out in the ODBC manager. The name of this system data source is submitted to the function DbConnect() as the parameter TxServerName.

The connection strings are constructed from the function parameters as follows.

ServerType	
1 (SQL Server Compact)	Data Source= **TxDatabaseName**;Password= **TxPassword**;Persist Security Info=False; **TxExtConnectionString**

2 (SQL Server)	With TxUserName and TxPassword
	Server= **TxServerName**; Initial Catalog=**TxDatabaseName** ; Integrated Security= False;User Id= **TxUserName**; Password= **TxPassword**; Persist Security Info=False; **TxExtConnectionString**
	
	Without TxUserName und TxPassword
	Server= **TxServerName**; Initial Catalog=**TxDatabaseName** ; Integrated Security= True; Persist Security Info=False; **TxExtConnectionString**

3 (Oracle)	Data Source= **TxServerName**; User Id= **TxUserName**; Password= **TxPassword**; **TxExtConnectionString**
4 (MySQL)	Server= **TxServerName**; Database= **TxDatabaseName**; Uid= **TxUserName**; Pwd= **TxPassword**; **TxExtConnectionString**
5 (ODBC)	Dsn= **TxServerName**; uid= **TxUserName**;pwd= **TxPassword**; **TxEntConnectionsString;**

Multithreading:

All functions of the database kit may only be called in the standard-execution thread. A call within a BEGIN_PARALLEL block (i.e. within sequence functions executed in a separate thread) is not allowed.

Examples:

A connection is set up to an MS SQLServer Compact Edition 4.0. the database is not password-protected.

```
DbInitialize()
ConnectId =DbConnect(1,"","D:
\DatanbaseStorage\Sample1\sample.sdf","","","")
if ConnectId < 0
      errortext=DbGetLastErrorText(0,1)
      exitsequence 1
end
```

A connection to an MS SQL Server is set up by means of Windows authentication.

```
DbInitialize()
ConnectId=DbConnect(2,"MyPC\SQL2008EXPRESS","SampleDB","","","")
if ConnectId < 0
      errortext=DbGetLastErrorText(0,1)
      exitsequence 1
end
```

A connection to an Oracle database server is set up without using the tnsnames.ora file.

```
DbInitialize()
ConnectId=DbConnect(3,"(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TC
                     P) (HOST=localhost)(PORT=1522)))(CONNECT_DATA=
                     (SERVER=DEDICATED)(SERVICE_NAME=orcl.berlin.imc-
                     berlin.de)))","","MyUsername","MyPassword","")
if ConnectId < 0
      errortext=DbGetLastErrorText(0,1)
      exitsequence 1
end
```

A connection to an Oracle database server is set up by using the tnsnames.ora file.

```
DbInitialize()
ConnectId = DbConnect(3,"ORCL","","MyUsername"," MyPassword","")
if ConnectId < 0
      errortext=DbGetLastErrorText(0,1)
      exitsequence 1
end
```

A connection is set up to a MySql-server, which operates via the TCP-port 3307. The port number is specified as an extension of the connection string.

```
DbInitialize()
ConnectId=DbConnect(4,"localhost","Sample_DB","MyUsername",
"MyPassword ", "port=3307;")
if ConnectId < 0
      errortext=DbGetLastErrorText(0,1)
exitsequence 1
end
```

A connection is set up to a MySQL- server about ODBC. A system data source is established with the name "DSN_MySQL" in the ODBC manager.

```
DbInitialize()
ConnectId = DbConnect(5,"DSN_MySQL","","MyUsername","MyPassword","")
if ConnectId < 0
      errortext=DbGetLastErrorText(0,1)
      EXITSEQUENCE
end
```

References:

DbDisconnect 14, DbGetLastErrorText 28, DbGetLastErrorCode 30

2.4.1.3 DbDisconnect

Disconnects the connection to a database server.

Declaration:

DbDisConnect (ConnectId) -> ErrorCode

Parameter:

ConnectId	Connection identifier
ErrorCode	= 0: no error
	< 0: error code

Description:

The connection to a database server is disconnected. An error can only occur if an invalid connection identifier was transferred.

The functions DbConnect() and DbDisConnect() should always be called in conjunction with each other.

Examples:

A connection is established with an Oracle database server. In that case, the system accesses the database system. In the end, the connection is disconnected.

```
ConnectId = DbConnect(3,"ORCL","","MyUsername"," MyPassword","")
:
:
errorcode = DbDisconnect(ConnectId)
```

References:

DbConnect 11, DbGetLastErrorText 28, DbGetLastErrorCode 30

2.4.2 Data Access

Functions for accessing the database

DbSql 15	Executes a SQL-instruction
DbSelect 16	Executes a SELECT instruction with return of data
DbInsert 18	Executes an INSERT instruction with data linkage for multiple database rows
DbUpdate1 21	Executes an UPDATE instruction with data transfer
DbUpdate 24	Repeated execution of an UPDATE instruction with data transfer

2.4.2.1 DbSql

Executes an SQL-instruction

Declaration:

DbSql (ConnectId, TxSqlStatement) -> Result

Parameters:

ConnectId	Connection identifier
TxSqlStatement	SQL-instruction to be executed This instruction is transferred unchanged to the database system.
Result	>= 0: number of rows affected
	< 0: error code

Description:

With this function, it is possible to execute SQL-instructions which do not return results (UPDATE, INSERT, DELETE) (DML commands). However, it is also possible to execute such DDL commands as "ALTER TABLE Measurement ADD MyCol INTEGER", for example. Once a DML-command has been successfully executed, the return value is the number of rows affected. With a DDL-command, this value is 0 upon successful execution.

