

MBS Real Studio SQL Plugin Documentation

Christian Schmitz

May 15, 2012

0.1 Introduction

This is the PDF version of the documentation for the Real Studio Plug-in from Monkeybread Software Germany. Plugin part: MBS Real Studio SQL Plugin

0.2 Content

- 1 List of all topics 3
- 2 All items in this plugin 13
- 3 List of all classes 121

Chapter 1

List of Topics

• 2 SQL	13
– 2.1 class SQLPositionMBS	13
* 2.1.1 Constructor(withID as integer)	13
* 2.1.1 Constructor(withName as string)	13
– 2.4 class SQLParamMBS	25
* 2.4.1 Name as string	25
* 2.4.1 Option(name as string) as string	25
* 2.4.1 ParamDirType as integer	26
* 2.4.1 ParamNativeType as integer	26
* 2.4.1 ParamPrecision as integer	26
* 2.4.1 ParamScale as integer	27
* 2.4.1 ParamSize as integer	27
* 2.4.1 ParamType as integer	27
* 2.4.1 ReadLongOrLob(consumer as SQLDataConsumerMBS, BlockSize as integer)	27
* 2.4.2 kParamDirTypeInput=0	28
* 2.4.2 kParamDirTypeInputOutput=1	28
* 2.4.2 kParamDirTypeOutput=2	28
* 2.4.2 kParamDirTypeReturn=3	28
– 2.2 class SQLPreparedStatementMBS	14
* 2.2.1 Bind(values() As Variant)	14
* 2.2.1 Bind(zeroBasedIndex As Integer, value As Variant)	14
* 2.2.1 Bind(zeroBasedIndex As Integer, value As Variant, type as integer)	15
* 2.2.1 BindType(types() As Integer)	15
* 2.2.1 BindType(zeroBasedIndex As Integer, type As Integer)	15
* 2.2.1 Constructor	16

* 2.2.1	SQLExecute(ParamArray bindItems As Variant)	16
* 2.2.1	SQLSelect(ParamArray bindItems As Variant) As RecordSet	16
* 2.2.2	kTypeBlob = 14	16
* 2.2.2	kTypeBool = 1	17
* 2.2.2	kTypeBytes = 11	17
* 2.2.2	kTypeClob = 15	17
* 2.2.2	kTypeDateTime = 8	17
* 2.2.2	kTypeDouble = 6	17
* 2.2.2	kTypeLong = 4	18
* 2.2.2	kTypeLongBinary = 12	18
* 2.2.2	kTypeLongChar = 13	18
* 2.2.2	kTypeShort = 2	18
* 2.2.2	kTypeString = 10	18
* 2.2.2	kTypeULong = 5	19
* 2.2.2	kTypeUnknown = 0	19
* 2.2.2	kTypeUShort = 3	19
– 2.3	class SQLStringMBS	19
* 2.3.1	Compare(text as SQLStringMBS) as integer	20
* 2.3.1	Compare(text as string) as integer	20
* 2.3.1	CompareNoCase(text as SQLStringMBS) as integer	21
* 2.3.1	CompareNoCase(text as string) as integer	21
* 2.3.1	Constructor	21
* 2.3.1	Constructor(Data as string, isText as boolean=true)	22
* 2.3.1	Constructor(other as SQLStringMBS)	22
* 2.3.1	CopyBinaryData as string	22
* 2.3.1	CopyText as string	22
* 2.3.1	Empty	22
* 2.3.1	GetBinaryLength as UInt32	23
* 2.3.1	GetLength as UInt32	23
* 2.3.1	IsEmpty as boolean	23
* 2.3.1	Left(count as integer) as SQLStringMBS	23
* 2.3.1	MakeLower	23
* 2.3.1	MakeUpper	24
* 2.3.1	Mid(first as integer) as SQLStringMBS	24
* 2.3.1	Mid(first as integer, Count as integer) as SQLStringMBS	24
* 2.3.1	Right(count as integer) as SQLStringMBS	24
* 2.3.1	TrimLeft	25
* 2.3.1	TrimRight	25
– 2.5	class SQLIntervalMBS	29
* 2.5.1	Constructor	29
* 2.5.1	Constructor(days as integer, hours as integer, minutes as integer, seconds as integer)	29

* 2.5.1 Constructor(value as double)	29
* 2.5.1 Dec(interval as SQLIntervalMBS)	30
* 2.5.1 DoubleValue as double	30
* 2.5.1 GetDays as integer	30
* 2.5.1 GetHours as integer	30
* 2.5.1 GetMinutes as integer	31
* 2.5.1 GetSeconds as integer	31
* 2.5.1 GetTotalDays as double	31
* 2.5.1 GetTotalHours as double	31
* 2.5.1 GetTotalMinutes as double	31
* 2.5.1 GetTotalSeconds as double	32
* 2.5.1 Inc(interval as SQLIntervalMBS)	32
* 2.5.1 SetInterval(days as integer, hours as integer, minutes as integer, seconds as integer)	32
* 2.5.1 StringValue as string	32
– 2.25 class SQLGlobalsMBS	109
* 2.25.1 GetVersionBuild as integer	110
* 2.25.1 GetVersionMajor as integer	110
* 2.25.1 GetVersionMinor as integer	110
* 2.25.1 SetLicenseCode(n as string, enddate as integer, v1 as integer, v2 as integer)	110
* 2.25.1 Setlocale(category as integer, locale as string)	110
* 2.25.2 LocaleAll=0	111
* 2.25.2 LocaleCollate=1	111
* 2.25.2 LocaleCTYPE=2	112
* 2.25.2 LocaleMessages=6	112
* 2.25.2 LocaleMonetary=3	112
* 2.25.2 LocaleNumeric=4	112
* 2.25.2 LocaleTime=5	112
– 2.6 class SQLite3MBS	33
* 2.6.1 Version as string	33
* 2.6.1 VersionNumber as integer	33
– 2.7 class SQLNumericMBS	34
* 2.7.1 Constructor	34
* 2.7.1 Constructor(value as double)	34
* 2.7.1 Constructor(value as string)	34
* 2.7.1 DoubleValue as double	35
* 2.7.1 Int64Value as Int64	35
* 2.7.1 precision as integer	35
* 2.7.1 scale as integer	35
* 2.7.1 sign as integer	35
* 2.7.1 StringValue as string	36
* 2.7.1 UInt64Value as UInt64	36

– 2.8 class SQLLongBinaryMBS	36
* 2.8.1 Constructor(data as SQLStringMBS)	36
* 2.8.1 Constructor(dataProvider as SQLDataProviderMBS, BlockSize as UInt32)	37
– 2.9 class SQLLongCharMBS	37
* 2.9.1 Constructor(data as SQLStringMBS)	37
* 2.9.1 Constructor(dataProvider as SQLDataProviderMBS, BlockSize as UInt32)	38
– 2.13 class SQLValueReadMBS	39
* 2.13.1 asBlob as SQLStringMBS	39
* 2.13.1 asBool as boolean	40
* 2.13.1 asBytes as SQLStringMBS	40
* 2.13.1 asClob as SQLStringMBS	41
* 2.13.1 asDate as Date	41
* 2.13.1 asDateTime as SQLDateTimeMBS	42
* 2.13.1 asDouble as double	43
* 2.13.1 asInterval as SQLIntervalMBS	43
* 2.13.1 asLong as integer	44
* 2.13.1 asLongBinary as SQLStringMBS	44
* 2.13.1 asLongChar as SQLStringMBS	45
* 2.13.1 asNumeric as SQLNumericMBS	45
* 2.13.1 asShort as Int16	46
* 2.13.1 asString as SQLStringMBS	47
* 2.13.1 asStringValue as String	48
* 2.13.1 asULong as UInt32	48
* 2.13.1 asUShort as UInt16	49
* 2.13.1 Constructor(DataType as integer)	49
* 2.13.1 Constructor(value as SQLValueReadMBS)	50
* 2.13.1 DataType as integer	50
* 2.13.1 isNull as boolean	50
* 2.13.1 LongOrLobReaderMode as integer	50
* 2.13.2 kDataTypeBlob=14	51
* 2.13.2 kDataTypeBool=1	51
* 2.13.2 kDataTypeBytes=11	51
* 2.13.2 kDataTypeClob=15	51
* 2.13.2 kDataTypeCursor=16	52
* 2.13.2 kDataTypeDateTime=8	52
* 2.13.2 kDataTypeDouble=6	52
* 2.13.2 kDataTypeInterval=9	52
* 2.13.2 kDataTypeLong=4	52
* 2.13.2 kDataTypeLongBinary=12	53
* 2.13.2 kDataTypeLongChar=13	53
* 2.13.2 kDataTypeNumeric=7	53

