BS ISO/IEC 9075-3:2023

BSI Standards Publication

Information technology — Database languages SQL

Part 3: Call-Level Interface (SQL/CLI)

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-3:2023

National foreword

This British Standard is the UK implementation of ISO/IEC 9075-3:2023.

The UK participation in its preparation was entrusted to Technical Committee IST/40, Data management and interchange.

A list of organizations represented on this committee can be obtained on request to its committee manager.

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

© The British Standards Institution 2023
Published by BSI Standards Limited 2023

ISBN 978 0 539 21709 4

ICS 35.060

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 June 2023.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-3:2023

INTERNATIONAL STANDARD

ISO/IEC 9075-3

Sixth edition
2023-06

Information technology — Database languages SQL —

Part 3: Call-Level Interface (SQL/CLI)

Reference number
ISO/IEC 9075-3:2023(E)

Documents
Document Sharing Hub

© ISO/IEC 2023

Full document details: BS ISO/IEC 9075-3:2023

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2023

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-3:2023

Contents

Page

Foreword . . . ix
Introduction . . . xi
1 Scope . . . 1
2 Normative references . . . 2
3 Terms and definitions . . . 3
4 Concepts . . . 4
4.1 Notations and conventions . . . 4
4.1.1 Notations . . . 4
4.1.2 Specification of routine definitions . . . 4
4.2 Introduction to SQL/CLI . . . 4
4.3 Return codes . . . 8
4.4 Diagnostics areas in SQL/CLI . . . 9
4.4.1 Introduction to diagnostics areas in SQL/CLI . . . 9
4.4.2 Setting of ROW_NUMBER and COLUMN_NUMBER fields . . . 12
4.5 Miscellaneous characteristics . . . 12
4.5.1 Handles . . . 12
4.5.2 Null-terminated strings . . . 12
4.5.3 Null pointers . . . 13
4.5.4 Environment attributes . . . 13
4.5.5 Connection attributes . . . 13
4.5.6 Statement attributes . . . 14
4.5.7 CLI descriptor areas . . . 14
4.5.8 Obtaining diagnostics during multi-row fetch . . . 15
4.6 SQL-invoked routines . . . 16
4.6.1 Result sets returned by SQL-invoked procedures . . . 16
4.7 Cursors . . . 16
4.7.1 General description of cursors . . . 16
4.8 Client-server operation . . . 16
5 Lexical elements . . . 17
5.1 <token> and <separator> . . . 17
6 Call-Level Interface specifications . . . 18
6.1 <CLI routine> . . . 18
6.2 <CLI routine> invocation . . . 26
6.3 Implicit set connection . . . 29
6.4 Preparing a statement . . . 30
6.5 Executing a statement . . . 33
6.6 Implicit CLI prepared cursor . . . 35
6.7 Implicit CLI procedural result cursor . . . 37

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

6.8 Initial CLI cursor. . . . 38
6.9 Implicit DESCRIBE USING clause. . . . 39
6.10 Implicit EXECUTE USING and OPEN USING clauses. . . . 45
6.11 Implicit CALL USING clause. . . . 51
6.12 Fetching a rowset. . . . 56
6.13 Implicit FETCH USING clause. . . . 60
6.14 Character string retrieval. . . . 66
6.15 Binary string retrieval. . . . 68
6.16 Deferred parameter check. . . . 70
6.17 Description of CLI item descriptor areas. . . . 71
6.18 Other tables associated with CLI. . . . 82
6.19 SQL/CLI data type correspondences. . . . 106
7 SQL/CLI routines. . . . 116
7.1 Introduction to SQL/CLI routines. . . . 116
7.2 AllocConnect(). . . . 116
7.3 AllocEnv(). . . . 117
7.4 AllocHandle(). . . . 118
7.5 AllocStmt(). . . . 122
7.6 BindCol(). . . . 123
7.7 BindParameter(). . . . 125
7.8 Cancel(). . . . 129
7.9 CloseCursor(). . . . 131
7.10 ColAttribute(). . . . 132
7.11 ColumnPrivileges(). . . . 134
7.12 Columns(). . . . 140
7.13 Connect(). . . . 149
7.14 CopyDesc(). . . . 153
7.15 DataSources(). . . . 154
7.16 DescribeCol(). . . . 156
7.17 Disconnect(). . . . 158
7.18 EndTran(). . . . 160
7.19 Error(). . . . 164
7.20 ExecDirect(). . . . 166
7.21 Execute(). . . . 167
7.22 Fetch(). . . . 168
7.23 FetchScroll(). . . . 169
7.24 ForeignKeys(). . . . 170
7.25 FreeConnect(). . . . 182
7.26 FreeEnv(). . . . 183
7.27 FreeHandle(). . . . 184
7.28 FreeStmt(). . . . 187
7.29 GetConnectAttr(). . . . 189
7.30 GetCursorName(). . . . 191
7.31 GetData(). . . . 192
7.32 GetDescField(). . . . 198
7.33 GetDescRec(). . . . 200
7.34 GetDiagField(). . . . 202

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

7.35 GetDiagRec() . . . 211
7.36 GetEnvAttr() . . . 213
7.37 GetFeatureInfo() . . . 215
7.38 GetFunctions() . . . 218
7.39 GetInfo() . . . 219
7.40 GetLength() . . . 223
7.41 GetParamData() . . . 225
7.42 GetPosition() . . . 231
7.43 GetSessionInfo() . . . 233
7.44 GetStmtAttr() . . . 235
7.45 GetSubString() . . . 238
7.46 GetTypeInfo() . . . 240
7.47 MoreResults() . . . 244
7.48 NextResult() . . . 245
7.49 NumResultCols() . . . 246
7.50 ParamData() . . . 247
7.51 Prepare() . . . 252
7.52 PrimaryKeys() . . . 253
7.53 PutData() . . . 258
7.54 RowCount() . . . 261
7.55 SetConnectAttr() . . . 262
7.56 SetCursorName() . . . 264
7.57 SetDescField() . . . 266
7.58 SetDescRec() . . . 271
7.59 SetEnvAttr() . . . 273
7.60 SetStmtAttr() . . . 275
7.61 SpecialColumns() . . . 279
7.62 StartTran() . . . 286
7.63 TablePrivileges() . . . 288
7.64 Tables() . . . 293

8 Additional data manipulation rules . . . 300
8.1 Effect of opening a cursor . . . 300

9 Dynamic SQL . . . 301
9.1 <preparable dynamic cursor name> . . . 301

10 Status codes . . . 302
10.1 SQLSTATE . . . 302

11 Conformance . . . 305
11.1 Claims of conformance to SQL/CLI . . . 305
11.2 Additional conformance requirements for SQL/CLI . . . 305
11.3 Implied feature relationships of SQL/CLI . . . 305

Annex A (informative) **SQL conformance summary . . . 306**

Annex B (informative) **Implementation-defined elements . . . 308**

Annex C (informative) **Implementation-dependent elements . . . 329**

Annex D (informative) **SQL optional feature taxonomy . . . 334**

Annex E (informative) **Deprecated features . . . 335**

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-3:2023

Annex F (informative) **Incompatibilities with ISO/IEC 9075:2016** **336**
Annex G (informative) **Defect Reports not addressed in this edition of this document** **337**
Annex H (informative) **Example header files** **338**
H.1 C header file sqlcli.h 338
H.2 COBOL library item SQLCLI 349
Annex I (informative) **Example C programs** **357**
I.1 Introduction to Example C programs 357
I.2 Create table, insert, select 357
I.3 Interactive Query 360
I.4 Providing long dynamic arguments at Execute time 363
Index **366**

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

Tables

Table

Page

1 Header fields in SQL/CLI diagnostics areas. . . . 10
2 Status record fields in SQL/CLI diagnostics areas. . . . 10
3 Supported calling conventions of SQL/CLI routines by language. . . . 21
4 Abbreviated SQL/CLI generic names. . . . 21
5 Fields in SQL/CLI row and parameter descriptor areas. . . . 76
6 Codes used for implementation data types in SQL/CLI. . . . 78
7 Codes used for application data types in SQL/CLI. . . . 79
8 Codes associated with datetime data types in SQL/CLI. . . . 80
9 Codes associated with <interval qualifier> in SQL/CLI. . . . 80
10 Codes associated with <parameter mode> in SQL/CLI. . . . 81
11 Codes associated with user-defined types in SQL/CLI. . . . 81
12 Codes used for SQL/CLI diagnostic fields. . . . 82
13 Codes used for SQL/CLI handle types. . . . 83
14 Codes used for transaction termination. . . . 84
15 Codes used for environment attributes. . . . 84
16 Codes used for connection attributes. . . . 84
17 Codes used for statement attributes. . . . 84
18 Codes used for FreeStmt options. . . . 85
19 Data types of attributes. . . . 85
20 Codes used for SQL/CLI descriptor fields. . . . 86
21 Ability to set SQL/CLI descriptor fields. . . . 88
22 Ability to retrieve SQL/CLI descriptor fields. . . . 90
23 SQL/CLI descriptor field default values. . . . 93
24 Codes used for fetch orientation. . . . 95
25 Multi-row fetch status codes. . . . 95
26 Miscellaneous codes used in CLI. . . . 95
27 Codes used to identify SQL/CLI routines. . . . 96
28 Codes and data types for implementation information. . . . 99
29 Codes and data types for session implementation information. . . . 101
30 Values for TRANSACTION ISOLATION OPTION with StartTran. . . . 101
31 Values for TRANSACTION ACCESS MODE with StartTran. . . . 101
32 Codes used for concise data types. . . . 101
33 Codes used with concise datetime data types in SQL/CLI. . . . 103
34 Codes used with concise interval data types in SQL/CLI. . . . 104
35 Concise codes used with datetime data types in SQL/CLI. . . . 104
36 Concise codes used with interval data types in SQL/CLI. . . . 104
37 Special parameter values. . . . 105
38 Column types and scopes used with SpecialColumns. . . . 105
39 SQL/CLI data type correspondences for Ada. . . . 106
40 SQL/CLI data type correspondences for C. . . . 107
41 SQL/CLI data type correspondences for COBOL. . . . 108
42 SQL/CLI data type correspondences for Fortran. . . . 110
43 SQL/CLI data type correspondences for M. . . . 111
44 SQL/CLI data type correspondences for Pascal. . . . 112
45 SQL/CLI data type correspondences for PL/I. . . . 114
46 SQLSTATE class and subclass codes. . . . 302
47 Implied feature relationships of SQL/CLI. . . . 305

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-3:2023

A.1 Feature definitions outside of Conformance Rules. 306
D.1 Feature taxonomy for optional features. 334

© ISO/IEC 2023 – All rights reserved
Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-3:2023

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC have not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

This sixth edition cancels and replaces the fifth edition (ISO/IEC 9075-3:2016), which has been technically revised. It also incorporates the Technical Corrigendum ISO/IEC 9075-3:2016/Cor.1:2022.