The function performs the following operations:

- The connection to the database server is opened.
- Starting of a database transaction
- Execution of the SQL-instruction TxSqlStatement
- If any errors occurred, a rollback is performed.
- If the SQL instruction was performed successfully, it is concluded with Commit.
- The connection to the database server is closed.

Examples:

All values in the column "Status" are set to "Ready".

```
result =DbSql(ConnectId,"Update Measurement set Status ='Ready'")
if result < 0
    errortext=DbGetLastErrorText(ConnectId,1)
end
```

The column "MyCol" with the data type Integer is added to the table "Measurement".

```
result =DbSql(ConnectId,"ALTER TABLE Measurement ADD MyCol Integer")
if result < 0
    errortext=DbGetLastErrorText(ConnectId,1)
end
```

References:

DbGetLastErrorText 28, DbGetLastErrorCode 30, DbInsert 18, DbUpdate1 21, DbUpdate 24

2.4.2.2 DbSelect

Execution of a SELECT instruction, with return of the data

Declaration:

DbSelect (ConnectId, TxSqlStatement, TxTimeColumn) -> GrpResult

Parameters:

ConnectId	Connection identifier
TxSqlStatement	SQL SELECT instruction This instruction is transferred unchanged to the database system.
TxTimeColumn	The content of this column is used to assign a time track to the numerical group elements. This specification is optional.
GrpResult	This group contains the result of the query. The columns selected are converted to data sets and text arrays and returned as a FAMOS group. In case of an error, an empty group is returned.

Description:

With this function, an SQL SELECT-instruction is executed. The function performs the following operations:

- The connection to the database server is opened.
- The SELECT-instruction is executed.
- The results are saved in intermediate storage in a memory-table.
- The connection to the database server is closed.
- The contents of the memory-table are converted to data sets and text arrays. These are joined together to a FAMOS group.

Each column results in a group element of the same name (data set or text array). The number of rows imported determines the size of the group elements. The section "Conversion of the Data 32" contains an overview of how the data types of the columns are converted to FAMOS objects.

The SQL SELECT instruction is transferred without change to the database server.

It is to be noted that column names may need to appear in quotation marks under some circumstances. Reference values consisting of a string or a date must be bracketed in apostrophes. With this function, it is also possible to import contents from multiple tables. The SELECT instruction must be constructed accordingly.

If a query contains one or more Blob columns, the result may only consist of one row. The values of the Blob-column arrays each result in just one data set. If the query returns multiple resulting rows, then the Blob-columns are not converted.

If the query was faulty, an empty group is returned. In this case, it is possible to retrieve the error code using the function DbGetLastErrorCode().

If the parameter TxTimeColumn is set and if it corresponds to a numerical column, then all other numerical columns are converted to XY-data sets, with the time column as the X-component.

The optional parameter TimeStampMap = yes is used to set that all columns of the types Date or Time are converted to a text array. By default, these columns are converted to a normal data set in the imc time format.

With the optional parameter NumericAsDouble = yes, all numerical columns are converted to data sets in 8-Byte Real (double) format. In the default case, the conversion is documented as presented in the section "Converting the data 32".

Example:

The query

```
      sqlStatement="SELECT Id,~034A B~034 ,Name FROM Measurement"
      grpResult = DbSelect(ConnectID, sqlStatement,"")
```

returns a group.

The group contains the normal data sets ID and A B as well as the text array Name.

One special feature in this query is the column A B. Since the name contains a space, this column must appear in quotation marks. Since the query was sent to an Oracle database server, the quotations marks must be used here. In the FAMOS sequence, the substitute character ~034 must be used.

With this query, all columns of the table Measurement are supposed to be imported.

```
grpResult = DbSelect(ConnectID,"Select * from Measurement WHERE Id >
0","")
errorcode=DbGetLastErrorCode(ConnectID,0)
if errorcode < 0
      ; posts a warning, since the column CHANNEL is a Blob-column
      ; and the result contains multiple rows
      errortext=DbGetLastErrorText(ConnectID,1)
      BoxMessage("Error",errortext,"S1")
end
```

Since the column CHANNEL is a Blob-column and there are multiple rows in the result, the function returns an error. The contents of the Blob-column can not be converted. For this reason, the results group contains only the group elements ID, NAME, MEAS_DATE and MAXIMUM; the group element CHANNEL is missing. If the contents of the Blob-column are also to be read and converted, the query would have to be changed so that only one result row per query would be generated.

```
grpResult = DbSelect(ConnectID,"Select * from Measurement WHERE Id =
1","")
```

References:

DbGetLastErrorText 28, DbGetLastErrorCode 30, DbOption 31

2.4.2.3 DbInsert

Execution of INSERT instructions with data linkage for multiple database rows.