* 2.13.2 kDataTypeShort=2	53
* 2.13.2 kDataTypeSpecificToDBMS=17	53
* 2.13.2 kDataTypeString=10	54
* 2.13.2 kDataTypeULong=5	54
* 2.13.2 kDataTypeUnknown=0	54
* 2.13.2 kDataTypeUShort=3	54
* 2.13.2 kLongOrLobReaderModeDefault=0	54
* 2.13.2 kLongOrLobReaderModeManual=1	55
– 2.15 class SQLValueMBS	55
* 2.15.1 Constructor(DataType as integer)	55
* 2.15.1 isDefault as boolean	56
* 2.15.1 setAsBlob(data as SQLDataProviderMBS, BlockSize as UInt32)	56
* 2.15.1 setAsBlob(data as SQLStringMBS)	56
* 2.15.1 setAsBlob(data as string)	56
* 2.15.1 setAsBool(value as boolean)	57
* 2.15.1 setAsBytes(value as SQLStringMBS)	57
* 2.15.1 setAsClob(data as SQLDataProviderMBS, BlockSize as UInt32)	57
* 2.15.1 setAsClob(text as SQLStringMBS)	57
* 2.15.1 setAsClob(text as string)	58
* 2.15.1 setAsDate(value as date)	58
* 2.15.1 setAsDateTime(value as SQLDateTimeMBS)	58
* 2.15.1 setAsDefault	58
* 2.15.1 setAsDouble(value as double)	59
* 2.15.1 setAsInterval(value as SQLIntervalMBS)	59
* 2.15.1 setAsLong(value as Int32)	59
* 2.15.1 setAsLongBinary(data as SQLDataProviderMBS, BlockSize as UInt32)	60
* 2.15.1 setAsLongBinary(data as SQLStringMBS)	60
* 2.15.1 setAsLongBinary(data as string)	60
* 2.15.1 setAsLongChar(data as SQLDataProviderMBS, BlockSize as UInt32)	61
* 2.15.1 setAsLongChar(text as SQLStringMBS)	61
* 2.15.1 setAsLongChar(text as string)	61
* 2.15.1 setAsNull	62
* 2.15.1 setAsNumeric(value as SQLNumericMBS)	62
* 2.15.1 setAsShort(value as Int16)	62
* 2.15.1 setAsString(text as string)	62
* 2.15.1 setAsString(value as SQLStringMBS)	62
* 2.15.1 setAsULong(value as UInt32)	63
* 2.15.1 setAsUnknown	63
* 2.15.1 setAsUShort(value as UInt16)	63
* 2.15.1 setAsValueRead(value as SQLValueReadMBS)	63
– 2.22 class SQLCommandMBS	86

* 2.22.1 Cancel	87
* 2.22.1 Close	88
* 2.22.1 CommandText as string	88
* 2.22.1 CommandType as integer	89
* 2.22.1 Connection as SQLConnectionMBS	89
* 2.22.1 Constructor	90
* 2.22.1 Constructor(connection as SQLConnectionMBS, SQLCommand as String, CommandType as integer=0)	90
* 2.22.1 CreateParam(name as string, ParamType as integer, DirType as integer=0) as SQLParamMBS	90
* 2.22.1 CreateParam(name as string, ParamType as integer, NativeType as integer, ParamSize as integer, ParamPrecision as integer, ParamScale as integer, DirType as integer=0) as SQLParamMBS	91
* 2.22.1 DestroyParams	92
* 2.22.1 Execute	92
* 2.22.1 ExecuteCommand(SQLCommand as string, CommandType as integer=0)	93
* 2.22.1 ExecuteCommandMT(SQLCommand as string, CommandType as integer=0)	93
* 2.22.1 ExecuteMT	93
* 2.22.1 FetchFirst as boolean	94
* 2.22.1 FetchLast as boolean	94
* 2.22.1 FetchNext as boolean	95
* 2.22.1 FetchPrior as boolean	95
* 2.22.1 Field(index as integer) as SQLFieldMBS	95
* 2.22.1 Field(name as string) as SQLFieldMBS	96
* 2.22.1 FieldCount as integer	97
* 2.22.1 isExecuted as boolean	97
* 2.22.1 isOpened as boolean	97
* 2.22.1 isResultSet as boolean	97
* 2.22.1 Open	97
* 2.22.1 Option(name as string) as string	98
* 2.22.1 Param(ID as integer) as SQLParamMBS	98
* 2.22.1 Param(name as string) as SQLParamMBS	99
* 2.22.1 ParamByIndex(index as integer) as SQLParamMBS	100
* 2.22.1 ParamCount as integer	100
* 2.22.1 Prepare	101
* 2.22.1 RowsAffected as integer	101
* 2.22.1 setCommandText(SQLCommand as string, CommandType as integer=0)	101
* 2.22.2 Working	102
* 2.22.3 kCommandTypeSQLStatement=1	102
* 2.22.3 kCommandTypeSQLStatementRaw=2	102
* 2.22.3 kCommandTypeStoredProcedure=3	102
* 2.22.3 kCommandTypeUnknown=0	103

* 2.22.3 kOptionPreFetchRows="PreFetchRows"	103
* 2.22.3 kParamDirTypeInput=0	103
* 2.22.3 kParamDirTypeInputOutput=1	103
* 2.22.3 kParamDirTypeOutput=2	103
* 2.22.3 kParamDirTypeReturn=3	104
– 2.16 class SQLConnectionMBS	63
* 2.16.1 AutoCommit as integer	65
* 2.16.1 Client as integer	65
* 2.16.1 ClientVersion as integer	65
* 2.16.1 Commit	65
* 2.16.1 Connect(DBString as string, UserID as string, Password as string, client as integer=0)	66
* 2.16.1 Disconnect	67
* 2.16.1 isAlive as boolean	67
* 2.16.1 isConnected as boolean	67
* 2.16.1 IsolationLevel as integer	68
* 2.16.1 NativeAPI as SQLAPIMBS	68
* 2.16.1 Option(name as string) as string	68
* 2.16.1 Rollback	68
* 2.16.1 ServerVersion as integer	69
* 2.16.1 ServerVersionString as string	69
* 2.16.1 SetFileOption(name as string, file as folderitem)	69
* 2.16.1 SQLExecute(command as string, CommandType as integer=0)	70
* 2.16.1 SQLExecuteMT(command as string, CommandType as integer=0)	70
* 2.16.1 SQLSelect(command as string, CommandType as integer=0) as string	70
* 2.16.1 SQLSelectMT(command as string, CommandType as integer=0) as string	71
* 2.16.2 Working	71
* 2.16.3 kANSILevel0 = 0	72
* 2.16.3 kANSILevel1 = 1	72
* 2.16.3 kANSILevel2 = 2	72
* 2.16.3 kANSILevel3 = 3	72
* 2.16.3 kAutoCommitOff = 0	72
* 2.16.3 kAutoCommitOn = 1	73
* 2.16.3 kAutoCommitUnknown = -1	73
* 2.16.3 kClientNotSpecified = 0	73
* 2.16.3 kDB2Client = 6	73
* 2.16.3 kFirebirdClient = 4	73
* 2.16.3 kInformixClient = 7	74
* 2.16.3 kInterBaseClient = 4	74
* 2.16.3 kLevelUnknown = -1	74
* 2.16.3 kMySQLClient = 9	74