The main changes are as follows:

— improve the presentation and accuracy of the summaries of implementation-defined and implementation-dependent aspects of this document;

— introduction of several digital artifacts;

— alignment with updated ISO house style and other guidelines for creating standards.

This sixth edition of ISO/IEC 9075-3 is designed to be used in conjunction with the following editions of other parts of the ISO/IEC 9075 series, all published 2023:

— ISO/IEC 9075-1, sixth edition;

— ISO/IEC 9075-2, sixth edition;

— ISO/IEC 9075-4, seventh edition;

Documents
Document Sharing Hub

© ISO/IEC 2023 – All rights reserved

Full document details: BS ISO/IEC 9075-3:2023

— ISO/IEC 9075-9, fifth edition;

— ISO/IEC 9075-10, fifth edition;

— ISO/IEC 9075-11, fifth edition;

— ISO/IEC 9075-13, fifth edition;

— ISO/IEC 9075-14, sixth edition;

— ISO/IEC 9075-15, second edition;

— ISO/IEC 9075-16, first edition.

A list of all parts in the ISO/IEC 9075 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Documents
Document Sharing Hub

© ISO/IEC 2023 – All rights reserved

Full document details: BS ISO/IEC 9075-3:2023

Introduction

The organization of this document is as follows:

1) Clause 1, "Scope", specifies the scope of this document.

2) Clause 2, "Normative references", identifies additional standards that, through reference in this document, constitute provisions of this document.

3) Clause 3, "Terms and definitions", defines the terms and definitions used in this document.

4) Clause 4, "Concepts", presents concepts used in the definition of the Call-Level Interface.

5) Clause 5, "Lexical elements", defines the lexical elements of the language.

6) Clause 6, "Call-Level Interface specifications", defines facilities for using SQL through a Call-Level Interface.

7) Clause 7, "SQL/CLI routines", defines each of the routines that comprise the Call-Level Interface.

8) Clause 8, "Additional data manipulation rules", defines additional rules for data manipulation.

9) Clause 9, "Dynamic SQL", defines the SQL dynamic statements.

10) Clause 10, "Status codes", defines values that identify the status of the execution of SQL-statements and the mechanisms by which those values are returned.

11) Clause 11, "Conformance", defines the criteria for conformance to this document.

12) Annex A, "SQL conformance summary", is an informative Annex. It summarizes the conformance requirements of the SQL language.

13) Annex B, "Implementation-defined elements", is an informative Annex. It lists those features for which the body of this document states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or other aspect is partly or wholly implementation-defined.

14) Annex C, "Implementation-dependent elements", is an informative Annex. It lists those features for which the body of this document states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or other aspect is partly or wholly implementation-dependent.

15) Annex D, "SQL optional feature taxonomy", is an informative Annex. It identifies the optional features of the SQL language specified in this document by an identifier and a short descriptive name. This taxonomy is used to specify conformance.

16) Annex E, "Deprecated features", is an informative Annex. It lists features that the responsible Technical Committee intends not to include in a future edition of this document.

17) Annex F, "Incompatibilities with ISO/IEC 9075:2016", is an informative Annex. It lists incompatibilities with the previous edition of this document.

18) Annex G, "Defect Reports not addressed in this edition of this document", is an informative Annex. It describes the Defect Reports that were known at the time of publication of this document. Each of these problems is a problem carried forward from the previous edition of document. No new problems have been created in the drafting of this edition of this document.

19) Annex H, "Example header files", is an informative Annex. It provides examples of typical definition files for application programs using the SQL Call-Level Interface.

20) Annex I, "Example C programs", is an informative Annex. It provides examples of using the SQL Call-Level Interface in the C programming language.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 9075-3:2023

In the text of this document, Clauses and Annexes begin new odd-numbered pages, and in Clause 6, "Call-Level Interface specifications", through Clause 11, "Conformance", Subclauses begin new pages. Any resulting blank space is not significant.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

Information technology — Database language SQL —

Part 3:
Call-Level Interface (SQL/CLI)

1 Scope

This document defines the structures and procedures that can be used to execute statements of the database language SQL from within an application written in a programming language in such a way that procedures used are independent of the SQL statements to be executed.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 1539-1:2018, *Information technology — Programming languages — Fortran — Part 1: Base language*

ISO/IEC 1539-2:2000, *Information technology — Programming languages — Fortran — Part 2: Varying length character strings*

ISO 1989:2014, *Information technology — Programming languages — COBOL*

ISO 6160:1979, *Programming languages — PL/I (Endorsement of ANSI X3.53-1976)*

ISO 7185:1990, *Information technology — Programming languages — Pascal*

ISO/IEC 8652:2012, *Information technology — Programming languages — Ada*

ISO/IEC 8652:2012/Cor.1:2016, *Information technology — Programming languages — Ada — Technical Corrigendum 1*

ISO/IEC 9075-1, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*

ISO/IEC 9075-2, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*

ISO/IEC 9075-11, *Information technology — Database languages — SQL — Part 11: Information and Definition Schemas (SQL/Schemata)*

ISO/IEC 9899:2018, *Information technology — Programming languages — C*

ISO/IEC 10206:1991, *Information technology — Programming languages — Extended Pascal*

ISO/IEC 11756:1999, *Information technology — Programming languages — M*

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 9075-1, ISO/IEC 9075-2 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
data source
SQL-server that is part of the current SQL-connection

3.2
handle
CLI object returned by an SQL/CLI implementation when a CLI resource is allocated and used by an SQL/CLI application to reference that CLI resource

3.3
pseudo-column
column that is part of a table but is not part of the descriptor for that table

Note 1 to entry: An example of such a pseudo-column is an implementation-defined row identifier.

3.4
rowset
one or more rows retrieved in a single invocation of the Fetch and FetchScroll routines

3.5
SQL/CLI application
application that invokes <CLI routine>s specified in this document

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

4 Concepts

This Clause modifies Clause 4, "Concepts", in ISO/IEC 9075-2.

4.1 Notations and conventions

This Subclause modifies Subclause 4.1, "Notations and conventions", in ISO/IEC 9075-2.

4.1.1 Notations

This Subclause modifies Subclause 4.1.1, "Notations", in ISO/IEC 9075-2.

The notations used in this document are defined in ISO/IEC 9075-1.

The syntax defined in this document is available from the ISO website as a "digital artifact". See `https://standards.iso.org/iso-iec/9075/-3/ed-6/en/` to download digital artifacts for this document. To download the syntax defined in a plain-text format, select the file named `ISO_IEC_9075-3(E)_CLI.bnf.txt`. To download the syntax defined in an XML format, select the file named `ISO_IEC_9075-3(E)_CLI.bnf.xml`.

4.1.2 Specification of routine definitions

The routines in this document are specified in terms of the following characteristics.

— **Function**: A short statement of the purpose of the routine.

— **Definition**: The name of the routine and the name, mode, and data type of each of its parameters.

— **General Rules**: A specification of the run-time effect of the routine. Where more than one General Rule is used to specify the effect of a routine, the required effect is that which would be obtained by beginning with the first General Rule and applying the Rules in numeric sequence until a Rule is applied that specifies or implies a change in sequence or termination of the application of the Rules. Unless otherwise specified or implied by a specific Rule that is applied, application of General Rules terminates when the last in the sequence has been applied.

4.2 Introduction to SQL/CLI

This Subclause is modified by Subclause 4.11, "Introduction to SQL/CLI", in ISO/IEC 9075-9.

The Call-Level Interface (SQL/CLI) is a binding style for executing SQL statements. This document provides specifications for routines that:

— allocate and deallocate resources;

— control connections to SQL-servers;

— execute SQL statements using mechanisms similar to dynamic SQL;

— obtain diagnostic information;

— control transaction termination;

— obtain information about the SQL/CLI implementation and the SQL-implementation.

A *handle* is a CLI object returned by an SQL/CLI implementation when a CLI resource is allocated; the handle is used by an SQL/CLI application to reference that CLI resource. The AllocHandle routine allocates

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

the resources to manage an SQL-environment, an SQL-connection, a CLI descriptor area, or SQL-statement processing; when invoked, it returns an environment handle, a connection handle, a descriptor handle, or a statement handle, respectively. An SQL-connection is allocated in the context of an allocated SQL-environment. CLI descriptor areas and SQL-statements are allocated in the context of an allocated SQL-connection. The FreeHandle routine deallocates a specified resource. The AllocConnect, AllocEnv, and AllocStmt routines can be used to allocate the resources to manage an SQL-connection, an SQL-environment, and SQL-statement processing, respectively, instead of using the AllocHandle routine. The FreeConnect, FreeEnv, and FreeStmt routines can be used to deallocate the specific resource instead of using FreeHandle.

Each allocated SQL-environment has an attribute that determines whether output character strings are null-terminated by the SQL/CLI implementation. The SQL/CLI application can set the value of this attribute by using the routine SetEnvAttr and can retrieve the current value of the attribute by using the routine GetEnvAttr.

The Connect routine establishes an SQL-connection, which becomes the *current SQL-connection*. The Disconnect routine terminates an established SQL-connection. Switching between established SQL-connections occurs automatically whenever the SQL/CLI application switches processing to a dormant SQL-connection, which then becomes the *current SQL-connection*.

The ExecDirect routine is used for a one-time execution of an SQL-statement. The Prepare routine is used to prepare an SQL-statement for subsequent execution using the Execute routine. In all three cases, the executed SQL-statement can contain dynamic parameters.

The interface for a description of dynamic parameters, dynamic parameter values, the result columns of a <dynamic select statement> or <dynamic single row select statement>, and the target specifications for the result columns is a CLI descriptor area. A CLI descriptor area for each type of interface is automatically allocated when an SQL-statement is allocated. The SQL/CLI application may allocate additional CLI descriptor areas and nominate them for use as the interface for the description of dynamic parameter values or the description of target specifications by using the routine SetStmtAttr. The SQL/CLI application can determine the handle value of the CLI descriptor area currently being used for a specific interface by using the routine GetStmtAttr. The GetDescField and GetDescRec routines enable information to be retrieved from a CLI descriptor area. The CopyDesc routine enables the contents of a CLI descriptor area to be copied to another CLI descriptor area.