Declaration:

DbInsert (ConnectId, TxTableName, GrpInsertValues) -> Result

Parameters:

ConnectId	Connection identifier
TxTableName	Name of the database table in which values are to be inserted
GrpInsertValues	The FAMOS group contains data used for transfer to the database. Each sample from the group elements initiates INSERT of a new row into the table
Result	>= 0: Number of rows inserted
	< 0: Error code

Description:

With this function, data from FAMOS-objects can be inserted into a database table. The names of the data sets and text arrays in the group must be present in the database table as columns. All element contained in the group are inserted into the table. The data sets and text arrays of the group must have the same size. The size determines the number of table rows inserted.

If any Blob-column is included, then the size of the group elements may only be 1. The name of the data set corresponding to the Blob-column is inserted into the table's one array. If the size of the group elements is > 1, then only the data with the first index are inserted.

The function performs the following operations:

- An empty memory table of the specified table is created.
- The data sets and text arrays of the FAMOS group are checked against the memory table.
  Does a column having the same name exist for each group element?
  Do all group elements have the same number of values?
  Are all data sets belonging to the FAMOS group of the data type "Normal" ?
- The data from the FAMOS group's elements are converted and saved in the memory table for intermediate storage. The conversion is always oriented to the data type of the database table's column.
- The connection to the database server is opened.
- A database transaction is started.

- An INSERT instruction is constructed from the table name and the name of the group elements. All column names appear in quotation marks.
- For each row of the memory-table, an INSERT command with the associated values is sent to the database server.
- If errors occurred in the course of executing INSERT, a rollback is performed.
- If all INSERT commands were executed successfully, a commit is performed.
- The connection to the database server is closed.

Examples:

A new row with a Blob-column is generated.

```
; --- Inserting a channel as a Blob ----------------------------
; --- Important! Only one row can be loaded ------------
grpResult = DbSelect(ConnectID,"Select * from Measurement  WHERE Id =
1","")
errorcode = DbGetLastErrorCode(ConnectID,0)
if errorcode < 0
     errortext = DbGetLastErrorText(ConnectID,1)
     BoxMessage("Error",errortext,"S1")
end
; --- Determining a new ID value for the insertion ------------------
grpMax = DbSelect(ConnectID,"Select Max(Id) AS MaxID from Measurement
","")
maxid = grpMax:MaxID[1]
grpResult:ID[1]=maxid+1
; --- Loading the channel Sintest1 ----------------------------------
FileLoad("Sintest1.dat","",0)
grpResult:CHANNEL=sintest1
; --- Saving the triggering time as a date --------------------------
grpResult:Meas_Date[1]=Time?(grpResult:CHANNEL)
grpResult:Name[1] ="Sintest1"
grpResult:Maximum[1]=Max(sintest1)
; --- The channel is inserted into the table as a Blob --------------
; --- Important! The group elements may only have the size 1 -----
Result = DbInsert(ConnectID,"Measurement",grpResult)
if result < 0
     errortext = DbGetLastErrorText(ConnectID,1)
     BoxMessage("Error",errortext,"S1")
end
```

Multiple rows are imported from the table, edited and inserted as new rows.

```
grpResult =DbSelect(ConnectID,"Select Id, Name, Meas_Date, Maximum from
                  Measurement WHERE Id > 5 And Name like 'Mess%' ","")
; --- Determining new ID value for the insertion ------------------
grpMax = DbSelect(ConnectID,"Select Max(Id) AS MaxID from Measurement
","")
; --- Setting new ID values for the insertion ---------------------
maxid = grpMax:MaxID[1]+1
FOREACH SAMPLE s in grpResult:ID
     s=s+maxid
END
; --- Setting a new name for the insertion ------------------------
FOREACH ELEMENT s in grpResult:NAME
     s=s+"_New"
END
; --- Setting new date for the insertion -----------------------
FOREACH SAMPLE s in grpResult:MEAS_DATE
     s=s+ 86400
END
; --- Setting new maximum for the insertion ---------------------
FOREACH SAMPLE s in grpResult:Maximum
     s=0
END
; --- Beginning transaction (not actually necessary, ----------
; --- serves to test the functions DbBeginTransaction() and  ---
; --- DbEndTransaction() ----------------------------------------
DbBeginTransaction(ConnectID)
; --- Inserting group in the table Measurement --------------------
result=DbInsert(ConnectID,"Measurement",grpResult)
if result < 0
     errortext=DbGetLastErrorText(ConnectID,1)
     BoxMessage("Error",errortext,"S1")
     ; --- Concluding transaction with a Rollback ----------
     DbEndTransaction(ConnectId,0)
Else
     ; --- Concluding transaction with a Commit ------------
     DbEndTransaction(ConnectId,1)
End
```

References:

DbGetLastErrorText 28, DbGetLastErrorCode 30, DbBeginTransaction 26, DbEndTransaction 28

2.4.2.4 DbUpdate1

Executes an SQL-update instruction

Declaration:

DbUpdate1 (ConnectId, TxSqlStatement, GrpUpdateValues, SampleIndex) -> Result

Parameters:

ConnectId	Connection identifier
TxSqlStatement	SQL UPDATE instruction The values to be set are not specified in the instruction, but only placeholders of the type "?Name of the group element".
GrpUpdateValues	Contains updating data and data for the WHERE-condition The FAMOS group contains data used for updating and for the WHERE-condition. Only such data belonging to the group elements are used which are addressed by the SampleIndex.
SampleIndex	Updating of the database values (1...) is only performed on data from the group elements which are addressed by the SampleIndex.
Result	>= 0: Number of updated rows
	< 0: Error code

Description:

With this function, an UPDATE-instruction is executed. The UPDATE-instruction must be specified completely as a TxSqlStatement. The placeholder to use for the group element values to be replaced is the question mark **?+Name of the group element**. The placeholders in the SET-portion and in the WHERE-condition are replaced with data from the group GrpUpdateValues.