* 2.16.3 kODBCClient = 1	74
* 2.16.3 kOptionAPPNAME = "APPNAME"	75
* 2.16.3 kOptionLibraryDB2 = "DB2CLI.LIBS"	75
* 2.16.3 kOptionLibraryFirebird = "IBASE.LIBS"	75
* 2.16.3 kOptionLibraryInformix = "INFCLI.LIBS"	75
* 2.16.3 kOptionLibraryInterbase = "IBASE.LIBS"	75
* 2.16.3 kOptionLibraryMySQL = "MYSQL.LIBS"	76
* 2.16.3 kOptionLibraryODBC = "ODBC.LIBS"	76
* 2.16.3 kOptionLibraryOracle = "OCI8.LIBS"	76
* 2.16.3 kOptionLibraryPostgreSQL = "LIBPQ.LIBS"	76
* 2.16.3 kOptionLibrarySeparator = ":"	76
* 2.16.3 kOptionLibrarySQLBase = "SQLBASE.LIBS"	77
* 2.16.3 kOptionLibrarySQLite = "SQLITE.LIBS"	77
* 2.16.3 kOptionLibrarySybaseComn = "SYBCOMN.LIBS"	77
* 2.16.3 kOptionLibrarySybaseCS = "SYBCS.LIBS"	77
* 2.16.3 kOptionLibrarySybaseCT = "SYBCT.LIBS"	77
* 2.16.3 kOptionLibrarySybaseIntl = "SYBINTL.LIBS"	78
* 2.16.3 kOptionLibrarySybaseTCL = "SYBTCL.LIBS"	78
* 2.16.3 kOptionWSID = "WSID"	78
* 2.16.3 kOracleClient = 2	78
* 2.16.3 kPostgreSQLClient = 10	78
* 2.16.3 kReadCommitted = 1	79
* 2.16.3 kReadUncommitted = 0	79
* 2.16.3 kRepeatableRead = 2	79
* 2.16.3 kSerializable = 3	79
* 2.16.3 kSQLBaseClient = 5	79
* 2.16.3 kSQLiteClient = 11	80
* 2.16.3 kSQLServerClient = 3	80
* 2.16.3 kSybaseClient = 8	80
- 2.18 class SQLAPIMBS	81
* 2.18.1 Connection as SQLConnectionMBS	82
- 2.17 class SQLBLobMBS	80
* 2.17.1 Constructor(data as SQLStringMBS)	81
* 2.17.1 Constructor(dataProvider as SQLDataProviderMBS, BlockSize as UInt32)	81
- 2.21 class PostgreSQLAPIMBS	83
* 2.21.1 DB(conn as SQLConnectionMBS) as string	84
* 2.21.1 ErrorMessage(conn as SQLConnectionMBS) as string	84
* 2.21.1 Field(cmd as SQLCommandMBS, RecordIndex as integer, FieldIndex as integer) as string	84
* 2.21.1 Field(cmd as SQLCommandMBS, RecordIndex as integer, FieldName as string) as string	84

* 2.21.1 FieldCount(cmd as SQLCommandMBS) as integer	84
* 2.21.1 Host(conn as SQLConnectionMBS) as string	85
* 2.21.1 Options(conn as SQLConnectionMBS) as string	85
* 2.21.1 Password(conn as SQLConnectionMBS) as string	85
* 2.21.1 Port(conn as SQLConnectionMBS) as string	85
* 2.21.1 RecordCount(cmd as SQLCommandMBS) as integer	85
* 2.21.1 TTY(conn as SQLConnectionMBS) as string	85
* 2.21.1 User(conn as SQLConnectionMBS) as string	86
– 2.19 class SQLCLOBMBS	82
* 2.19.1 Constructor(data as SQLStringMBS)	82
* 2.19.1 Constructor(dataProvider as SQLDataProviderMBS, BlockSize as UInt32)	82
– 2.20 class SQLBytesMBS	83
* 2.20.1 Constructor(data as SQLStringMBS)	83
– 2.23 class SQLDateTimeMBS	104
* 2.23.1 Constructor(other as SQLDateTimeMBS)	104
* 2.23.1 Constructor(value as double)	105
* 2.23.1 Constructor(Year as integer, Month as integer, Day as integer, Hour as integer, Minute as integer, Second as integer)	105
* 2.23.1 DoubleValue as double	106
* 2.23.1 Fraction as UInt32	106
* 2.23.1 GetDay as integer	106
* 2.23.1 GetDayOfWeek as integer	107
* 2.23.1 GetDayOfYear as integer	107
* 2.23.1 GetHour as integer	107
* 2.23.1 GetMinute as integer	107
* 2.23.1 GetMonth as integer	107
* 2.23.1 GetSecond as integer	107
* 2.23.1 GetYear as integer	108
* 2.23.1 StringValue as string	108
– 2.28 class SQLDataConsumerMBS	116
* 2.28.1 Write(PieceType as integer, data as string, Length as UInt32, BlobSize as UInt32)	116
* 2.28.2 kFirstPiece=1	116
* 2.28.2 kLastPiece=3	116
* 2.28.2 kNextPiece=2	117
* 2.28.2 kOnePiece=4	117
– 2.24 class SQLDataProviderMBS	108
* 2.24.1 Read(byref PieceType as integer, Length as UInt32) as string	108
* 2.24.2 kFirstPiece=1	109
* 2.24.2 kLastPiece=3	109
* 2.24.2 kNextPiece=2	109

* 2.24.2 kOnePiece=4	109
– 2.27 class SQLFieldMBS	113
* 2.27.1 FieldNativeType as integer	113
* 2.27.1 FieldPrecision as integer	113
* 2.27.1 FieldScale as integer	114
* 2.27.1 FieldSize as integer	114
* 2.27.1 FieldType as integer	114
* 2.27.1 isFieldRequired as boolean	114
* 2.27.1 Name as string	114
* 2.27.1 Option(name as string) as string	115
* 2.27.1 Pos as integer	115
* 2.27.1 ReadLongOrLob(consumer as SQLDataConsumerMBS, BlockSize as integer)	115
– 2.29 class SQLDatabaseMBS	117
* 2.29.1 Connect as boolean	118
* 2.29.1 GetConnection as SQLConnectionMBS	119
* 2.29.1 Option(name as string) as string	119
* 2.29.1 SetFileOption(name as string, file as folderitem)	119

Chapter 2

SQL

2.1 class SQLPositionMBS

class SQLPositionMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class for a position value.

2.1.1 Methods

Constructor(withID as integer)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new Position value with an ID.

See also:

- 2.1.1 Constructor(withName as string)

13

Constructor(withName as string)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new Position value with a name.

See also:

- 2.1.1 Constructor(withID as integer) 13

2.2 class SQLPreparedStatementMBS

class SQLPreparedStatementMBS

Plugin Version: 11.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class for prepared statements if you work with SQLDatabaseMBS class.

Notes:

If you work with SQLCommandMBS class, you can set parameters there directly.

For the SQL string you number parameters with colon and number. Like this: :1, :2, :3.

2.2.1 Methods

Bind(values() As Variant)

Plugin Version: 11.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets the value list with the given values.

Notes: You can either pass values to the SQLExecute/SQLSelect method or call Bind methods to set values. You have to define for each parameter both the type and the value.

See also:

- 2.2.1 Bind(zeroBasedIndex As Integer, value As Variant) 14
- 2.2.1 Bind(zeroBasedIndex As Integer, value As Variant, type as integer) 15

Bind(zeroBasedIndex As Integer, value As Variant)

Plugin Version: 11.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Defines the value for one parameter.

Notes: You can either pass values to the SQLExecute/SQLSelect method or call Bind methods to set values.

2.2. CLASS SQLPREPAREDSTATEMENTMBS 15

You have to define for each parameter both the type and the value.
See also:

- 2.2.1 Bind(values() As Variant) 14
- 2.2.1 Bind(zeroBasedIndex As Integer, value As Variant, type as integer) 15

Bind(zeroBasedIndex As Integer, value As Variant, type as integer)

Plugin Version: 11.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Defines one parameter with value and type.