When a <dynamic select statement> or <dynamic single row select statement> is prepared or executed immediately, a description of the result columns is automatically provided in the applicable CLI implementation descriptor area. In this case, the SQL/CLI application may additionally retrieve information by using the DescribeCol and/or the ColAttribute routine to obtain a description of a single result column and by using the NumResultCols routine to obtain a count of the number of result columns. The SQL/CLI application sets values in the CLI application descriptor area for the description of the corresponding target specifications either explicitly, by using the routines SetDescField and SetDescRec, or implicitly, by using the routine BindCol.

When an SQL-statement is prepared or executed immediately, a description of the dynamic parameters is automatically provided in the applicable CLI implementation descriptor area if this facility is supported by the current SQL-connection. An attribute associated with the allocated SQL-connection indicates whether this facility is supported. The value of the attribute may be retrieved using the routine GetConnectAttr. Regardless of whether automatic description is supported, all dynamic input and input/output parameters shall be defined in the application descriptor area before SQL-statement execution. This can be done either explicitly, by using the routines SetDescField and SetDescRec, or implicitly, by using the routine BindParameter. The value of a dynamic input or input/output parameter may be established before SQL-statement execution (immediate parameter value) or may be provided during SQL-statement execution (deferred parameter value). Its description in the CLI descriptor area determines which method is in use. The ParamData routine is used to cycle through and process deferred input and input/output parameter values. The PutData routine is used to provide the deferred values. The PutData routine also enables the values of character string input and input/output parameters to be provided piece by piece.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

Before a <call statement> is prepared or executed immediately, the SQL/CLI application may choose whether or not to bind dynamic output parameters in the CLI application descriptor area. This can be done either explicitly, by using the routines SetDescField and SetDescRec, or implicitly, by using the routine BindParameter. After execution of the statement, values of unbound output and input/output parameters can be individually retrieved using the GetParamData routine. The GetParamData routine also enables the retrieval of the values of character and binary string output and input/output parameters to be accomplished piece by piece.

When a <dynamic select statement> or <dynamic single row select statement> is executed, a CLI prepared cursor is implicitly declared and opened. The name of the cursor is determined by the cursor name property associated with the allocated SQL-statement, which can be supplied by the SQL/CLI application by using the routine SetCursorName. If a cursor name is not supplied by the SQL/CLI application, the value of the cursor name property associated with the allocated SQL-statement is an implementation-dependent (UV124) cursor name. The cursor name property associated with the allocated SQL-statement can be retrieved by using the GetCursorName routine. The operational sensitivity, scrollability, and holdability properties of a CLI prepared cursor are determined by the CURSOR SENSITIVITY, CURSOR SCROLLABLE, and CURSOR HOLDABLE attributes, respectively, of the allocated SQL-statement at the time the CLI cursor is declared and opened. The SQL/CLI application can set the values of these attributes by using the SetStmtAttr routine and can retrieve the current values of these attributes by using the GetStmtAttr routine. The operational returnability property of a CLI prepared cursor is implementation-defined (IV031).

The Fetch and FetchScroll routines are used to position an open CLI cursor on a row and to retrieve the values of bound columns for that row. A bound column is one whose target specification in the specified CLI descriptor area defines a location for the target value. The Fetch routine always positions the open CLI cursor on the next row, whereas the FetchScroll routine may be used to position the open CLI cursor on a specified row. The use of FetchScroll with a FetchOrientation other than NEXT is permitted only if the operational scrollability property of the CLI cursor is SCROLL. The Fetch and FetchScroll routines can also retrieve multiple rows in a single call; the set of rows thus retrieved is called a *rowset*. This is accomplished by setting the ARRAY_SIZE field of the applicable application row descriptor to the desired number of rows. Note that the single row fetch is just a special case of multi-row fetch, where the rowset size is 1 (one).

Values for unbound columns can be individually retrieved by using the GetData routine. The GetData routine also enables the retrieval of the values of character and binary string columns to be accomplished piece by piece. The current row of a CLI cursor is a row of the current rowset indicated by the CURRENT OF POSITION attribute of the allocated SQL-statement associated with the CLI cursor. The current row can be deleted or updated by executing a <preparable dynamic delete statement: positioned> or a <preparable dynamic update statement: positioned>, respectively, for that CLI cursor under a different allocated SQL-statement to the one under which the CLI cursor was opened. The CloseCursor routine enables a CLI cursor to be closed.

Result sets can be returned to the SQL/CLI application as a result of invoking the Execute or ExecDirect routine, supplying a statement handle whose current statement is a <call statement>. If the <call statement> invokes an SQL-invoked procedure *SIP* that returns a non-empty result set sequence *RSS*, then a CLI procedural result cursor is automatically associated with the statement handle. The result set of this CLI procedural result cursor is the first result set of RSS. The SQL/CLI application can learn that a cursor has been automatically opened by invoking NumResultCols to determine if the ColumnCount is positive. If there is more than one result set in the result set sequence, then the others can be processed one at a time or in parallel. To process the result sets one at a time, once the processing of a given result set is complete, the MoreResults routine is used to determine whether there are additional result sets and, if there are, to position the CLI procedural result cursor before the first row in the next result set. To process the result sets in parallel, the NextResult routine is used to determine whether there are additional result sets and, if there are, to position a CLI procedural result cursor associated with another statement handle before the first row in the next result set.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

When a CLI procedural result cursor is associated with a result set, the operational sensitivity, scrollability, and holdability properties of the CLI procedural result cursor are those of the result set as it was received from the stored procedure. (The CURSOR SENSITIVITY, CURSOR SCROLLABLE, and CURSOR HOLDABLE attributes of the allocated SQL-statement are ignored; using SetStmtAttr to set these attributes has no effect on the corresponding operational properties of a CLI procedural result cursor.) The operational returnability property of a CLI procedural result cursor is implementation-defined (IV031). A CLI procedural result cursor is not updatable. Otherwise, a CLI procedural result cursor is processed in the same way as a CLI prepared cursor.

Special routines, called *catalog routines* are available to return result sets from the Information Schema. These routines are described here.

— ColumnPrivileges: Returns a list of the privileges held on the columns whose names adhere to the requested pattern(s) within a single specified table. Most of this information can also be obtained by using the ExecDirect routine to issue an appropriate query on the COLUMN_PRIVILEGES view of the Information Schema.

— Columns: Returns the column names and attributes for all columns whose names adhere to the requested pattern(s). Most of this information can also be obtained by using the ExecDirect routine to issue an appropriate query on the COLUMNS view of the Information Schema.

— ForeignKeys: Returns either the primary key of a single specified table together with the foreign keys in all other tables that reference that primary key or the foreign keys of a single specified table together with all the primary and unique keys in all other tables that are referenced by those foreign keys. Most of this information can also be obtained by using the ExecDirect routine to issue an appropriate query on the TABLE_CONSTRAINTS view and the REFERENTIAL_CONSTRAINTS view of the Information Schema.

— PrimaryKeys: Returns a list of the columns that constitute the primary key of a single specified table. Most of this information can also be obtained by using the ExecDirect routine to issue an appropriate query on the TABLE_CONSTRAINTS view and the KEY_COLUMN_USAGE view of the Information Schema.

— SpecialColumns: Returns a list of the columns that can uniquely identify a particular row within a single specified table. Most of this information can also be obtained by using the ExecDirect routine to issue an appropriate query on the COLUMNS view of the Information Schema.

— Tables: Returns information about the tables whose names adhere to the requested pattern(s) and type(s). Most of this information can also be obtained by using the ExecDirect routine to issue an appropriate query on the TABLES view of the Information Schema.

— TablePrivileges: Returns a list of the privileges held on tables whose names adhere to the requested pattern(s). Most of this information can also be obtained by using the ExecDirect routine to issue an appropriate query on the TABLE_PRIVILEGES view of the Information Schema.

These special routines are only available for a small portion of the metadata that is available in the Information Schema. Other metadata (for example, that about SQL-invoked routines, triggers, and user-defined types) can be obtained by executing appropriate queries on the views of the Information Schema.

The GetPosition, GetLength, and GetSubString routines can each be used with its own independent statement handle to access a string value at the server that is represented by a Large Object locator in order to do any of the following.

— The GetPosition routine may be used to determine whether a given substring exists within that string and, if it does, to obtain an integer value that indicates the starting position of the first appearance of the given substring.

— The GetLength routine may be used to obtain the length of that string as an integer.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

— The GetSubString routine may be used to retrieve a portion of a string, or alternatively, to create a new Large Object value at the server which is a portion of the string and to return a Large Object locator that represents that value.

The Error, GetDiagField, and GetDiagRec routines obtain diagnostic information about the most recent routine operating on a particular resource. The Error routine always retrieves information from the next status record, whereas the GetDiagField and GetDiagRec routines may be used to retrieve information from an identified status record.

The number of rows affected by the last executed SQL-statement can be obtained by using the RowCount or GetDiagField routine.

An SQL-transaction is terminated by using the EndTran routine. An SQL-transaction is implicitly initiated whenever a CLI routine is invoked that requires the context of an SQL-transaction and no SQL-transaction is active. An SQL-transaction is explicitly started, and its characteristics set, by using the StartTran routine.

NOTE 1 — Applications are prohibited from using the ExecDirect or Execute routines to execute <start transaction statement>s, <commit statement>s, <rollback statement>s, and <release savepoint statement>s.

The Cancel routine is used to cancel the execution of a concurrently executing SQL/CLI routine; it is also used to terminate the processing of deferred parameter values and the execution of the associated SQL-statement.

09 The GetFeatureInfo, GetFunctions, GetInfo, GetSessionInfo, and GetTypeInfo routines are used to obtain information about the SQL/CLI implementation. The DataSources routine returns a list of names that identify SQL-servers to which the SQL/CLI application may be able to connect and returns a description of each such SQL-server.

4.3 Return codes

The execution of a CLI routine causes one or more conditions to be raised. The status of the execution is indicated by a code that is returned either as the result of invoking a CLI routine that is a CLI function or as the value of the ReturnCode argument of a CLI routine that is a CLI procedure.

The return code values and meanings are described in the following list. If more than one return code is possible, then the one appearing later in the list is the one returned.

— A value of 0 (zero) indicates **Success**. The CLI routine executed successfully.

— A value of 1 (one) indicates **Success with information**. The CLI routine executed successfully but a completion condition was raised: *warning (01000)*.

— A value of 100 indicates **No data found**. The CLI routine executed successfully but a completion condition was raised: *no data (02000)*.