For the Update-instruction, only such data belonging to the group elements are used which are addressed by the SampleIndex. This group may contain more group elements than required for the Update-instruction. Only such group elements are used which are specified in the Update-instruction.

The function performs the following operations:

- The SQL UPDATE-instruction is tested for the presence of the keywords UPDATE and SET.
- The table name is determined from the UPDATE-instruction. An empty memory table of the specified table is created.
- In the SET-portion, the columns to be updated are determined. The placeholder "**?Name of group element**" is replaced with a named parameter of the database system. For the column, a group element having the same name must be present in the group GrpUpdateValues.
- In the WHERE-condition, the "?Name of group element"-placeholders are replaced with named parameters of the database system. For the column, a group element having the same name must be present in the group GrpUpdateValues.
- The data addressed via the SampleIndex are converted to the corresponding data types of the memory-table.
- The connection to the database server is opened.
- A database transaction is started.
- The UPDATE-instruction is executed.
- If errors occurred, a rollback is performed.
- If all UPDATE commands were executed successfully, a commit is performed.
- The connection to the database server is closed.

Examples:

The MAXIMUM of the row with the ID-value 11 is to be set to 0.

```
grpResult =DbSelect(ConnectID,"Select Id,Name,Meas_Date,Maximum from
                  Measurement”, "")
grpResult:Maximum[11]=0;
updatestatement="update Measurement set Maximum=?Maximum where Id =?Id
                "
result=DbUpdate1(ConnectId,updatestatement,grpResult,11)
if result < 0
     errortext=DbGetLastErrorText(ConnectID,1)
     BoxMessage("Error",errortext,"S1")
End
```

To begin, the values of the columns ID, NAME, MEAS_DATE and MAXIMUM are imported.

The following illustration shows the components of the FAMOS group. The 11th value of the data set MAXIMUM is set to 0.

The UPDATE-instruction “Update Measurement Set Maximum=?Maximum Where Id =?ID“ contains 2 placeholders. The first placeholder is in the SET-portion and the second in the WHERE-condition.

The placeholders are replaced with the concrete values imported from the SampleIndex = 11. The columns NAME and Meas_Date are also in this group, but they are ignored since only the columns MAXIMUM and ID appear with a placeholder in the UPDATE-instruction. In this case, the UPDATE-command

“Update Measurement Set Maximum=0 Where Id =11“

is sent to the database server. The function's result is 1, since the WHERE-condition only permits one row in the database table.

		grpResult			
	Sample Index	ID	Name	MEAS_DATE	MAXIMUM
	1	1	Measurement1	1140681480.0000	45.3000
	2	2	Measurement2	1140681600.0000	39.8000
	3	3	Measurement3	1140681800.0000	46.0000
	4	4	Measurement4	1140682470.0000	43.9990
	5	5	Measurement5	1140682820.0000	34.8900
	6	6	Measurement6	1140682860.0000	42.7700
	7	7	Measurement7	1140683400.0000	51.0100
	8	8	Measurement8	1140684361.0000	46.8900
	9	9	Measurement9	1140684620.0000	47.0800
	10	10	Measurement10	1140685260.0000	45.6000
->	11	11	Measurement11	1140685425.0000	50.0000 -> 0
	12	12	Measurement12	1140685521.0000	48.3000
	13	13	Measurement13	1140685595.0000	46.9870

↑ ↑

This group element is used in the WHERE-condition.		This group element is used in the UPDATE-clause

In the next example, the values of MAXIMUM are to be set to 1000 wherever the name ends with 1x.

```
grpResult =DbSelect(ConnectID,"Select Id,Name,Meas_Date,Maximum from
                  Measurement”, "")
grpResult:Maximum[1]=1000;
updatestatement="update Measurement set Maximum=?Maximum where Name
                  LIKE '%1_' "
result=DbUpdate1(ConnectId,updatestatement,grpResult, 1)
if result < 0
     errortext=DbGetLastErrorText(ConnectID,1)
     BoxMessage("Error",errortext,"S1")
end
```

Once the columns have been imported, the value of Maximum having the SampleIndex = 1 is set to 1000.

In the UPDATE-instruction, there is only a placeholder in the SET-portion. The WHERE-condition "Name LIKE ‘%1_‘ " finds 4 rows to be updated. The function's result is 4.

The following sequence steps update a channel in a Blob-column.

```
grpResult =DbSelect(ConnectID,"Select Id,
Name,Meas_Date,Maximum,Channel from Measurement where ID=10 ","")
; --- Loading channel Sintest1
-------------------------------------------
FileLoad("Sintest1.dat","",0)
; --- The data set sintest1 is assigned to the group element Channel -
grpResult:CHANNEL=sintest1
updatestatement="update Measurement set channel=?channel where Id =?Id"
result=DbUpdate1(ConnectId,updatestatement,grpResult, 1)
if result < 0
      errortext=DbGetLastErrorText(ConnectID,1)
      BoxNachricht("Fehler",errortext,"S1")
end
```

The row with the ID = 10 is imported from the database table. The channel sintest1 is imported and assigned to the group element CHANNEL. By running DbUpdate1(), the entire channel is assigned to the Blob-array CHANNEL in the table row having the ID=10. The result of the function is 1.