Notes: You can either pass values to the SQLExecute/SQLSelect method or call Bind methods to set values.
You have to define for each parameter both the type and the value.
See also:

- 2.2.1 Bind(values() As Variant) 14
- 2.2.1 Bind(zeroBasedIndex As Integer, value As Variant) 14

BindType(types() As Integer)

Plugin Version: 11.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Defines the types for all values.

Notes: You can either pass values to the SQLExecute/SQLSelect method or call Bind methods to set values.
You have to define for each parameter both the type and the value.
See also:

- 2.2.1 BindType(zeroBasedIndex As Integer, type As Integer) 15

BindType(zeroBasedIndex As Integer, type As Integer)

Plugin Version: 11.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Defines the type of one value.

Notes: You can either pass values to the SQLExecute/SQLSelect method or call Bind methods to set values.
You have to define for each parameter both the type and the value.
See also:

- 2.2.1 BindType(types() As Integer) 15

Constructor

Plugin Version: 11.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The private constructor.

Notes:

This constructor makes sure you don't create useless `SQLPreparedStatementMBS` objects by error. The only way to create an object is to use the `prepare` method in the database class.

This constructor is private to make sure you don't create an object from this class by error. Please use designated functions to create objects.

`SQLExecute(ParamArray bindItems As Variant)`

Plugin Version: 11.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Runs the SQL command with the given parameters.

Notes: You can decide whether you pass values here or call `Bind` methods.

`SQLSelect(ParamArray bindItems As Variant) As RecordSet`

Plugin Version: 11.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Runs the query with the given parameters.

Notes:

Returns the recordset object or nil on error.

You can decide whether you pass values here or call `Bind` methods.

2.2.2 Constants

`kTypeBlob = 14`

Plugin Version: 11.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the data type constants.

Notes: Binary large Object.

kTypeBool = 1

Plugin Version: 11.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the data type constants.

Notes: Boolean

kTypeBytes = 11

Plugin Version: 11.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the data type constants.

Notes: a binary string.

kTypeClob = 15

Plugin Version: 11.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the data type constants.

Notes: Character Large Object

kTypeDateTime = 8

Plugin Version: 11.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the data type constants.

Notes: Date and/or Time.

kTypeDouble = 6

Plugin Version: 11.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the data type constants.

Notes: double float value.

kTypeLong = 4

Plugin Version: 11.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the data type constants.

Notes: signed 32 bit integer

kTypeLongBinary = 12

Plugin Version: 11.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the data type constants.

Notes: Long binary.

kTypeLongChar = 13

Plugin Version: 11.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the data type constants.

Notes: Long string.

kTypeShort = 2

Plugin Version: 11.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the data type constants.

Notes: signed 16 bit integer

kTypeString = 10

Plugin Version: 11.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the data type constants.

Notes: String

kTypeULong = 5

Plugin Version: 11.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the data type constants.

Notes: unsigned 32 bit integer

kTypeUnknown = 0

Plugin Version: 11.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the data type constants.

Notes: unknown type

kTypeUShort = 3

Plugin Version: 11.2 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the data type constants.

Notes: unsigned 16 bit integer

2.3 class SQLStringMBS

class SQLStringMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class for strings in this plugin.

Example:

```
dim s as new SQLStringMBS("Hello ")
```

```
MsgBox "Characters: "+str(s.GetLength)+" Bytes: "+str(s.GetBinaryLength)
```

```
dim a as string= s.CopyBinaryData
```

```
dim b as string= s.CopyText
```

```
MsgBox a // RB shows garbage as it tries to display bytes as UTF8 which does not work
```

```
MsgBox b // displays correct
```

Notes: A string can be text (with text encoding) or bytes (raw binary data).

2.3.1 Methods

Compare(text as **SQLStringMBS**) as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Compares this string object with another string.

Notes:

Function performs a case-sensitive comparison of the strings, and is not affected by locale.

Returns zero if the strings are identical, <0 if this string object is less than text, or >0 if this string object is greater than text.

See also:

- 2.3.1 Compare(text as string) as integer 20

Compare(text as string) as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Compares this string object with another string.

Notes:

Function performs a case-sensitive comparison of the strings, and is not affected by locale.

Returns zero if the strings are identical, <0 if this string object is less than text, or >0 if this string object is greater than text.

See also:

- 2.3.1 Compare(text as **SQLStringMBS**) as integer 20

CompareNoCase(text as SQLStringMBS) as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Compares this string object with another string.

Notes:

Function performs a case-insensitive comparison of the strings, and is not affected by locale.

Returns zero if the strings are identical (ignoring case), <0 if this string object is less than text (ignoring case), or >0 if this string object is greater than text (ignoring case).

See also:

- 2.3.1 CompareNoCase(text as string) as integer 21

CompareNoCase(text as string) as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Compares this string object with another string.

Notes:

Function performs a case-insensitive comparison of the strings, and is not affected by locale.

Returns zero if the strings are identical (ignoring case), <0 if this string object is less than text (ignoring case), or >0 if this string object is greater than text (ignoring case).

See also:

- 2.3.1 CompareNoCase(text as SQLStringMBS) as integer 21

Constructor

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new empty string object.

See also:

- 2.3.1 Constructor(Data as string, isText as boolean=true) 22
- 2.3.1 Constructor(other as SQLStringMBS) 22

Constructor(Data as string, isText as boolean=true)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new string object with data or text copied from the data string.

Notes: If isText is true, the data is interpreted as text and string encoding conversion may modify it. If isText is false the bytes are copied raw.

See also:

- 2.3.1 Constructor 21
- 2.3.1 Constructor(other as SQLStringMBS) 22

Constructor(other as SQLStringMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new string object with data copied from the other string object.

See also:

- 2.3.1 Constructor 21
- 2.3.1 Constructor(Data as string, isText as boolean=true) 22

CopyBinaryData as string

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies the bytes from the internal buffer ignoring any text encoding.

CopyText as string

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies the characters of this string as text.

Notes: Text encoding conversion may happen.

Empty

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Forces a string to have 0 length.

GetBinaryLength as UInt32

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns a count of the bytes in the binary data buffer.

GetLength as UInt32

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the number of characters in a SAString object.

Notes: For multibyte character sets, GetLength counts each 8-bit character; that is, a lead and trail byte in one multibyte character are counted as two bytes.

IsEmpty as boolean

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Tests whether a String object contains no characters.

Notes: Returns true if length is zero.

Left(count as integer) as SQLStringMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Extracts the left part of a string.

MakeLower

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Changes all characters in the string to lower case.

MakeUpper

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Changes all characters in the string to upper case.

Mid(first as integer) as SQLStringMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Extracts the middle part of a string.

Notes: first: The zero-based index of the first character in this string object that is to be included in the extracted substring.

See also:

- 2.3.1 Mid(first as integer, Count as integer) as SQLStringMBS 24

Mid(first as integer, Count as integer) as SQLStringMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Extracts the middle part of a string.

Notes:

first: The zero-based index of the first character in this string object that is to be included in the extracted substring.

count: The number of characters to extract from this string object. If this parameter is not supplied, then the remainder of the string is extracted.

See also:

- 2.3.1 Mid(first as integer) as SQLStringMBS 24

Right(count as integer) as SQLStringMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Extracts the right part of a string.

TrimLeft

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Trim leading whitespace characters from the string.

TrimRight

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Trim trailing whitespace characters from the string.

2.4 class SQLParamMBS

class SQLParamMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The SQL class for parameters.

Notes: Subclass of the SQLValueMBS class.

2.4.1 Methods

Name as string

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The name of the parameter.

Option(name as string) as string

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The string value of a specific parameter option.

Notes:

see also:

http://www.sqlapi.com/OnLineDoc/Param_Option.html
(Read and Write computed property)

ParamDirType as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The direction type of parameter (input, output, etc.).

Notes:

Use the kParamDirType* constants.

Usually the Library automatically detects parameter's direction type and implicitly creates an appropriate SAParam object. But not all of DBMS clients/servers provide complete parameters information. In that situation programmer need to describe parameter's direction type explicitly. See Server specific notes for details.

http://www.sqlapi.com/OnLineDoc/Param_ParamDirType.html
(Read and Write computed property)

ParamNativeType as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The native type code of the parameter.