— A value of 99 indicates **Data needed**. The CLI routine did not complete its execution because additional data is needed. An exception condition was raised: *CLI-specific condition — dynamic parameter value needed (HYHHG)*.

— A value of −1 (negative one) indicates **Error**. The CLI routine did not execute successfully. An exception condition other than *CLI-specific condition — invalid handle (HYHHH)* or *CLI-specific condition — dynamic parameter value needed (HYHHG)* was raised.

— A value of −2 indicates **Invalid handle**. The CLI routine did not execute successfully because an exception condition was raised: *CLI-specific condition — invalid handle (HYHHH)*.

After the execution of a CLI routine, the values of every output argument that corresponds to an output parameter whose value is not explicitly defined by this document is implementation-dependent (UV052).

In addition to providing the return code, for all CLI routines other than Error, GetDiagField, and GetDiagRec, the SQL/CLI implementation records information about completion conditions and about exception

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

conditions other than *CLI-specific condition — invalid handle (HYHHH)* in the diagnostics area associated with the resource being utilized. The *resource being utilized* by a routine is the resource identified by its input handle. In the case of CopyDesc, which takes two input handles, the resource being utilized is the one identified by TargetDescHandle.

4.4 Diagnostics areas in SQL/CLI

4.4.1 Introduction to diagnostics areas in SQL/CLI

Each diagnostics area comprises header information consisting of fields that contain general information relating to the routine that was executed and zero (0) or more status records containing information about individual conditions that occurred during the execution of the CLI routine. A condition that causes a status record to be generated is referred to as a *status condition*.

At the beginning of the execution of each CLI routine other than Error, GetDiagField, and GetDiagRec, the diagnostics area for the resource being utilized is emptied. If the execution of such a routine does not result in the exception condition *CLI-specific condition — invalid handle (HYHHH)* or the exception condition *CLI-specific condition — dynamic parameter value needed (HYHHG)*, then:

— header information is generated in the diagnostics area;

— if the routine's return code indicates **Success**, then no status records are generated;

— if the routine's return code indicates **Success with information** or **Error**, then one or more status records are generated;

— if the routine's return code indicates **No data found**, then no status record is generated corresponding to SQLSTATE value '02000' but there may be status records generated corresponding to SQLSTATE value '02*nnn*', where '*nnn*' is an implementation-defined (IC001) subclass code.

When Fetch or FetchScroll is invoked, the resulting rowset has one or more rows, and exceptions or warnings are generated, then the corresponding records in the diagnostics area have the ROW_NUMBER field set to the row number of the row in the rowset associated with the exceptions or warnings. If a status record does not correspond to any row in the rowset, or the record is generated as a result of calling a routine other than Fetch or FetchScroll, the ROW_NUMBER field is set to zero. The COLUMN_NUMBER field of the status record contains the column number (if any) to which this exception or warning condition applies. If the status record does not apply to any column, then COLUMN_NUMBER is set to zero.

Status records in the diagnostics area are ordered by ROW_NUMBER. If multiple status records are generated for the same ROW_NUMBER value, then the order in which the second and subsequent of those status records appear is implementation-dependent (US025). Which of those status records appears first is also implementation-dependent (US025), except that:

— status records corresponding to *transaction rollback (40000)* have precedence over status records corresponding to other exceptions, which in turn have precedence over status records corresponding to the completion condition *no data (02000)*, which in turn have precedence over status records corresponding to the completion condition *warning (01000)*;

— apart from status records corresponding to an implementation-defined (IC001) *no data (02000)*, status records corresponding to an implementation-defined (IC001) condition that duplicates, in whole or in part, a condition defined in this document shall not be the first status record.

The routines GetDiagField and GetDiagRec retrieve information from a diagnostics area. The SQL/CLI application identifies which diagnostics area is to be accessed by providing the handle of the relevant resource as an input argument. The routines return a result code but do not modify the identified diagnostics area.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

The Error routine also retrieves information from a diagnostics area. The Error routine retrieves the status records in the identified diagnostics area one at a time but does not permit already processed status records to be retrieved. Error returns a result code but does not modify the identified diagnostics area.

The RowCount routine retrieves the ROW_COUNT field from the diagnostics area for the specified statement handle. RowCount returns a result code and may cause status records to be generated.

A CLI diagnostics area comprises the header fields specified under "Header fields" Table 1, "Header fields in SQL/CLI diagnostics areas", as well as zero (0) or more status records, each of which comprises the fields specified under "Status record fields" Table 2, "Status record fields in SQL/CLI diagnostics areas".

Table 1 — Header fields in SQL/CLI diagnostics areas

Field	Data type
DYNAMIC_FUNCTION	CHARACTER VARYING (*L1*) [1]
DYNAMIC_FUNCTION_CODE	INTEGER
MORE	INTEGER
NUMBER	INTEGER
RETURNCODE	SMALLINT
ROW_COUNT	INTEGER
TRANSACTIONS_COMMITTED	INTEGER
TRANSACTIONS_ROLLED_BACK	INTEGER
TRANSACTION_ACTIVE	INTEGER
implementation-defined (IE017) header field	implementation-defined (IE017) data type

[1] **Where** *L1* is an implementation-defined (IL035) integer not less than 254.

Table 2 — Status record fields in SQL/CLI diagnostics areas

Field	Data type
CATALOG_NAME	CHARACTER VARYING (*L*) [1]
CLASS_ORIGIN	CHARACTER VARYING (*L1*) [1]
COLUMN_NAME	CHARACTER VARYING (*L*) [1]
COLUMN_NUMBER	INTEGER
CONDITION_IDENTIFIER	CHARACTER VARYING (*L*) [1]
CONDITION_NUMBER	INTEGER

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

Field	Data type
CONNECTION_NAME	CHARACTER VARYING (*L*) [1]
CONSTRAINT_CATALOG	CHARACTER VARYING (*L*) [1]
CONSTRAINT_NAME	CHARACTER VARYING (*L*) [1]
CONSTRAINT_SCHEMA	CHARACTER VARYING (*L*) [1]
CURSOR_NAME	CHARACTER VARYING (*L*) [1]
MESSAGE_LENGTH	INTEGER
MESSAGE_OCTET_LENGTH	INTEGER
MESSAGE_TEXT	CHARACTER VARYING (*L1*) [1]
NATIVE_CODE	INTEGER
PARAMETER_MODE	CHARACTER VARYING (*L*) [1]
PARAMETER_NAME	CHARACTER VARYING (*L*) [1]
PARAMETER_ORDINAL_POSITION	INTEGER
ROUTINE_CATALOG	CHARACTER VARYING (*L*) [1]
ROUTINE_NAME	CHARACTER VARYING (*L*) [1]
ROUTINE_SCHEMA	CHARACTER VARYING (*L*) [1]
ROW_NUMBER	INTEGER
SCHEMA_NAME	CHARACTER VARYING (*L*) [1]
SERVER_NAME	CHARACTER VARYING (*L*) [1]
SQLSTATE	CHARACTER (5)
SPECIFIC_NAME	CHARACTER VARYING (*L*) [1]
SUBCLASS_ORIGIN	CHARACTER VARYING (*L1*) [1]
TABLE_NAME	CHARACTER VARYING (*L*) [1]
TRIGGER_CATALOG	CHARACTER VARYING (*L*) [1]
TRIGGER_NAME	CHARACTER VARYING (*L*) [1]
TRIGGER_SCHEMA	CHARACTER VARYING (*L*) [1]

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

Field	Data type
implementation-defined (IE018) status field	implementation-defined (IE018) data type
[1] **Where** *L* is an implementation-defined (IL035) integer not less than 128 and *L1* is an implementation-defined (IL035) integer not less than 254.	

All diagnostics area fields specified in other parts of the ISO/IEC 9075 series that are not included in this table are not applicable to SQL/CLI.

4.4.2 Setting of ROW_NUMBER and COLUMN_NUMBER fields

Except where otherwise specified in this document, the ROW_NUMBER and COLUMN_NUMBER fields in a status record are always 0 (zero).

4.5 Miscellaneous characteristics

4.5.1 Handles

The AllocHandle routine returns a handle that uniquely identifies the allocated resource. Although the data type of a handle parameter is INTEGER, its value has no meaning in any other context and should not be used as a numeric operand or modified in any way.

In general, if the related resource cannot be allocated, then a handle value of zero is returned. However, even if a resource has been successfully allocated, processing of that resource can subsequently fail due to memory constraints as follows:

— if additional memory is required but is not available, then an exception condition is raised: *CLI-specific condition — memory allocation error (HY001)*;

— if previously allocated memory cannot be accessed, then an exception condition is raised: *CLI-specific condition — memory management error (HY013)*.

NOTE 2 — No diagnostic information is generated in this case.

The validity of a handle in a compilation unit other than the one in which the identified resource was allocated is implementation-defined (IA139).

Specifying (the address of) a valid handle as the output handle for an invocation of AllocHandle does not have the effect of reinitializing the identified resource. Instead, a new resource is allocated and a new handle value overwrites the old one.

4.5.2 Null-terminated strings

An input character string provided by the SQL/CLI application may be terminated by the implementation-defined (IV030) null character that terminates C character strings. If this technique is used, the application may set the associated length argument to either the length of the string excluding the null terminator or to −3, indicating NULL TERMINATED.

If the NULL TERMINATION attribute for the SQL-environment is *True*, then all output character strings returned by the SQL/CLI implementation are terminated by the implementation-defined (IV030) null character that terminates C character strings. If the NULL TERMINATION attribute is *False*, then output character strings are not null-terminated.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

4.5.3 Null pointers

If the programming language of the invoking SQL/CLI application supports pointers, then the SQL/CLI application may provide a zero-valued pointer, referred to as a null pointer, in the following circumstances.

— In lieu of an output argument that is to receive the length of a returned character string. This indicates that the SQL/CLI application wishes to prohibit the return of this information.

— In lieu of other output arguments where specifically allowed by this document. This indicates that the SQL/CLI application wishes to prohibit the return of this information.

— In lieu of input arguments where specifically allowed by this document. The semantics of such a specification depend on the context.

If the SQL/CLI application provides a null pointer in any other circumstances, then an exception condition is raised: *CLI-specific condition — invalid use of null pointer (HY009)*.

If the NULL TERMINATION attribute for the SQL-environment is *False*, then specifying a zero buffer size for an output argument is equivalent to specifying a null pointer for that output argument.

4.5.4 Environment attributes

Environment attributes are associated with each allocated SQL-environment and affect the behavior of CLI functions in that SQL-environment.