References

DbSelect 16, DbUpdate 24, DbGetLastErrorText 28, DbGetLastErrorCode 30, DbBeginTransaction 26, DbEndTransaction 28

2.4.2.5 DbUpdate

Repeated execution of an SQL-Update instruction

Declaration:

DbUpdate (ConnectId, TxSqlStatement, GrpUpdateValues) -> Result

Parameters:

ConnectId	Connection identifier
TxSqlStatement	SQL UPDATE instructions The values to be set are not specified in the instruction, but only placeholders ("?Name of group element").
GrpUpdateValues	Contains data to be updated and data for the WHERE-condition The FAMOS group contains data used for updating and for the WHERE-condition.
Result	>= 0: Number of updated rows
	< 0: Error code

Description:

With this function, one UPDATE-instruction is carried out for each index of the data sets passed to it. The UPDATE-instruction must be specified completely as a TxSqlStatement. The placeholder to use for the group element values to be replaced is the question mark "**?+Name of group element**".

If the UPDATE-instruction contains a Blob-column, then only the 1st row in the database table is updated.

The function performs the following operations:

- The SQL UPDATE-instruction is tested for the presence of the keywords UPDATE and SET.
- The table name is determined from the UPDATE-instruction. An empty memory table of the specified table is created.
- In the SET-portion, the columns to be updated are determined. The placeholder "**?Name of group element**" is replaced with a named parameter of the database system. For the column, a group element having the same name must be present in the group GrpUpdateValues.
- In the WHERE-condition, the "**?Name of group element**"-placeholders are replaced with named parameters of the database system. For the column, a group element having the same name must be present in the group GrpUpdateValues.
- The data belonging to the group GrpUpdateValues are converted to the corresponding data types of the memory-table. All group elements must have the same number of values.
- The connection to the database server is opened.
- A database transaction is started.
- The UPDATE-instruction is executed once for each row in the memory-table.
- If errors occurred, a rollback is performed.
- If all UPDATE commands were executed successfully, a Commit is performed.
- The connection to the database server is closed.

Examples:

The columns ID, NAME and MEAS_DATE are imported. The group elements have the size 13. Next, the individual names are changed. The changed group subsequently updates the database table.

```
grpResult =DbSelect(ConnectID,"Select Id, Name,Meas_Date from
Measurement","")
; --- Forming new name for the update ---------------
for i= 1 to leng?(grpResult:ID)  Step 1
    measurement=grpResult:Name[i] +"_"+ TForm(grpResult:ID[i],"")
    grpResult:Name[i]=measurement
end

updatestatement="update Measurement set Name=?Name where Id =?Id "
result=DbUpdate(ConnectId,updatestatement,grpResult)
if result < 0
    errortext=DbGetLastErrorText(ConnectID,1)
    BoxMessage("Error",errortext,"S1")
end
```

The UPDATE-instruction is carried out 13 times. Upon each UPDATE, the placeholders are replaced with the corresponding data from the group elements. The function's result is 13. The figure below shows the table contents both before and after the update. Although the column Meas_Date is imported, since it doesn't appear in the UPDATE-instruction, it is ignored.

Table: before

ID	Name
1	Measurement1
2	Measurement2
3	Measurement3
4	Measurement4
5	Measurement5
6	Measurement6
7	Measurement7
8	Measurement8
9	Measurement9
10	Measurement10
11	Measurement11
12	Measurement12
13	Measurement13

Table: after

ID	Name
1	Measurement1_1
2	Measurement2_2
3	Measurement3_3
4	Measurement4_4
5	Measurement5_5
6	Measurement6_6
7	Measurement7_7
8	Measurement8_8
9	Measurement9_9
10	Measurement10_10
11	Measurement11_11
12	Measurement12_12
13	Measurement13_13

References

DbSelect 16, DbUpdate1 21, DbGetLastErrorText 28, DbGetLastErrorCode 30, DbBeginTransaction 26, DbEndTransaction 28

2.4.3 Transactions

Functions for controlling transactions

`DbBeginTransaction` 26	Begin transaction
`DbEndTransaction` 28	End transaction

2.4.3.1 DbBeginTransaction

Starts a database transaction.

Declaration:

DbBeginTransaction (ConnectId) -> ErrorCode

Parameters:

ConnectId	Connection identifier
ErrorCode	= 0: no error
	< 0: error code

Description:

This function starts a transaction. The function can be used when multiple tasks in a sequence are to be joined together in order that they can be executed as a single procedural unit.

For instance, suppose data are to be inserted into Table A and simultaneously, associated data are to be inserted into Table B. If an error occurs in the insertion into Table B, then insertion into Table A is to be canceled and rolled back. This can be achieved with the two functions DbBeginTransaction() and DbEndTransaction() working in conjunction as a pair.

These functions do not need to be used where there is only a single call of DbSql(), DbInsert(), DbUpdate() or DbUpdate1(). These functions have an internal transaction control system. For instance, if the elements of a group are to be inserted using DbInsert(), then either all data belonging to the elements are inserted (and a Commit is performed), or if an error occurred, all changes are discarded (an internal rollback is performed).