Notes: (Read and Write computed property)

ParamPrecision as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The precision of the parameter value (the total number of allowable digits).

Notes: (Read and Write computed property)

ParamScale as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The scale of the parameter value (the number of digits to the right of the decimal point).

Notes: (Read and Write computed property)

ParamSize as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The parameter's data size.

Notes: (Read and Write computed property)

ParamType as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The parameter's data type.

Notes:

See the `kDataType` constants.

(Read and Write computed property)

ReadLongOrLob(consumer as SQLDataConsumerMBS, BlockSize as integer)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The Long or Lob data reading mode.

Notes:

SQLAPI++ Library provides two ways to read Long or BLOB(CLOB) object's value (usually SQLField or SQLParam objects):

1. reading of Long or Lob data at once into an internal buffer (like ordinary string or binary values);
 2. piecewise reading of Long or Lob data using user defined callback.
- `kLongOrLobReaderDefault` reading mode used by default.

If you want to control piecewise reading of Long or BLOB(CLOB) data you should set `LongOrLobReaderMode`

and use `kLongOrLobReaderManual` reading mode for Long or BLOB(CLOB) parameters or fields you want to process with your data consumer. After that each fetch will skip reading Long and BLOB(CLOB) parameters that you set to be read manually. To read field or parameter defined to be read manually you should call `ReadLongOrLob` method for each of them after the fetch. `ReadLongOrLob` method will repeatedly call the data consumer Write event.

2.4.2 Constants

`kParamDirTypeInput=0`

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the parameter direction type constants.

Notes: Input parameter.

`kParamDirTypeInputOutput=1`

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the parameter direction type constants.

Notes: Input/output parameter.

`kParamDirTypeOutput=2`

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the parameter direction type constants.

Notes: Output parameter.

`kParamDirTypeReturn=3`

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the parameter direction type constants.

Notes: Returning parameter.

2.5 class SQLIntervalMBS

class SQLIntervalMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class in the SQL Plugin for an interval.

2.5.1 Methods

Constructor

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new zero interval.

See also:

- 2.5.1 Constructor(days as integer, hours as integer, minutes as integer, seconds as integer) 29
- 2.5.1 Constructor(value as double) 29

Constructor(days as integer, hours as integer, minutes as integer, seconds as integer)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new interval with the given values.

See also:

- 2.5.1 Constructor 29
- 2.5.1 Constructor(value as double) 29

Constructor(value as double)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new interval with the given time delta.

Example:

`dim n as new SQLIntervalMBS(5)`

MsgBox n.StringValue // shows "120:00:00" for 120 hours

See also:

- 2.5.1 Constructor 29
- 2.5.1 Constructor(days as integer, hours as integer, minutes as integer, seconds as integer) 29

Dec(interval as SQLIntervalMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Decrements the interval.

DoubleValue as double

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The value of this interval.

Example:

```
dim n as new SQLIntervalMBS(1,2,3,4)
```

MsgBox str(n.DoubleValue) // shows "1.085463" for 1 day, 2 hours, 3 minutes and 4 seconds

GetDays as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The days in this interval.

GetHours as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The hours value.

GetMinutes as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The minutes value.

GetSeconds as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The seconds value.

GetTotalDays as double

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The total days value.

Example:

```
dim n as new SQLIntervalMBS(1,2,3,4)
MsgBox str(n.GetTotalDays) // shows "1"
```

GetTotalHours as double

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The total hours value.

Example:

```
dim n as new SQLIntervalMBS(1,2,3,4)
MsgBox str(n.GetTotalHours) // shows "26"
```

GetTotalMinutes as double

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The total minutes value.

Example:

```
dim n as new SQLIntervalMBS(1,2,3,4)
MsgBox str(n.GetTotalMinutes) // shows "1563"
```

GetTotalSeconds as double

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The total seconds value.

Notes:

```
dim n as new SQLIntervalMBS(1,2,3,4)
MsgBox str(n.GetTotalSeconds) // shows "93784"
```

Inc(interval as SQLIntervalMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Increments the interval.

SetInterval(days as integer, hours as integer, minutes as integer, seconds as integer)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets the interval values.

StringValue as string

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The interval as a string.

Example:

```
dim n as new SQLIntervalMBS(5)
MsgBox n.StringValue // shows "120:00:00" for 120 hours
```

2.6 class *SQLite3MBS*

class *SQLite3MBS*

Plugin Version: 9.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class for the native SQLite API.

Example:

```
dim con as new SqlConnectionMBS
dim path as string = "/tmp/test.db" // change path for Windows!

con.Connect(path, "", "", SqlConnectionMBS.kSQLiteClient)

dim api as SQLAPIMBS = con.NativeAPI
if api isa SQLite3MBS then
dim s as SQLite3MBS = SQLite3MBS(api)
MsgBox s.Version
end if
```

Notes: Subclass of the *SQLAPIMBS* class.

2.6.1 Methods

Version as string

Plugin Version: 9.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The version string of the SQLite library.

VersionNumber as integer

Plugin Version: 9.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The version number of the SQLite library.

2.7 class SQLNumericMBS

class SQLNumericMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class for numeric values.

2.7.1 Methods

Constructor

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates an empty numeric object.

See also:

- 2.7.1 Constructor(value as double) 34
- 2.7.1 Constructor(value as string) 34

Constructor(value as double)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new numeric object based on the given double value.

See also:

- 2.7.1 Constructor 34
- 2.7.1 Constructor(value as string) 34

Constructor(value as string)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new numeric object based on the given string.

See also:

2.7. CLASS SQLNUMERICMBS 35

- 2.7.1 Constructor 34
- 2.7.1 Constructor(value as double) 34

DoubleValue as double

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The double value for this number.

Int64Value as Int64

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The number value as an int64.

precision as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The maximum number of digits in base 10.

scale as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The number of digits to the right of the decimal point.

sign as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The sign: 1 for positive numbers, 0 for negative numbers.

StringValue as string

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The string value of this number.

UInt64Value as UInt64

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The number value as an uint64.

2.8 class SQLLongBinaryMBS**class SQLLongBinaryMBS**

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class for a long binary object.

Notes:

Basicly this is a SQLStringMBS which is always marked to contain binary data. You only need this class to use the constructor with dataprovider to stream data to the database.
Subclass of the SQLLongOrLobMBS class.

2.8.1 Methods**Constructor(data as SQLStringMBS)**

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new long binary object from a string object.
See also:

- 2.8.1 Constructor(dataProvider as SQLDataProviderMBS, BlockSize as UInt32)

Constructor(dataProvider as SQLDataProviderMBS, BlockSize as UInt32)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new long binary object from a data provider.

Notes:

The blocksize specifies in which sizes data is requested from the provider.

You must make sure that the data provider and this new blob object life long enough. Because the actual data is requested later when you do the update on the database.

If BlockSize is 0, the default block size is used.

See also:

- 2.8.1 Constructor(data as SQLStringMBS)

36

2.9 class SQLLongCharMBS**class SQLLongCharMBS**

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class for the long character data type.

Notes:

Basicly this is a SQLStringMBS which is always marked to contain text. You only need this class to use the constructor with dataprovider to stream data to the database.

Subclass of the SQLLongOrLobMBS class.

2.9.1 Methods**Constructor(data as SQLStringMBS)**

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new long character object from a string object.

See also:

- 2.9.1 Constructor(dataProvider as SQLDataProviderMBS, BlockSize as UInt32)

38

Constructor(dataProvider as SQLDataProviderMBS, BlockSize as UInt32)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new long character object from a data provider.

Notes:

The blocksize specifies in which sizes data is requested from the provider.

You must make sure that the data provider and this new long character object life long enough. Because the actual data is requested later when you do the update on the database.

If BlockSize is 0, the default block size is used.

See also:

- 2.9.1 Constructor(data as SQLStringMBS)

37

2.10 class SQLNotInitializedExceptionMBS

class SQLNotInitializedExceptionMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The exception raised if you call a method on an object which was not properly initialized.