The GetEnvAttr routine enables the SQL/CLI application to determine the current value of a specific attribute. For attributes that may be set by the user, the SetEnvAttr routine enables the SQL/CLI application to set the value of a specific attribute. Attribute values may be set by the SQL/CLI application whenever there are no SQL-connections allocated within the SQL-environment.

Table 15, "Codes used for environment attributes", and Table 19, "Data types of attributes", in Subclause 6.18, "Other tables associated with CLI", indicate for each attribute its name, code value, data type, possible values, and whether the attribute may be set using SetEnvAttr.

The NULL TERMINATION attribute determines whether output character strings are null-terminated by the SQL/CLI implementation. The attribute is set to *True* when an SQL-environment is allocated.

4.5.5 Connection attributes

Connection attributes are associated with each allocated SQL-connection and affect the behavior of CLI functions operating in the context of that allocated SQL-connection.

The GetConnectAttr routine enables the SQL/CLI application to determine the current value of a specific connection attribute. For connection attributes that may be set by the user, the SetConnectAttr routine enables the SQL/CLI application to set the value of a specific connection attribute.

Table 16, "Codes used for connection attributes", and Table 19, "Data types of attributes", in Subclause 6.18, "Other tables associated with CLI", indicate for each connection attribute its name, code value, data type, possible values and whether the connection attribute may be set using SetConnectAttr.

The POPULATE IPD attribute determines whether the SQL/CLI implementation will populate the implementation parameter descriptor with an item descriptor area for each <dynamic parameter specification> when an SQL-statement is prepared or executed immediately. The POPULATE IPD attribute is automatically set each time an SQL-connection is established for the allocated SQL-connection.

The SAVEPOINT NAME connection attribute specifies the savepoint to be referenced in an invocation of the EndTran routine that uses the SAVEPOINT NAME ROLLBACK or SAVEPOINT NAME RELEASE CompletionType, respectively. The SAVEPOINT NAME attribute is set to a zero-length string when the SQL-connection is allocated.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

4.5.6 Statement attributes

Statement attributes are associated with each allocated SQL-statement and affect the processing of SQL-statements under that allocated SQL-statement.

The GetStmtAttr routine enables the SQL/CLI application to determine the current value of a specific statement attribute. For statement attributes that may be set by the user, the SetStmtAttr routine enables the SQL/CLI application to set the value of a specific statement attribute.

Table 17, "Codes used for statement attributes", and Table 19, "Data types of attributes", in Subclause 6.18, "Other tables associated with CLI", indicate for each statement attribute its name, code value, data type, possible values, and whether the statement attribute may be set by using SetStmtAttr.

The APD HANDLE statement attribute is the value of the handle of the current application parameter descriptor for the allocated SQL-statement. The statement attribute is set to the value of the handle of the automatically allocated application parameter descriptor when the SQL-statement is allocated.

The ARD HANDLE statement attribute is the value of the handle of the current application row descriptor for the allocated SQL-statement. The statement attribute is set to the value of the handle of the automatically allocated application row descriptor when the SQL-statement is allocated.

The IPD HANDLE statement attribute is the value of the handle of the implementation parameter descriptor associated with the allocated SQL-statement. The statement attribute is set to the value of the handle of the automatically allocated implementation parameter descriptor when the SQL-statement is allocated.

The IRD HANDLE statement attribute is the value of the handle of the implementation row descriptor associated with the allocated SQL-statement. The statement attribute is set to the value of the handle of the automatically allocated implementation row descriptor when the SQL-statement is allocated.

The CURSOR SCROLLABLE statement attribute determines the *scrollability* of the CLI prepared cursor implicitly declared when Execute or ExecDirect are invoked. The statement attribute is set to NONSCROLLABLE when the SQL-statement is allocated.

The CURSOR SENSITIVITY statement attribute determines the *sensitivity* to changes of the CLI prepared cursor implicitly declared when Execute or ExecDirect are invoked. The statement attribute is set to ASENSITIVE when the SQL-statement is allocated.

The CURSOR HOLDABLE statement attribute determines the *holdability* of the CLI prepared cursor implicitly declared when Execute or ExecDirect are invoked. The statement attribute is set to HOLDABLE or NONHOLDABLE when the statement is allocated, depending on the values of the CURSOR COMMIT BEHAVIOR item used by the GetInfo routine.

Whether or not a CLI cursor is returnable is implementation-defined (IA140).

The statement attribute CURRENT OF POSITION identifies the row in the rowset to which a positioned update or delete operation applies. This is set to 1 (one) when an SQL-statement is initially allocated. It is reset to 1 (one) whenever Fetch or FetchScroll are successfully executed when the ARRAY_SIZE is 1 (one) or the cursor is scrollable; otherwise, it is set to an implementation-defined (IA161) value indicating the current row within the rowset.

The NEST DESCRIPTOR statement attribute determines whether nested descriptor items are permitted in a CLI descriptor. Nested descriptor items are used to describe ROW, ARRAY, and MULTISET data types. The statement attribute is set to FALSE when the SQL-statement is allocated.

4.5.7 CLI descriptor areas

A *CLI descriptor area* provides an interface for a description of <dynamic parameter specification>s, <dynamic parameter specification> values, result columns of <dynamic select statement>s and <dynamic select statement>s, or <target specification>s for the result columns.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

Each descriptor area comprises *header fields* and zero or more *item descriptor areas*. The header fields are specified in Table 5, "Fields in SQL/CLI row and parameter descriptor areas". The header fields include a COUNT field that indicates the number of item descriptor areas and an ALLOC_TYPE field that indicates whether the CLI descriptor area was allocated by the user or automatically allocated by the SQL/CLI implementation.

The header fields include ARRAY_SIZE, ARRAY_STATUS_POINTER, and ROWS_PROCESSED_POINTER. These three fields are used to support the fetching of multiple rows with one invocation of Fetch or FetchScroll.

Each CLI item descriptor area consists of the fields specified following "Status record fields" in Table 5, "Fields in SQL/CLI row and parameter descriptor areas".

The CLI descriptor areas for the four interface types are referred to as an *implementation parameter descriptor* (IPD), an *application parameter descriptor* (APD), an *implementation row descriptor* (IRD), and an *application row descriptor* (ARD), respectively. IPDs and IRDs are collectively known as *implementation descriptor areas*; APDs and ARDs are collectively known as *application descriptor areas*.

When an SQL-statement is allocated, a CLI descriptor area of each type is automatically allocated by the SQL/CLI implementation. The ALLOC_TYPE fields for these CLI descriptor areas are set to indicate AUTOMATIC. A CLI descriptor area allocated by the user has its ALLOC_TYPE field set to indicate USER, and can only be used as an APD or ARD. The handle values of the IPD, IRD, current APD, and current ARD are attributes of the allocated SQL-statement. The SQL/CLI application can determine the current values of these attributes by using the routine GetStmtAttr. The current APD and ARD are initially the automatically-allocated APD and ARD, respectively, but can subsequently be changed by changing the corresponding attribute value using the routine SetStmtAttr.

The routines GetDescField and GetDescRec enable information to be retrieved from a specified CLI descriptor area. The routines SetDescField and SetDescRec enable information to be set in specified CLI descriptor areas except an IRD. The routine BindCol implicitly sets information in the current ARD. The routine BindParameter implicitly sets information in the current APD and the current IPD. The CopyDesc routine enables the contents of any CLI descriptor area to be copied to specified CLI descriptor areas except an IRD.

NOTE 3 — Although there is no need to set a DATA_POINTER field in the IPD to align with the consistency check that applies in the case of an APD or ARD, setting this field causes the item descriptor area to be validated.

4.5.8 Obtaining diagnostics during multi-row fetch

When Fetch or FetchScroll is used to fetch a rowset, exceptions or warnings may be raised during the retrieval of one or more rows in the rowset. The status of each row (that is, information about whether that row in the rowset was successfully retrieved or not) is available in the array addressed by the ARRAY_STATUS_POINTER field of the applicable IRD. The cardinality of this array is the same as the ARRAY_SIZE field of the corresponding ARD. For each row in the rowset, the corresponding element of this array has one of the following values, which are defined in Table 25, "Multi-row fetch status codes".

— A value of 0 (zero) indicates **Row success**, meaning that the row was fetched successfully.

— A value of 6 indicates **Row success with information**, meaning that the row was fetched successfully, but a completion condition was raised: *warning (01000)*.

— A value of 3 indicates **No row**, meaning that there is no row at this position in the rowset. This condition occurs when a partial rowset is retrieved because the result set ended.

— A value of 5 indicates **Row error**, meaning that the row was not fetched successfully and an exception condition was raised.

Each **Row success with information** or **Row Error** generates one or more status records in the diagnostics area. The ROW_NUMBER field for each status record has the value of the row position within the rowset to which this status record corresponds.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

4.6 SQL-invoked routines

This Subclause modifies Subclause 4.35, "SQL-invoked routines", in ISO/IEC 9075-2.

4.6.1 Result sets returned by SQL-invoked procedures

This Subclause modifies Subclause 4.35.6, "Result sets returned by SQL-invoked procedures", in ISO/IEC 9075-2.

Insert into the 7th paragraph, after the last list item:

— The current rowset, consisting of a contiguous subsequence of the sequence of rows. The current rowset may be an empty subsequence located before a specific row, or an empty subsequence located after the last row of the sequence of rows.

NOTE 4 — The position of the result set is a position within the current rowset of the result set, as indicated by the SQL-statement attribute CURRENT OF POSITION. If the value of this attribute does not indicate a row of the result set, then there is no current row.

4.7 Cursors

This Subclause modifies Subclause 4.40, "Cursors", in ISO/IEC 9075-2.

4.7.1 General description of cursors

This Subclause modifies Subclause 4.40.1, "General description of cursors", in ISO/IEC 9075-2.

Insert after the 3rd paragraph: A *CLI cursor* is a cursor created by the SQL/CLI implementation and associated with an allocated SQL-statement. If the allocated SQL-statement is processing a <dynamic select statement> or a <dynamic single row select statement>, then the CLI cursor is a *CLI prepared cursor*. If the CLI cursor is processing a result set returned by an SQL-invoked procedure, then the CLI cursor is a *CLI procedural result cursor*.

Insert into the 5th paragraph, in the 1st list item, after the last list item:

— CLI procedural result.

Insert into the 5th paragraph, in the 3rd list item, after the 2nd list item:

— If the cursor is a CLI cursor, then a <cursor name>.