When the function DbBeginTransaction() is called, it overrides the internal transaction control. It only becomes active again once DbEndTransaction() is called.

Nesting of calls to DbBeginTransaction() is not supported.

The function performs the following operations:

- The connection to the database server is opened.
- A database transaction is started.
- The connection to the database server remains open.

Always use DbBeginTransaction() and DbEndTransaction() in pairs.

If the function DbDisconnect() encounters an open transaction, it is concluded with a rollback.

Examples:

The FAMOS-group grpInsert1 is to be inserted into the table “Users” and the FAMOS-group grpInsert2 is to be inserted into the table “Group“. For this purpose, a transaction is started. if both functions were concluded without errors, the transaction is concluded with a Commit. In case of error, a rollback is performed.

```
errorcode=DbBeginTransaction(ConnectID)
if errorcode < 0
      errortext=DbGetLastErrorText(ConnectID,1)
end
result1=DbInsert(ConnectID,"Users",grpInsert1)
result2=DbInsert(ConnectID,"Group",grpInsert2)

commit=1
if result1 < 0 or result2 < 0
      errortext=DbGetLastErrorText(ConnectID,1)
      commit=0;
end
errorcode = DbEndtransaction(ConnectID,commit)
if errorcode < 0
      errortext=DbGetLastErrorText(ConnectID,1)
end
```

References:

DbEndTransaction 28, DbGetLastErrorText 28, DbGetLastErrorCode 30

2.4.3.2 DbEndTransaction

Save or rollback the changes since the last DbBeginTransaction.

Declaration:

DbEndTransaction (ConnectId, Commit) -> ErrorCode

Parameters:

ConnectId	Connection identifier
Commit	= 0: The transaction is reset. All changes are discarded (rollback)
	<> 0: The transaction is saved. Any changes are saved permanently. (Commit)
ErrorCode	= 0: no error
	< 0: error code

Description:

With this function, a transaction is concluded. Use of the function is only permitted in conjunction with the function DbBeginTransaction().

Use DbBeginTransaction() and DbEndTransaction() always jointly.

The function performs the following operations:

- The transaction is saved/reset according to the parameter “Commit“.
- The connection to the database server is closed.

References:

DbBeginTransaction 26, DbGetLastErrorText 28, DbGetLastErrorCode 30

2.4.4 Miscellaneous

Functions for error analysis and for options

DbGetLastErrorText 28	Finds the last error text
DbGetLastErrorCode 30	Finds the last error code
DbOption 31	Sets optional parameters

2.4.4.1 DbGetLastErrorText

Finds the last error text

Declaration:

DbGetLastErrorText (ConnectId, ClearError) -> TxError

Parameters:

ConnectId	Connection identifier
ClearError	0: Error remains in the error memory
	1: subsequently clear the error memory
TxError	Error text

Description:

This function enables the text associated with an error to be read. If a 1 is specified for the parameter "ClearError", then the error memory is subsequently cleared. Each connection has its own error memory. For this reason, the parameter "ConnectId" needs to be specified.

If the function DbConnect() fails, then when calling DbGetLastError() it is necessary to specify the parameter ConnectId = 0 (DbGetLastError(0,1)).

The errors are described in the section Error Codes 35.

Reference:

DbGetLastErrorCode 30

2.4.4.2 DbGetLastErrorCode

Determines the last error number.

Declaration:

DbGetLastErrorCode (ConnectId, ClearError) -> ErrorCode

Parameters:

ConnectId	Connection identifier
ClearError	0: Error remains int he error memory
	1: Clear error memory subsequently
ErrorCode	Error number

Description:

This function enables retrieval of the number of the last error to occur. If a 1 is provided as the value of the parameter "ClearError", then the error memory is subsequently cleared. Each connection has its own error memory. For this reason, the parameter "ConnectId"must be specified.

In case of error, most Kit-functions return the error number. Error numbers are always negative numbers.

The errors are described in the section Error Codes 35.

Reference:

DbGetLastErrorText 28

2.4.4.3 DbOption

Sets optional parameters

Declaration:

DbOption (ConnectId, TxParametername, TxParameterValue) -> ErrorCode

Parameters:

ConnectId	Connection identifier
TxParametername	Name of the optional parameter
TxParameterValue	Values of the optional parameter
ErrorCode	=0: No error
	<0: Error code

The following parameters are defined:

Parameter name	Value	Remarks
“TimeStampMap”	“yes” or “no”	Affects the function DbSelect() If the parameter is set to “yes“, the contents of the DateTime columns are converted to a text array. For a “no“, these columns are converted to a normal data set. Default value: “no“
“NumericAsDouble”	“yes” or “no”	Affects the function DbSelect() If the parameter is set to “yes“, the contents of numerical columns are always converted to a normal data set with the data format 8-Byte Real. If the value is “no“, the conversion is done as explained in the section "Converting the data 32". Default value: “no“
“CommandTimeOut“	“xx” in seconds	Affects the functions DbSql(), DbSelect(), DbInsert(), DbUpdate1() and DbUpdate() by means of these parameters, the time is specified in seconds to wait until an attempt to execute a command is abandoned and en error message is posted. With Microsoft SQL Server Compact Edition 4.0, this parameter has no relevance. The default value is 30 seconds.

Description:

With this function, it is possible to set optional parameter for each connection separately.

The name of the optional parameter must match a stipulated name.