Notes: Subclass of the SQLExceptionMBS class.

2.11 class SQLNullMBS

class SQLNullMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class used internally for null values.

2.12 class SQLLongOrLobMBS

class SQLLongOrLobMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The super class for Long Binary/Text and BLOB/CLOB classes.

Notes: Subclass of the SQLStringMBS class.

2.13 class SQLValueReadMBS

class SQLValueReadMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class used in the SQL Plugin for value objects which can be read.

2.13.1 Methods

asBlob as SQLStringMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the value as Blob (SQLString) data.

Notes:

If the value of current object is NULL, asBlob method returns an empty string. Use isNull method to make sure if the value is NULL or not.

If the value's type of current object is string (kDataTypeString), bytes (kDataTypeBytes), long binary (kDataTypeLongBinary), long character (kDataTypeLongChar), Blob (kDataTypeBlob) or Clob (kDataTypeClob), asBlob method returns the object's value as SQLString object.

If the value's type of current object is bool (kDataTypeBool), short (kDataTypeShort), long (kDataTypeLong), double (kDataTypeDouble), numeric (kDataTypeNumeric), date-time (kDataTypeDateTime) or cursor (kDataTypeCursor), the result is undefined and debug version asserts.

Use `DataType` method to get the value's type of `SQLValueRead` object.

asBool as boolean

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the value of current object as bool data.

Notes:

If the value of current object is NULL, `asBool` method returns false. Use `isNull` method to make sure if the value is NULL or not.

If the value's type of current object is bool (`kDataTypeBool`), `asBool` method returns the original value with no conversion.

If the value's type of current object is short (`kDataTypeShort`), long (`kDataTypeLong`) or double (`kDataTypeDouble`), `asBool` method converts it to bool data type. Conversion returns false if the value is 0; true otherwise.

If the value's type of current object is any data type except the described above, the result is undefined and debug version asserts.

Use `DataType` method to get the value's type of `SQLValueRead` object.

asBytes as SQLStringMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the value as binary string data (`SQLString`).

Notes:

If the value of current object is NULL, `asBytes` method returns an empty string. Use `isNull` method to make sure if the value is NULL or not.

If the value's type of current object is string (`kDataTypeString`), bytes (`kDataTypeBytes`), long binary (`kDataTypeLongBinary`), long character (`kDataTypeLongChar`), BLOB (`kDataTypeBlob`) or CLOB (`kDataTypeClob`), `asBytes` method returns the object's value as `SQLString` object.

If the value's type of current object is bool (kDataTypeBool), short (kDataTypeShort), long (kDataTypeLong), double (kDataTypeDouble), numeric (kDataTypeNumeric) or date-time (kDataTypeDateTime), asBytes method returns a block of data with size sizeof(value's type) as SQLString object.

If the value's type of current object is cursor (kDataTypeCursor), the result is undefined and debug version asserts.

Use DataType method to get the value's type of SQLValueRead object.

asCLob as SQLStringMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the value as CLob (SQLString) data.

Notes:

If the value of current object is NULL, asCLob method returns an empty string. Use isNull method to make sure if the value is NULL or not.

If the value's type of current object is string (kDataTypeString), bytes (kDataTypeBytes), long binary (kDataTypeLongBinary), long character (kDataTypeLongChar), BLOB (kDataTypeBLOB) or CLob (kDataTypeCLob), asCLob method returns the object's value as SQLString object.

If the value's type of current object is bool (kDataTypeBool), short (kDataTypeShort), long (kDataTypeLong), double (kDataTypeDouble), numeric(kDataTypeNumeric), date-time (kDataTypeDateTime) or cursor (kDataTypeCursor), the result is undefined and debug version asserts.

Use DataType method to get the value's type of SQLValueRead object.

asDate as Date

Plugin Version: 11.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the value as date.

Example:

```
dim cmd as SQLCommandMBS // your command object
dim d as date = cmd.Field("mydate").asDate // read date value
```

Notes:

If the value of current object is NULL, asDate method returns an empty date object. Use isNull method to make sure if the value is NULL or not.

If the value's type of current object is kDataTypeDateTime, asDate method returns date object.

If the value's type of current object is any data type except kDataTypeDateTime, the result is undefined and debug version asserts.

Use DataType method to get the value's type of SQLValueRead object.

asDateTime as SQLDateTimeMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the value as SQLDateTimeMBS object.

Notes:

If the value of current object is NULL, asDateTime method returns an empty SQLDateTime object. Use isNull method to make sure if the value is NULL or not.

If the value's type of current object is kDataTypeDateTime, asDateTime method returns SQLDateTime object.

If the value's type of current object is any data type except kDataTypeDateTime, the result is undefined and debug version asserts.

Use DataType method to get the value's type of SQLValueRead object.

asDouble as double

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** returns the value as double data.

Notes:

If the value of current object is NULL, asDouble method returns 0. Use isNull method to make sure if the value is NULL or not.

If the value's type of current object is double (kDataTypeDouble), asDouble method returns the original value with no conversion.

If the value's type of current object is bool (kDataTypeBool), short (kDataTypeShort), long (kDataTypeLong) or numeric (kDataTypeNumeric), asDouble method converts it to double (kDataTypeDouble) data type.

If the value's type of current object is string (kDataTypeString), asDouble method tries to convert it to double value. If the conversion is possible and correct, asDouble returns kDataTypeDouble value. If conversion is incorrect asDouble method throws an exception.

If the value's type of current object is kDataTypeDateTime, asDouble method converts it to standard double representation. Days are represented by whole number increments starting with 30 December 1899, midnight as time zero. Hour values are expressed as the absolute value of the fractional part of the number. See `SQLDateTime::operator double()` for more details.

If the value's type of current object is any data type except the described above, the result is undefined and debug version asserts.

Use `DataType` method to get the value's type of `SQLValueRead` object.

asInterval as SQLIntervalMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the value as interval (`SQLIntervalMBS`).

asLong as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the value as long data.

Notes:

If the value of current object is NULL, asLong method returns 0. Use isNull method to make sure if the value is NULL or not.

If the value's type of current object is long (kDataTypeLong), asLong method returns the original value with no conversion.

If the value's type of current object is bool (kDataTypeBool), short (kDataTypeShort), double (kDataTypeDouble) or numeric (kDataTypeNumeric), asLong method converts it to long data type. Note, that in this case the returned value can be truncated.

If the value's type of current object is string (kDataTypeString), asLong method tries to convert it to long (kDataTypeLong) value. If the conversion is possible and correct, asLong returns kDataTypeLong value. If conversion is incorrect asLong method throws an exception.

If the value's type of current object is any data type except the described above, the result is undefined and debug version asserts.

Use DataType method to get the value's type of SQLValueRead object.

asLongBinary as SQLStringMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the value as long binary (SQLString) data.

Notes:

If the value of current object is NULL, asLongBinary method returns an empty string. Use isNull method to make sure if the value is NULL or not.

If the value's type of current object is string (kDataTypeString), bytes (kDataTypeBytes), long binary (kDataTypeLongBinary), long character (kDataTypeLongChar), BLOB (kDataTypeBLOB) or CLOB (kDataType-

CLob), asLongBinary method returns the object's value as SQLString object.

If the value's type of current object is bool (kDataTypeBool), short (kDataTypeShort), long (kDataTypeLong), double (kDataTypeDouble), numeric (kDataTypeNumeric), date-time (kDataTypeDateTime) or cursor (kDataTypeCursor), the result is undefined and debug version asserts.

Use DataType method to get the value's type of SQLValueRead object.

asLongChar as SQLStringMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the value as long character (SQLString) data.

Notes:

If the value of current object is NULL, asLongChar method returns an empty string. Use isNull method to make sure if the value is NULL or not.

If the value's type of current object is string (kDataTypeString), bytes (kDataTypeBytes), long binary (kDataTypeLongBinary), long character (kDataTypeLongChar), BLOB (kDataTypeBLOB) or CLOB (kDataTypeCLOB), asLongChar method returns the object's value as SQLString object.