Insert into the 5th paragraph, in the 4th list item, after the 5th list item:

— If the cursor is a CLI cursor, then the allocated SQL-statement associated with the cursor.

4.8 Client-server operation

This Subclause modifies Subclause 4.47, "Client-server operation", in ISO/IEC 9075-2.

Insert after the 4th paragraph: If the execution of a CLI routine causes the implicit or explicit execution of an <SQL procedure statement> by an SQL-server, diagnostic information is passed in an implementation-dependent (UW010) manner to the SQL-client and then into the appropriate diagnostics area. The effect on diagnostic information of incompatibilities between the character repertoires supported by the SQL-client and the SQL-server is implementation-dependent (UA051).

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

5 Lexical elements

This Clause modifies Clause 5, "Lexical elements", in ISO/IEC 9075-2.

5.1 <token> and <separator>

This Subclause modifies Subclause 5.2, "<token> and <separator>", in ISO/IEC 9075-2.

Function

Specify lexical units (tokens and separators) that participate in SQL language.

Format

```
<reserved word> ::=
    !! All alternatives from ISO/IEC 9075-2

<non-reserved word> ::=
    !! All alternatives from ISO/IEC 9075-2
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

6 Call-Level Interface specifications

This Clause is modified by Clause 18, "Call-Level Interface specifications", in ISO/IEC 9075-9.
This Clause is modified by Clause 19, "Call-Level Interface specifications", in ISO/IEC 9075-14.
This Clause is modified by Clause 17, "Call-Level Interface specifications", in ISO/IEC 9075-15.

6.1 <CLI routine>

This Subclause is modified by Subclause 18.1, "<CLI routine>", in ISO/IEC 9075-9.

Function

Describe SQL/CLI routines in a generic fashion.

Format

```
09 <CLI routine> ::=
  <CLI routine name> <CLI parameter list> [ <CLI returns clause> ]

<CLI routine name> ::=
  <CLI name prefix> <CLI generic name>

<CLI name prefix> ::=
    <CLI by-reference prefix>
  | <CLI by-value prefix>

<CLI by-reference prefix> ::=
  SQLR

<CLI by-value prefix> ::=
  SQL

<CLI generic name> ::=
    AllocConnect
  | AllocEnv
  | AllocHandle
  | AllocStmt
  | BindCol
  | BindParameter
  | Cancel
  | CloseCursor
  | ColAttribute
  | ColumnPrivileges
  | Columns
  | Connect
  | CopyDesc
  | DataSources
  | DescribeCol
  | Disconnect
  | EndTran
  | Error
  | ExecDirect
  | Execute
  | Fetch
  | FetchScroll
```


© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

```
  | ForeignKeys
  | FreeConnect
  | FreeEnv
  | FreeHandle
  | FreeStmt
  | GetConnectAttr
  | GetCursorName
  | GetData
  | GetDescField
  | GetDescRec
  | GetDiagField
  | GetDiagRec
  | GetEnvAttr
  | GetFeatureInfo
  | GetFunctions
  | GetInfo
  | GetLength
  | GetParamData
  | GetPosition
  | GetSessionInfo
  | GetStmtAttr
  | GetSubString
  | GetTypeInfo
  | MoreResults
  | NextResult
  | NumResultCols
  | ParamData
  | Prepare
  | PrimaryKeys
  | PutData
  | RowCount
  | SetConnectAttr
  | SetCursorName
  | SetDescField
  | SetDescRec
  | SetEnvAttr
  | SetStmtAttr
  | SpecialColumns
  | StartTran
  | TablePrivileges
  | Tables
  | <implementation-defined CLI generic name>

<CLI parameter list> ::=
  <left paren> <CLI parameter declaration>
      [ { <comma> <CLI parameter declaration> }... ] <right paren>

<CLI parameter declaration> ::=
  <CLI parameter name> <CLI parameter mode> <CLI parameter data type>

<CLI parameter name> ::=
  !! See the Syntax Rules.

<CLI parameter mode> ::=
    IN
  | OUT
  | DEFIN
  | DEFOUT
  | DEF

<CLI parameter data type> ::=
    INTEGER
  | SMALLINT
```

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