References:

DbSelect 16, DbSql 15, DbInsert 18, DbUpdate1 21, DbUpdate 24, DbGetLastErrorText 28, DbGetLastErrorCode 30

2.5 Converting the data

When data are imported from a table, the values are converted to either data sets or text arrays according to the column's data type.

Data type of Memory-Table	Value range	Option	FAMOS-object	FAMOS-data format
Boolean	true, false	1	Normal data set	1 Byte Integer
Byte	0...255	1	Normal data set	1 Byte Unsigned Integer
Char			Text array	
DateTime		2	Normal data set	8 Byte Real
			Text array	
DateTime Offset		2	Normal data set	8 Byte Real
			Text array	
Decimal	-79228162514264337593543950335m 79228162514264337593543950335m		Normal data set	8 Byte Real
Double	-1,79769e+308 ... 1,79769e+308		Normal data set	8 Byte Real
Guid			Text array	
Int16	-32768 ... 32767	1	Normal data set	2 Byte Integer
Int32	-2147483648 ... 2147483647	1	Normal data set	4 Byte Integer
Int64	-9223372036854775808 ... 9223372036854775807	1	Normal data set	8 Byte Integer
SByte	-128 ... 127	1	Normal data set	1 Byte Integer
Single	-3,40282e+038f ... 3,40282e+038f	1		4 Byte Real
String			Text array	
TimeSpan		2	Normal data set	8 Byte Real
			Text array	
UInt16	0 ... 65535	1	Normal data set	2 Byte Unsigned Integer
UInt32	0 ... 4294967295	1	Normal data set	4 Byte Unsigned Integer
UInt64	0... 18446744073709551615	1		8 Byte Unsigned Integer
Byte[]			Normal data set	1 Byte Unsigned Integer

If a value of any of the data types Decimal, Double or Single lies outside of the limits of the FAMOS value range, then this value will be reset to the value limit (-10^{35} ... 10^{35}) .

If a Date/Time column is converted to a data set in the imc time format, and the specified time is prior to 1980, then it is reset to FAMOS' value limit of 1.1.1980.

Option 1

By default, the column's data type is converted to the specified FAMOS data format. If the optional parameter "NumericAsDouble" is set to "yes", then these columns are converted to the FAMOS format 8-Byte Real.

Option 2

By default, the column is converted to a normal data set. If the optional parameter "TimeStampMap" is set to "yes", these columns are converted to a text array.

Note

The following Microsoft SQL Server data types are not supported:

- SqlGeography
- SqlGeometry
- SqlHierarchyId
- SQLVariant

2.6 Treatment of null values

Null values from the database tables can not be saved in the FAMOS-objects (data sets and text arrays).

When null values are converted to text arrays, the value is always an empty string.

When converting null values of numerical columns to FAMOS data sets, the following replacement values are applied:

Data type of Memory-Table	Value range	FAMOS data format in the data set	Replacement value for null value
Boolean	True, false	Signed1	-128
Byte	0…255	Unsigned1	0
DateTime		Double	0.0
Decimal	-79228162514264337593543950335m 79228162514264337593543950335m	Double	1e35
Double	-1,79769e+308 ... 1,79769e+308	Double	1e35
Int16	-32768 ... 32767	Signed2	-32768
Int32	-2147483648 ... 2147483647	Signed4	-2147483648
Int64	-9223372036854775808 ... 9223372036854775807	Signed8	-9223372036854775808
SByte	-128 ... 127	Signed1	-128
Single	-3,40282e+038f ... 3,40282e+038f	Float	1e35
TimeSpan		Double	1e35
UInt16	0 ... 65535	Unsigned2	0
UInt32	0 ... 4294967295	Unsigned4	0
UInt64	0... 18446744073709551615	Unsigned8	0

2.7 Error Codes

The error numbers are always negative numbers.

Error number	Description
-1	Failure to specify a value for a parameter. E.g. an empty SQL instruction or a missing table/server name
-2	The function DbConnect() returns a connection identifier. All other functions expect a valid connection identifier and check it to find whether a connection exists. If not, this error number appears.
-3	Provider error: Error text This error can appear when the connection to the database server is established. The provider raises an exception. In case of error, the cause of this exception is shown.
-4	DBMS error: Error text If an SQL instruction (SELECT, UPDATE ...) is defective, then the database server reports an error. This error is returned as an error text.
	DbConnect()
-10	Invalid value for the server type Parameter The function DbConnect() was called with a server type not previously stipulated. Valid values range from 1 to 4.
-11	No provider installed for the selected database system. Required: No connection can be established to the specified server type, since the required ADO.NET provider is not installed on the PC. The required provider is stated in the error message.
-12	Unable to establish a connection to the DBMS. Reason: Unable to generate a connection object to the database system. The cause may be a defective connection string. The cause of failure is stated in the error text.
-13	Unable to establish a connection to the DBMS. Reason: The Microsoft SQL Server Compact Edition database is write-protected. This error message only appears for the Microsoft SQL Server Compact edition. It can be resolved by removing the database file's write-protection.
-14	Connection to the server version is not supported. Access to this server version, e.g. Oracle 8 or Microsoft SQL Server 2000, is not supported.
	DbSelect()
-20	For the following columns, no conversion rule is implemented ... The columns whose data type can not be converted to FAMOS-objects are enumerated. These data types include: SqlGeography, SqlGeometry, SqlHierarchyId, SQL variant of Microsoft SQL Server. For all other columns, data sets/text arrays were generated.