If the value's type of current object is bool (kDataTypeBool), short (kDataTypeShort), long (kDataTypeLong), double (kDataTypeDouble), numeric (kDataTypeNumeric), date-time (kDataTypeDateTime) or cursor (kDataTypeCursor), the result is undefined and debug version asserts.

Use DataType method to get the value's type of SQLValueRead object.

asNumeric as SQLNumericMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the value as SQLNumeric data.

Notes:

If the value of current object is NULL, asNumeric method returns 0. Use isNull method to make sure if the value is NULL or not.

If the value's type of current object is exact numeric value (`kDataTypeNumeric`), `asNumeric` method returns the original value with no conversion.

If the value's type of current object is `bool` (`kDataTypeBool`), `short` (`kDataTypeShort`), `double` (`kDataTypeDouble`) or `long` (`kDataTypeLong`), `asNumeric` method converts it to `SQLNumeric` .

If the value's type of current object is `string` (`kDataTypeString`), `asNumeric` method tries to convert it from `SQLChar*` value. If the conversion is possible and correct, `asNumeric` converts to `SQLNumeric` from `SQLChar*` value. If conversion is incorrect `asNumeric` method throws an exception.

If the value's type of current object is any data type except the described above, the result is undefined and debug version asserts.

Use `DataType` method to get the value's type of `SQLValueRead` object.

asShort as Int16

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the value as short.

Notes:

If the value of current object is `NULL`, `asShort` method returns 0. Use `isNull` method to make sure if the value is `NULL` or not.

If the value's type of current object is `short`, `asShort` method returns the original value with no conversion.

If the value's type of current object is `bool` (`kDataTypeBool`), `long` (`kDataTypeLong`), `unsigned long` (`SkDataTypeULong`), `double` (`kDataTypeDouble`) or `numeric` (`kDataTypeNumeric`), `asShort` method converts it to short data type. Note, that in this case the returned value can be truncated.

If the value's type of current object is `string` (`kDataTypeString`), `asShort` method tries to convert it to short value. If the conversion is possible and correct, `asShort` returns the value. If conversion is incorrect `asShort` method throws an exception.

If the value's type of current object is any data type except the described above, the result is undefined and debug version asserts.

Use `DataType` method to get the value's type of `SValueRead` object.

asString as SQLStringMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** returns the value as string (`SQLString`) data.

Notes:

If the value of current object is `NULL`, `asString` method returns an empty string. Use `isNull` method to make sure if the value is `NULL` or not.

If the value's type of current object is `bool` (`kDataTypeBool`), `asString` method returns "true" or "false" string (`SQLString` object).

If the value's type of current object is `short` (`kDataTypeShort`), `asString` method converts it to decimal string (`SQLString` object) like function `printf("%hd", ...)` does.

If the value's type of current object is `long` (`kDataTypeLong`), `asString` method converts it to decimal string (`SQLString` object) like function `printf("%ld", ...)` does.

If the value's type of current object is `double` (`kDataTypeDouble`), `asString` method converts it to decimal string (`SQLString` object) like function `printf("%g", ...)` does.

If the value's type of current object is `numeric` (`kDataTypeNumeric`), `asString` method converts it to decimal string (`SQLString` object) without precision loss.

If the value's type of current object is `date-time` (`kDataTypeDateTime`), `asString` method converts it to string (`SQLString` object) like function `asctime(...)` does.

If the value's type of current object is `string` (`kDataTypeString`, `kDataTypeLongChar`, `kDataTypeCLob`), `asString` method returns the original object's value as `SQLString` object.

If the value's type of current object is binary (`kDataTypeBytes`, `kDataTypeLongBinary`, `kDataTypeBLob`), `asString` method converts it to hexadecimal string (`SQLString` object).

If the value's type of current object is cursor (`kDataTypeCursor`), the result is undefined and debug version asserts.

Use `DataType` method to get the value's type of `SQLValueRead` object.

asStringValue as String

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** returns the value string.

Notes: Same as `asString` but returns a REALbasic string.

asULong as UInt32

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the value as an unsigned 32 bit integer value.

Notes:

If the value of current object is `NULL`, `asULong` method returns 0. Use `isNull` method to make sure if the value is `NULL` or not.

If the value's type of current object is long (`kDataTypeLong`), `asULong` method returns the original value with no conversion.

If the value's type of current object is bool (`kDataTypeBool`), short (`kDataTypeShort`), double (`kDataTypeDouble`) or numeric (`kDataTypeNumeric`), `asULong` method converts it to long data type. Note, that in this case the returned value can be truncated.

If the value's type of current object is string (`kDataTypeString`), `asULong` method tries to convert it to long (`kDataTypeLong`) value. If the conversion is possible and correct, `asULong` returns `kDataTypeLong` value. If conversion is incorrect `asULong` method throws an exception.

If the value's type of current object is any data type except the described above, the result is undefined and debug version asserts.

Use `DataType` method to get the value's type of `SQLValueRead` object.

asUShort as UInt16

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the value as unsigned short.

Notes:

If the value of current object is `NULL`, `asUShort` method returns 0. Use `isNull` method to make sure if the value is `NULL` or not.

If the value's type of current object is unsigned short, `asUShort` method returns the original value with no conversion.

If the value's type of current object is `bool` (`kDataTypeBool`), `long` (`kDataTypeLong`), unsigned long (`SkDataTypeULong`), `double` (`kDataTypeDouble`) or `numeric` (`kDataTypeNumeric`), `asUShort` method converts it to unsigned short data type. Note, that in this case the returned value can be truncated.

If the value's type of current object is `string` (`kDataTypeString`), `asUShort` method tries to convert it to unsigned short value. If the conversion is possible and correct, `asUShort` returns the value. If conversion is incorrect `asUShort` method throws an exception.

If the value's type of current object is any data type except the described above, the result is undefined and debug version asserts.

Use `DataType` method to get the value's type of `SValueRead` object.

Constructor(DataType as integer)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new empty value object for the given data type.

See also:

- 2.13.1 Constructor(value as SQLValueReadMBS) 50

Constructor(value as SQLValueReadMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new value object by copying the given one.

See also:

- 2.13.1 Constructor(DataType as integer) 49

DataType as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns SAValueRead object's data type.

Notes: One of the kDataType constants.

isNull as boolean

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns true if the value of current object is NULL; otherwise false.

LongOrLobReaderMode as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The Long or Lob data reading mode.

Notes:

SQLAPI++ Library provides two ways to read Long or BLOB(CLOB) object's value (usually SQLField or SQLParam objects):

1. reading of Long or Lob data at once into an internal buffer (like ordinary string or binary values);
2. piecewise reading of Long or Lob data using user defined callback.

kLongOrLobReaderDefault reading mode used by default.

If you want to control piecewise reading of Long or BLOB(CLOB) data you should set LongOrLobReaderMode and use kLongOrLobReaderManual reading mode for Long or BLOB(CLOB) parameters or fields you want to process with your data consumer. After that each fetch will skip reading Long and BLOB(CLOB) parameters that you set to be read manually. To read field or parameter defined to be read manually you should call ReadLongOrLob method for each of them after the fetch. ReadLongOrLob method will repeatedly call the data consumer Write event.

(Read and Write computed property)

2.13.2 Constants

kDataTypeBlob=14

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: Data type is BLOB data (SQLStringMBS).

kDataTypeBool=1

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: Data type is a boolean.

kDataTypeBytes=11

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: Data type is binary string (SQLStringMBS).

kDataTypeClob=15

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: Data type is Clob data (SQLStringMBS).

kDataTypeCursor=16

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: Data type is Oracle REF CURSOR (SQLCommand).

kDataTypeDateTime=8

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: Data type is SQLDateTime.

kDataTypeDouble=6

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: This is a normal double variable.

kDataTypeInterval=9

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: Data type is an interval (SQLIntervalMBS).

kDataTypeLong=4

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: A 32 bit integer.

kDataTypeLongBinary=12

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: Data type is long binary data (SQLStringMBS).

kDataTypeLongChar=13

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: Data type is long character data (SQLStringMBS).

kDataTypeNumeric=7

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: Data type is SQLNumeric (used internally).

kDataTypeShort=2

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: Data type is a 16 bit signed integer.

kDataTypeSpecificToDBMS=17

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: Data type is server-specific.

kDataTypeString=10

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: Data type is character string (SQLString).

kDataTypeULong=5

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: Data type is a 32 bit unsigned integer.

kDataTypeUnknown=0

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: Data type is unknown.

kDataTypeUShort=3

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the field type constants

Notes: A 16 bit unsigned integer.

kLongOrLobReaderModeDefault=0

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the read modes.