```
  | ANY
  | CHARACTER <left paren> <length> <right paren>

<CLI returns clause> ::=
  RETURNS SMALLINT

<implementation-defined CLI generic name> ::=
  !! See the Syntax Rules.
```

Syntax Rules

1) <CLI routine> is a pre-defined routine written in a programming language that is invoked by a compilation unit of the same programming language. Let *HL* be that programming language.

2) <CLI routine> that contains a <CLI returns clause> is called a *CLI function*. A <CLI routine> that does not contain a <CLI returns clause> is called a *CLI procedure*.

3) There shall be no <separator> between the <CLI name prefix> and the <CLI generic name>.

4) A <CLI parameter name> shall be a parameter name in the Definition section of the Subclause in Clause 7, "SQL/CLI routines", whose name is the <CLI generic name> of the <CLI routine name> of the containing <CLI routine>.

5) For each CLI function *CF*, there is a corresponding CLI procedure *CP*, with the same <CLI routine name>. The <CLI parameter list> for *CP* is the same as the <CLI parameter list> for *CF* but with the following additional <CLI parameter declaration>:

   ```
   ReturnCode OUT SMALLINT
   ```

6) *HL* shall support either the invocation of *CF* or the invocation of *CP*. It is implementation-defined (IA178) which is supported.

7) Case:

   a) If <CLI parameter mode> is IN, then the parameter is an *input parameter*. The value of an input argument is established when a CLI routine is invoked.

   b) If <CLI parameter mode> is OUT, then the parameter is an *output parameter*. The value of an output argument is established when a CLI routine is executed.

   c) If <CLI parameter mode> is DEFIN, then the parameter is a *deferred input parameter*. The value of a deferred input argument for a CLI routine *R* is not established when *R* is invoked, but subsequently during the execution of a related CLI routine.

   d) If <CLI parameter mode> is DEFOUT, then the parameter is a *deferred output parameter*. The value of a deferred output argument for a CLI routine *R* is not established by the execution of *R* but subsequently by the execution of a related CLI routine.

   e) If <CLI parameter mode> is DEF, then the parameter is a *deferred parameter*. The value of a deferred argument for a CLI routine *R* is not established by the execution of *R* but subsequently by the execution of a related CLI routine.

8) The value of an output, deferred output, deferred input, or deferred parameter is an address. It is either a non-pointer host variable passed by reference or a pointer host variable passed by value.

9) A *by-value version* of a CLI routine is a version that expects each of its non-character input parameters to be provided as actual values. A *by-reference version* of a CLI routine is a version that expects each of its input parameters to be provided as an address. By-value and by-reference versions of the CLI routines shall be supported according to Table 3, "Supported calling conventions of SQL/CLI routines by language", for each of the languages identified in the first column of that table.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

Table 3 — Supported calling conventions of SQL/CLI routines by language

Language	By-value	By-reference
Ada (ISO/IEC 8652:2012)	Optional	Required
C (ISO/IEC 9899:2018)	Required	Optional
COBOL (ISO 1989:2014)	Optional	Required
Fortran (ISO/IEC 1539-1:2018 and ISO/IEC 1539-2:2000)	Not supported	Required
M (ISO/IEC 11756:1999)	Optional	Required
Pascal (ISO 7185:1990 and ISO/IEC 10206:1991)	Optional	Required
PL/I (ISO 6160:1979)	Optional	Required

10) If a <CLI routine> is a by-reference routine, then its <CLI routine name> shall contain a <CLI by-reference prefix>. Otherwise, its <CLI routine name> shall contain a <CLI by-value prefix>.

11) The <implementation-defined CLI generic name> for an implementation-defined (IV034) CLI function shall be different from the <CLI generic name> of any other CLI function. The <implementation-defined CLI generic name> for an implementation-defined (IV034) CLI procedure shall be different from the <CLI generic name> of any other CLI procedure.

12) Every <CLI routine name> that cannot be used by an SQL/CLI implementation because of its length or because it is made identical to some other <CLI routine name> by truncation is effectively replaced with an abbreviated name according to the following rules:

    a) Any <CLI by-value prefix> remains unchanged.

    b) Any <CLI by-reference prefix> is replaced by SQR.

    c) The <CLI generic name> is replaced by an abbreviated version according to Table 4, "Abbreviated SQL/CLI generic names".

Table 4 — Abbreviated SQL/CLI generic names

Generic Name	Abbreviation
AllocConnect	AC
AllocEnv	AE
AllocHandle	AH
AllocStmt	AS
BindCol	BC
BindParameter	BP
Cancel	CAN
CloseCursor	CC

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

Generic Name	Abbreviation
ColAttribute	CO
ColumnPrivileges	CP
Columns	COL
Connect	CON
CopyDesc	CD
DataSources	DS
DescribeCol	DC
Disconnect	DIS
EndTran	ET
Error	ER
ExecDirect	ED
Execute	EX
Fetch	FT
FetchScroll	FTS
ForeignKeys	FK
FreeConnect	FC
FreeEnv	FE
FreeHandle	FH
FreeStmt	FS
GetConnectAttr	GCA
GetCursorName	GCN
GetData	GDA
GetDescField	GDF
GetDescRec	GDR
GetDiagField	GXF
GetDiagRec	GXR
GetEnvAttr	GEA
GetFeatureInfo	GFI

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

Generic Name	Abbreviation
GetFunctions	GFU
GetInfo	GI
GetLength	GLN
GetParamData	GPD
GetPosition	GPO
GetSessionInfo	GSI
GetStmtAttr	GSA
GetSubString	GSB
GetTypeInfo	GTI
MoreResults	MR
NextResult	NR
NumResultCols	NRC
ParamData	PRD
Prepare	PR
PrimaryKeys	PK
PutData	PTD
RowCount	RC
SetConnectAttr	SCA
SetCursorName	SCN
SetDescField	SDF
SetDescRec	SDR
SetEnvAttr	SEA
SetStmtAttr	SSA
SpecialColumns	SC
StartTran	STN
TablePrivileges	TP
Tables	TAB

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

Generic Name	Abbreviation
implementation-defined (IV034) CLI routine	implementation-defined (IV034) abbreviation

13) Let *CR* be a <CLI routine> and let *RN* be its <CLI routine name>. Let *RNU* be the value of UPPER(*RN*).

Case:

a) If *HL* supports case sensitive routine names, then the name used for the invocation of *CR* shall be *RN*.

b) If *HL* does not support <simple Latin lower-case letter>s, then the name used for the invocation of *CR* shall be *RNU*.

c) If *HL* does not support case sensitive routine names, then the name used for the invocation of *CR* shall be *RN* or *RNU*.

14) Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 6.19, "SQL/CLI data type correspondences". Refer to the two columns of the operative data type correspondence table as the "*SQL data type column*" and the "*host data type column*".

15) Let *TI*, *TS*, *TC*, and *TV* be the types listed in the host data type column for the rows that contains INTEGER, SMALLINT, CHARACTER(*L*) and CHARACTER VARYING(*L*), respectively, in the SQL data type column.

a) If *TS* is "None", then let *TS* = *TI*.

b) If *TC* is "None", then let *TC* = *TV*.

c) For each parameter *P*,

Case:

i) If the CLI parameter data type is INTEGER, then the type of the corresponding argument shall be *TI*.

ii) If the CLI parameter data type is SMALLINT, then the type of the corresponding argument shall be *TS*.

iii) If the CLI parameter data type is CHARACTER(*L*), then the type of the corresponding argument shall be *TC*.

iv) If the CLI parameter data type is ANY, then

Case:

1) If *HL* is *C*, then the type of the corresponding argument shall be "`void *`".

2) Otherwise, the type of the corresponding argument shall be a type (other than "None") listed in the host data type column.

d) If the CLI routine is a CLI function, then the type of the returned value is *TS*.

Access Rules

None.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

General Rules

1) The rules for invocation of a <CLI routine> are specified in Subclause 6.2, "<CLI routine> invocation".

Conformance Rules

1) Without Feature C001, "CLI routine invocation in Ada", a conforming SQL/CLI application shall not contain an invocation of a <CLI routine> written in Ada.

2) Without Feature C002, "CLI routine invocation in C", a conforming SQL/CLI application shall not contain an invocation of a <CLI routine> written in C.

3) Without Feature C003, "CLI routine invocation in COBOL ", a conforming SQL/CLI application shall not contain an invocation of a <CLI routine> written in COBOL.

4) Without Feature C004, "CLI routine invocation in Fortran", a conforming SQL/CLI application shall not contain an invocation of a <CLI routine> written in Fortran.

5) Without Feature C005, "CLI routine invocation in MUMPS ", a conforming SQL/CLI application shall not contain an invocation of a <CLI routine> written in M.

6) Without Feature C006, "CLI routine invocation in Pascal", a conforming SQL/CLI application shall not contain an invocation of a <CLI routine> written in Pascal.

7) Without Feature C007, "CLI routine invocation in PL/I", a conforming SQL/CLI application shall not contain an invocation of a <CLI routine> written in PL/I.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

6.2 <CLI routine> invocation

Function

Specify the rules for invocation of a <CLI routine>.

Format

No additional Format items.

Syntax Rules

1) Let *HL* be the programming language of the invoking host program.

2) A CLI function or CLI procedure is invoked by the *HL* mechanism for invoking functions or procedures, respectively.

3) Let *RNM* be the <CLI routine name> of the <CLI routine> invoked by the host program and let *RN* be the SQL/CLI routine identified by *RNM*. The number of arguments provided in the invocation shall be the same as the number of <CLI parameter declaration>s for *RN*.

4) Let *DA* be the data type of the *i*-th argument in the invocation and let *DP* be the <CLI parameter data type> of the *i*-th <CLI parameter declaration> of *RN*. *DA* shall be the *HL* equivalent of *DP* as specified by the rules of Subclause 6.1, "<CLI routine>".

Access Rules

None.

General Rules

1) If the value of any input argument provided by the host program is not a value of the data type of the parameter, or if the value of any output argument resulting from the execution of the <CLI routine> is not a value supported by the SQL/CLI application for that parameter, then the effect is implementation-defined (IA179).

2) Let *GRN* be the <CLI generic name> of *RN*.

3) When the <CLI routine> is called by the SQL/CLI application:

   a) The values of all input arguments to *RN* are established.

   b) Case:

      i) If *RN* is a CLI routine with a statement handle as an input parameter, *RN* has no accompanying handle type parameter, and *GRN* is not Error, then:

         1) If the statement handle does not identify an allocated SQL-statement, then an exception condition is raised: *CLI-specific condition — invalid handle (HYHHH)*. Otherwise, let *S* be the allocated SQL-statement identified by the statement handle.

         2) If *GRN* is not Cancel, then the diagnostics area associated with *S* is emptied.

         3) Let *C* be the allocated SQL-connection with which *S* is associated.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

4) If there is no established SQL-connection associated with *C*, then an exception condition is raised: *connection exception — connection does not exist (08003)*. Otherwise, let *EC* be the established SQL-connection associated with *C*.

5) If *EC* is not the current SQL-connection, then the General Rules of Subclause 6.3, "Implicit set connection", are applied with *EC* as *dormant SQL-connection*.

6) If *GRN* is neither Cancel nor ParamData nor PutData and there is a deferred parameter number associated with *S*, then an exception condition is raised: *CLI-specific condition — function sequence error (HY010)*.

7) *RN* is invoked.

ii) If *RN* is a CLI routine with a descriptor handle as an input parameter and *RN* has no accompanying handle type parameter and *GRN* is not CopyDesc, then:

1) If the descriptor handle does not identify an allocated CLI descriptor area, then an exception condition is raised: *CLI-specific condition — invalid handle (HYHHH)*. Otherwise, let *D* be the allocated CLI descriptor area identified by the descriptor handle.

2) The diagnostics area associated with *D* is emptied.

3) Let *C* be the allocated SQL-connection with which *D* is associated.

4) If there is no established SQL-connection associated with *C*, then an exception condition is raised: *connection exception — connection does not exist (08003)*. Otherwise, let *EC* be the established SQL-connection associated with *C*.

5) If *EC* is not the current SQL-connection, then the General Rules of Subclause 6.3, "Implicit set connection", are applied with *EC* as *dormant SQL-connection*.

6) *RN* is invoked.

iii) Otherwise, *RN* is invoked.

4) Case:

a) If *RN* is a CLI function, then:

i) The values of all output arguments are established.

ii) Let *RC* be the return value.

b) If *RN* is a CLI procedure, then:

i) The values of all output arguments are established except for the argument associated with the ReturnCode parameter.

ii) Let *RC* be the argument associated with the ReturnCode parameter.

5) Case:

a) If *RN* did not complete execution because it requires more input data, then:

i) *RC* is set to indicate **Data needed**.

ii) An exception condition is raised: *CLI-specific condition — dynamic parameter value needed (HYHHG)*.

b) If *RN* executed successfully, then:

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

i) Either a completion condition is raised: *successful completion (00000)*, or a completion condition is raised: *warning (01000)*, or a completion condition is raised: *no data (02000)*.

ii) Case:

1) If a completion condition is raised: *successful completion (00000)*, then *RC* is set to indicate **Success**.

2) If a completion condition is raised: *warning (01000)*, then *RC* is set to indicate **Success with information**.

3) If a completion condition is raised: *no data (02000)*, then *RC* is set to indicate **No data found**.

c) If *RN* did not execute successfully, then:

i) All changes made to SQL-data or schemas by the execution of *RN* are canceled.

ii) One or more exception conditions are raised as determined by the General Rules of this and other Subclauses of this document or by implementation-defined (IA180) rules.

iii) Case:

1) If an exception condition is raised: *CLI-specific condition — invalid handle (HYHHH)*, then *RC* is set to indicate **Invalid handle**.

2) Otherwise, *RC* is set to indicate **Error**.

6) Case:

a) If *GRN* is neither Error nor GetDiagField nor GetDiagRec, and *RC* indicates neither **Invalid handle** nor **Data needed**, then diagnostic information resulting from the execution of *RN* is placed into the appropriate diagnostics area as specified in Subclause 4.3, "Return codes", and Subclause 4.4, "Diagnostics areas in SQL/CLI".

b) Otherwise, no diagnostics area is updated.

Conformance Rules

None.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

6.3 Implicit set connection

Function

Specify the rules for an implicit SET CONNECTION statement.

Subclause Signature

```
"Implicit set connection" [General Rules] (
  Parameter: "dormant SQL-connection"
)
```

dormant SQL-connection — an existing SQL-connection.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *DC* be the *dormant SQL-connection* in an application of the General Rules of this Subclause.

2) If an SQL-transaction is active for the current SQL-connection and support for Feature C008, "Multiple server transactions in CLI" is not provided, then an exception condition is raised: *feature not supported — multiple server transactions (0A001)*.

3) If *DC* cannot be selected, then an exception condition is raised: *connection exception — connection failure (08006)*.

4) The current SQL-connection *CC* and current SQL-session become a dormant SQL-connection and a dormant SQL-session, respectively. The SQL-session context for *CC* is preserved and is not affected in any way by operations performed over the selected SQL-connection.

   NOTE 5 — The SQL-session context is defined in Subclause 4.45, "SQL-sessions", in ISO/IEC 9075-2.

5) *DC* becomes the current SQL-connection and the SQL-session associated with *DC* becomes the *current SQL-session*. The SQL-session context is restored to the same state as at the time *DC* became dormant.

   NOTE 6 — The SQL-session context information is defined in Subclause 4.45, "SQL-sessions", in ISO/IEC 9075-2.

6) The SQL-server for the subsequent execution of SQL-statements via CLI routine invocations is set to that of the current SQL-connection.

7) Evaluation of the General Rules is terminated and control is returned to the invoking Subclause.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

6.4 Preparing a statement

Function

Prepare a statement.

Subclause Signature