Error number	Description
-21	Since the query returned more than one result row, the Blob-columns "column name" could not be converted. When a query containing Blob-columns is made, and it returns multiple result rows, the Blob-columns can not be converted to FAMOS-objects. All other columns belonging to the query are contained in the resulting group as FAMOS-objects. In order to read a Blob-column, the query must be expressed in a way that only returns one results row.
-22	The time column xy does not exist in the query results. Conversion to XY-data sets is not possible. If the function DbSelect() is called with the parameter TxTimeColumn and this column is not present in the query results, then there is no conversion to XY-data sets. All numerical columns are returned as normal data sets.
-23	The time column xy is not a normal data set. The specified time column must be a numerical column which can be converted to a normal data set. If that is not the case, no conversion to XY-data sets will occur. All numerical columns are returned as normal data sets.
-24	There are no more data sets which can be converted to an XY-data sets with the time column as the X-component. Besides the time column there are no more numerical columns. for this reason, conversion to XY-data sets can not be performed.
	DbInsert(), DbUpdate1(), DbUpdate()
-30	The following columns do not exist in the table (table): (column) In the function DbInsert(), a group with group elements (data sets or text arrays) is passed. The names of the group elements are checked against the database table's column names. Any group elements names which do not match any column name are listed in the error message. Insertion is not performed.
-31	The group contains no elements. An emptu FAMOS group has been passed to one of the functions DbInsert(), DbUpdate1() or DbUpdate().
-32	The data type of the data set xyz is not supported. Only a "normal" data set is allowed. The FAMOS group passed contains a channel which is not of the data type "Normal data set". Only channels of the data type "Normal data set" are supported.
-33	Conversion error in the column (column name): (error text) Before data are inserted into/updated in the database, the contents of the group elements are converted to the data type of the database table column. If an error occurs in the conversion, this error message appears, in which the column and the cause are stated.
-34	The group elements are of different lengths. When data are inserted into or updated in the database, all elements of the FAMOS group must have the same size.

Error number	Description
-35	The data type (data type) of the column (column name) is not supported. Values were to be inserted into or updated in a a column of the database whose data type is not supported. Data types to which this applies include SqlGeography, SqlGeometry, SqlHierarchyId, SQLVariant by Microsoft SQL Server. Insertion/updating will not be performed.
-36	Error in the update-instruction; unable to find the keyword (keyword). In the SQL-instruction for either of the functions DbUpdate1() or DbUpdate(), the keyword UPDATE, or SET, was not found.
-37	Error in the update-instruction; no table name found Unable to find any table name for the SQL-instruction of either the function DbUpdate1() or DbUpdate().
-38	Error in the update-instruction; SET-portion is empty In the SQL-instruction for either of the functions DbUpdate1() or DbUpdate(), the SET-portion is empty.
-39	Error in the update-instruction; column (column name) does not exist. In the SQL-instruction for either of the functions DbUpdate1() or DbUpdate(), a column is specified in the SET-portion which does not exist.
-40	Error in the update-instruction in the region of... In the SQL-instruction for either of the functions DbUpdate1() or DbUpdate(), an error occurred in the WHERE-condition. The defective part of the WHERE-condition is specified in the error message.
-41	For the column (column name), no group element exists. In the SQL-instruction for either of the functions DbUpdate1() or DbUpdate(), a column has been specified for which there is no group element of that name.
-42	The Sample-Index is 0 or greater than the group element length. The parameter Sample Index in the function DbUpdate1() lies outside of the range (1... maximum number of group element values)
-43	The group element … contained in the update instruction does not exist of the group.
	DbOption()
-50	The optional parameter (parameter name) has not been defined. In the function DbOption(), a parameter name not previously defined has been specified.
-51	The timeout value is not valid. Specification: 1... in seconds The optional parameter "CommandTimeOut" has been set to an invalid value. The value specified must be an integer and stated in seconds.
	Internal error
-300	Internal error: DBConnection-object not present

Error number	Description
	When a connection was opened or database command generated, no connection object was present.
-301	Internal error: No data source present No data source present when executing an SQL-instruction
	Exception error in the Kit functions
-400	Exception in DbConnect: (reason) An unexpected exception occurred in the function.
-401	Exception in DbDisConnect: (reason) An unexpected exception occurred in the function.
-402	Exception in DbInitialize: (reason) An unexpected exception occurred in the function.
-500	Exception in DbSql: (reason) An unexpected exception occurred in the function.
-501	Exception in DbSelect: (reason) An unexpected exception occurred in the function.
-502	Exception in DbInsert: (reason) An unexpected exception occurred in the function.
-503	Exception in DbUpdate1: (reason) An unexpected exception occurred in the function.
-504	Exception in DbUpdate: (reason) An unexpected exception occurred in the function.
-600	Exception in DbBeginTransaction: (reason) An unexpected exception occurred in the function.
-601	Exception in DbEndTransaction: (reason) An unexpected exception occurred in the function.
-700	Exception in DbGetLastErrorText: (reason) An unexpected exception occurred in the function.
-701	Exception in DbGetLastErrorCode: (reason) An unexpected exception occurred in the function.
-702	Exception in DbOption: (reason) An unexpected exception occurred in the function.

Index