```
"Preparing a statement" [General Rules] (
  Parameter: "ALLOCATED STATEMENT",
  Parameter: "TEXT LENGTH",
  Parameter: "STATEMENT TEXT",
  Parameter: "INVOKER"
)
```

ALLOCATED STATEMENT — an allocated SQL-statement descriptor.

TEXT LENGTH — the length of STATEMENT TEXT, or the NULL TERMINATED indication (-3).

STATEMENT TEXT — the text of an SQL-statement to be prepared for execution.

INVOKER — a character string identifying the Subclause that invoked this Subclause (e.g., 'ExecDirect', 'Prepare', or 'ParamData').

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *S* be the *ALLOCATED STATEMENT*, let *TL* be the *TEXT LENGTH*, let *ST* be the *STATEMENT TEXT*, and let *INV* be the *INVOKER* in an application of the General Rules of this Subclause.

2) If an open CLI cursor is associated with *S*, then an exception condition is raised: *invalid cursor state (24000)*.

3) Case:

   a) If *TL* is not negative, then let *L* be *TL*.

   b) If *TL* indicates NULL TERMINATED, then let *L* be the number of octets of *ST* that precede the implementation-defined (IV030) null character that terminates a C character string.

   c) Otherwise, an exception condition is raised: *CLI-specific condition — invalid string length or buffer length (HY090)*.

4) Case:

   a) If *L* is zero, then an exception condition is raised: *CLI-specific condition — invalid string length or buffer length (HY090)*.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

b) Otherwise, let *P* be the first *L* octets of *ST*.

5) If *P* is a <preparable dynamic delete statement: positioned> or a <preparable dynamic update statement: positioned>, then let *CN* be the cursor name referenced by *P*. Let *C* be the allocated SQL-connection with which *S* is associated. If *CN* is not the name of a CLI cursor associated with another allocated SQL-statement associated with *C*, then an exception condition is raised: *invalid cursor name (34000)*.

6) If at least one of the following is true, then an exception condition is raised: *syntax error or access rule violation (42000)*.

a) *P* does not conform to the Format, Syntax Rules or Access Rules for a <preparable statement> or *P* is a <start transaction statement>, a <commit statement>, a <rollback statement>, or a <release savepoint statement>.

NOTE 7 — See Table 39, "SQL-statement codes", in ISO/IEC 9075-2 for the list of <preparable statement>s. Some other parts of the ISO/IEC 9075 series have corresponding tables that define additional codes representing statements defined by those parts of ISO/IEC 9075.

b) *P* contains a <simple comment>.

c) *P* contains a <dynamic parameter specification> whose data type is undefined as determined by the rules specified in Subclause 20.7, "<prepare statement>", in ISO/IEC 9075-2.

7) The data type of every <dynamic parameter specification> contained in *P* is determined by the rules specified in Subclause 20.7, "<prepare statement>", in ISO/IEC 9075-2.

8) Let *DTGN* be the default transform group name and *TFL* be the list of user-defined type name—transform group name pairs used to identify the group of transform functions for every user-defined type that is referenced in *P*. *DTGN* and *TFL* are not affected by the execution of a <set transform group statement> after *P* is prepared.

9) The following objects associated with *S* are destroyed:

a) Every prepared statement.

b) The cursor declaration descriptor every cursor instance descriptor of every CLI cursor.

c) Every select source.

d) If *INV* is "Prepare", then every executed statement.

If a cursor associated with *S* is destroyed, then so are all prepared statements that reference that cursor.

10) *P* is prepared.

11) If *INV* is "Prepare", then the prepared statement is associated with *S*.

12) If *P* is a <dynamic select statement> or a <dynamic single row select statement>, then *P* becomes the select source associated with *S*.

13) The General Rules of Subclause 6.9, "Implicit DESCRIBE USING clause", are applied with *SS* as *SOURCE* and *S* as *ALLOCATED STATEMENT*.

14) The validity of a prepared statement in an SQL-transaction different from the one in which the statement was prepared is implementation-defined (IA181).

15) Evaluation of the General Rules is terminated and control is returned to the invoking Subclause.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

Conformance Rules

None.

© ISO/IEC 2023 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

6.5 Executing a statement

Function

Execute a statement.

Subclause Signature

```
"Executing a statement" [General Rules] (
  Parameter: "ALLOCATED STATEMENT",
  Parameter: "PREPARED STATEMENT",
  Parameter: "INVOKER"
)
```

ALLOCATED STATEMENT — an allocated SQL-statement descriptor.

PREPARED STATEMENT — an SQL-statement that has been prepared for dynamic execution.

INVOKER — a character string identifying the Subclause that invoked this Subclause (e.g., 'ExecDirect', 'Prepare', or 'ParamData').

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *S* be the *ALLOCATED STATEMENT*, let *P* be the *PREPARED STATEMENT*, and let *INV* be the *INVOKER* in an application of the General Rules of this Subclause.

2) *P* is executed as follows.

   Case:

   a) If *P* is a <dynamic select statement> or a <dynamic single row select statement>, then the General Rules of Subclause 6.6, "Implicit CLI prepared cursor", are applied with *P* as *SELECT SOURCE*, *S* as *ALLOCATED STATEMENT*, and *INV* as *INVOKER*

   b) Otherwise:

      i) If *INV* is not "ParamData", then General Rules of Subclause 6.10, "Implicit EXECUTE USING and OPEN USING clauses", are applied with EXECUTE as *TYPE*, *P* as *SOURCE*, and *S* as *ALLOCATED STATEMENT*.

         NOTE 8 — When this Subclause is invoked from ParamData, Subclause 6.10, "Implicit EXECUTE USING and OPEN USING clauses", must have been previously invoked.

      ii) Case:

         1) If *P* is a <preparable dynamic delete statement: positioned>, then:

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

A) Let *CR* be the cursor referenced by *P* and let *SCR* be the allocated SQL-statement associated with *CR*.

B) Let *TT* be the implicit or explicit <target table> of *P*, as defined by the Syntax Rules for <preparable dynamic delete statement: positioned>.

C) The General Rules of Subclause 15.6, "Effect of a positioned delete", in ISO/IEC 9075-2, are applied with *CR* as *CURSOR*, *P* as *STATEMENT*, and *TT* as *TARGET*. For the purposes of the application of these Rules, the row in *CR* identified by *SCR*'s CURRENT OF POSITION statement attribute is the *current row* of *CR*.

D) If the execution of *P* deleted the current row of *CR*, then the effect on the fetched row, if any, associated with *SCR* is implementation-defined (IA182).

2) If *P* is a <preparable dynamic update statement: positioned>, then:

A) Let *CR* be the cursor referenced by *P* and let *SCR* be the allocated SQL-statement associated with *CR*.

B) Let *SCL* be the <set clause list> contained in *P*.

C) Let *TT* be the implicit or explicit <target table> of *P*, as defined by the Syntax Rules for <preparable dynamic update statement: positioned>.

D) The General Rules of Subclause 15.7, "Effect of a positioned update", in ISO/IEC 9075-2, are applied with *CR* as *CURSOR*, *SCL* as *SET CLAUSE LIST*, *P* as *STATEMENT*, and *TT* as *TARGET*. For the purposes of the application of these Rules, the row in *CR* identified by *SCR*'s CURRENT OF POSITION statement attribute is the *current row* of *CR*.

E) If the execution of *P* updated the current row of *CR*, then the effect on the fetched row, if any, associated with *SCR* is implementation-defined (IA183).

3) Otherwise, the results of the execution are the same as if the statement were contained in an <externally-invoked procedure> and executed; these are described in Subclause 13.3, "<externally-invoked procedure>", in ISO/IEC 9075-2.

iii) If *P* is a <call statement>, then:

1) The General Rules of Subclause 6.11, "Implicit CALL USING clause", are applied with *P* as *SOURCE* and *S* as *ALLOCATED STATEMENT*.

2) If the result set sequence *RSS* of the SQL-invoked procedure that was invoked by the <call statement> is non-empty, then the General Rules of Subclause 6.7, "Implicit CLI procedural result cursor", are applied with *S* as *ALLOCATED STATEMENT* and *RSS* as *RESULT SET SEQUENCE*.

3) Let *R* be the value of the ROW_COUNT field from the diagnostics area associated with *S*.

4) *R* becomes the row count associated with *S*.

5) If *P* executed successfully, then every executed statement associated with *S* is destroyed and *P* becomes the executed statement associated with *S*.

6) Evaluation of the General Rules is terminated and control is returned to the invoking Subclause.

Conformance Rules

None.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

6.6 Implicit CLI prepared cursor

Function

Specify the cursor declaration descriptor and cursor instance descriptor of a CLI prepared cursor.

Subclause Signature

```
"Implicit CLI prepared cursor" [General Rules] (
  Parameter: "SELECT SOURCE",
  Parameter: "ALLOCATED STATEMENT",
  Parameter: "INVOKER"
)
```

SELECT SOURCE — the text of the declaration of a cursor that is to be dynamically opened.

ALLOCATED STATEMENT — an allocated SQL-statement descriptor.

INVOKER — a character string identifying the Subclause that invoked this Subclause (e.g., 'ExecDirect', 'Prepare', or 'ParamData').

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *SS* be the *SELECT SOURCE*, let *AS* be the *ALLOCATED STATEMENT*, and let *INV* be the *INVOKER* in an application of the General Rules of this Subclause.

2) If there is no CLI cursor associated with *AS*, then the General Rules of Subclause 6.8, "Initial CLI cursor", are applied with *AS* as *ALLOCATED STATEMENT*.

3) Let *CID* be the cursor instance descriptor of the cursor associated with *AS*, and let *CDD* be the cursor declaration descriptor of *CID*.

4) The kind of cursor in *CID* is set to CLI prepared cursor.

5) The declared properties of the cursor declaration descriptor of *CID* are set as follows:

   a) The cursor's declared sensitivity is

      Case:

      i) If the value of the CURSOR SENSITIVITY attribute of *AS* is INSENSITIVE, then INSENSITIVE.

      ii) If the value of the CURSOR SENSITIVITY attribute of *AS* is SENSITIVE, then SENSITIVE.

      iii) Otherwise, ASENSITIVE.

   b) The cursor's declared scrollability is

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023

Case:

i) If the value of the CURSOR SCROLLABLE attribute of *AS* is SCROLLABLE, then SCROLL.

ii) Otherwise, NO SCROLL.

c) The cursor's declared holdability is

Case:

i) If the value of the CURSOR HOLDABLE attribute of *AS* is HOLDABLE, then WITH HOLD.

ii) Otherwise, WITHOUT HOLD.

d) The cursor's declared returnability is implementation-defined (IV031).

6) If *INV* is not "ParamData", then the General Rules of Subclause 6.10, "Implicit EXECUTE USING and OPEN USING clauses", are applied with OPEN as *TYPE*, *SS* as *SOURCE*, and *AS* as *ALLOCATED STATEMENT*.

NOTE 9 — When this Subclause is invoked from ParamData, Subclause 6.10, "Implicit EXECUTE USING and OPEN USING clauses", must have been previously invoked.

7) The General Rules of Subclause 8.1, "Effect of opening a cursor", are applied with *CID* as *CURSOR*.

NOTE 10 — In applying this Subclause, the values of <dynamic parameter specification>s are described by the implementation parameter descriptor and application parameter descriptor of *AS*, as explained in Subclause 6.10, "Implicit EXECUTE USING and OPEN USING clauses".

8) Evaluation of the General Rules is terminated and control is returned to the invoking Subclause.

Conformance Rules

None.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 9075-3:2023