Notes: Long or Lob(CLOB) data reading mode is default (automatic).

kLongOrLobReaderModeManual=1

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the read modes.

Notes: Long or Lob(Clob) data reading mode is manual.

2.14 class SQLUnsupportedExceptionMBS

class SQLUnsupportedExceptionMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class for an exception to report that the function is not supported on this platform.

Notes:

This one raises only if the plugin is compiled for Mac OS Classic.
Subclass of the SQLExceptionMBS class.

2.15 class SQLValueMBS

class SQLValueMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The SQL class for mutable values.

Notes: Subclass of the SQLValueReadMBS class.

2.15.1 Methods

Constructor(DataType as integer)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new empty value object with the given data type.

isDefault as boolean

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns true if the plugin has been forced to use parameter's default value by calling setAsDefault method; false otherwise.

setAsBlob(data as SQLDataProviderMBS, BlockSize as UInt32)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as BLOB data (SQLString)

Notes:

When you call the SQLCommandMBS.Execute method all input parameters are bound with their values, including Long and BLOB(CLOB) parameters.

That is the time when the data provider Read event runs to get the values in the block size you specify.

The default value for the block size is 0. If you use the default value, SQLAPI++ Library will automatically use the most appropriate size for current DBMS.

See also:

- 2.15.1 setAsBlob(data as SQLStringMBS) 56
- 2.15.1 setAsBlob(data as string) 56

setAsBlob(data as SQLStringMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as BLOB data (SQLString)

See also:

- 2.15.1 setAsBlob(data as SQLDataProviderMBS, BlockSize as UInt32) 56
- 2.15.1 setAsBlob(data as string) 56

setAsBlob(data as string)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as BLOB data (SQLString)

See also:

- 2.15.1 setAsBlob(data as SQLDataProviderMBS, BlockSize as UInt32) 56
- 2.15.1 setAsBlob(data as SQLStringMBS) 56

setAsBool(value as boolean)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as bool data.

setAsBytes(value as SQLStringMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as binary string data (SQLString).

setAsClob(data as SQLDataProviderMBS, BlockSize as UInt32)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as Clob data (SQLString)

Notes:

nWhen you call the SQLCommandMBS.Execute method all input parameters are bound with their values, including Long and BLOB(Clob) parameters.

That is the time when the data provider Read event runs to get the values in the block size you specify.

The default value for the block size is 0. If you use the default value, SQLAPI++ Library will automatically use the most appropriate size for current DBMS.

See also:

- 2.15.1 setAsClob(text as SQLStringMBS) 57
- 2.15.1 setAsClob(text as string) 58

setAsClob(text as SQLStringMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as Clob data (SQLString)

See also:

- 2.15.1 `setAsClob(data as SQLDataProviderMBS, BlockSize as UInt32)` 57
- 2.15.1 `setAsClob(text as string)` 58

`setAsClob(text as string)`

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as CLOB data (SQLString)

See also:

- 2.15.1 `setAsClob(data as SQLDataProviderMBS, BlockSize as UInt32)` 57
- 2.15.1 `setAsClob(text as SQLStringMBS)` 57

`setAsDate(value as date)`

Plugin Version: 11.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value with a date.

Example:

```
dim cmd as SQLCommandMBS // your command object
dim d as new date(2012,12,24,16,0,0) // some date
cmd.Param(3).setAsDate(d) // set third parameter
```

Notes: Same as `setAsDateTime`, but here we take a date object to make it more convenient.

`setAsDateTime(value as SQLDateTimeMBS)`

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as SQLDateTime data.

`setAsDefault`

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Forces to use the default parameter's value.

Notes:

Forces DBMS API to use the default parameter value as the input value for an input or input/output parameter in a procedure.

If DBMS API does not support the concept of "default parameter values" in stored procedures, this setting will be ignored.

If you set this flag for the parameter that doesn't have a default value, the effect is DBMS defined (e.g. an error can be returned or NULL can be bound).

To cancel using the default parameter value you should call any other SQLValue::setAs... method to bind a parameter value.

To check whether this flag is set or not use isDefault method.

setAsDouble(value as double)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as double data.

setAsInterval(value as SQLIntervalMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets an interval value.

setAsLong(value as Int32)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as long data.

setAsLongBinary(data as SQLDataProviderMBS, BlockSize as UInt32)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as long binary data (SQLString)

Notes:

When you call the SQLCommandMBS.Execute method all input parameters are bound with their values, including Long and BLOB(CLOB) parameters.

That is the time when the data provider Read event runs to get the values in the block size you specify.

The default value for the block size is 0. If you use the default value, SQLAPI++ Library will automatically use the most appropriate size for current DBMS.

See also:

- 2.15.1 setAsLongBinary(data as SQLStringMBS) 60
- 2.15.1 setAsLongBinary(data as string) 60

setAsLongBinary(data as SQLStringMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as long binary data (SQLString)

See also:

- 2.15.1 setAsLongBinary(data as SQLDataProviderMBS, BlockSize as UInt32) 60
- 2.15.1 setAsLongBinary(data as string) 60

setAsLongBinary(data as string)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as long binary data (SQLString)

See also:

- 2.15.1 setAsLongBinary(data as SQLDataProviderMBS, BlockSize as UInt32) 60
- 2.15.1 setAsLongBinary(data as SQLStringMBS) 60

setAsLongChar(data as SQLDataProviderMBS, BlockSize as UInt32)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as long character data (SQLString)

Notes:

When you call the SQLCommandMBS.Execute method all input parameters are bound with their values, including Long and BLOB(CLOB) parameters.

That is the time when the data provider Read event runs to get the values in the block size you specify.

The default value for the block size is 0. If you use the default value, SQLAPI++ Library will automatically use the most appropriate size for current DBMS.

See also:

- 2.15.1 setAsLongChar(text as SQLStringMBS) 61
- 2.15.1 setAsLongChar(text as string) 61

setAsLongChar(text as SQLStringMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as long character data (SQLString)

See also:

- 2.15.1 setAsLongChar(data as SQLDataProviderMBS, BlockSize as UInt32) 61
- 2.15.1 setAsLongChar(text as string) 61

setAsLongChar(text as string)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as long character data (SQLString)

See also:

- 2.15.1 setAsLongChar(data as SQLDataProviderMBS, BlockSize as UInt32) 61
- 2.15.1 setAsLongChar(text as SQLStringMBS) 61

setAsNull

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets value to null.

setAsNumeric(value as SQLNumericMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as SQLNumeric data.

setAsShort(value as Int16)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as short data.

setAsString(text as string)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as character string data.

Notes: Same as setAsString, but for your convenience with a REALbasic string instead of a SQLStringMBS object.

See also:

- 2.15.1 setAsString(value as SQLStringMBS) 62

setAsString(value as SQLStringMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as character string data (SQLString)

See also:

- 2.15.1 setAsString(text as string) 62

setAsULong(value as UInt32)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as unsigned long data.

setAsUnknown

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's type as unknown.

setAsUShort(value as UInt16)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value as unsigned short data.

Notes: Sets value as unsigned short data.

setAsValueRead(value as SQLValueReadMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets parameter's value from SQLParam or SQLField objects.

Notes: This method allows using SQLField or SQLParam object received from one SQL statement as a parameter for another SQL statement.

2.16 class SQLConnectionMBS

class SQLConnectionMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class for a SQL Plugin Database connection.

Example:

`dim` con `as new` SQLConnectionMBS

```

try

// where is the library?
con.SetFileOption con.kOptionLibraryMySQL, SpecialFolder.UserHome.Child("libmysqlclient.dylib")

// connect to database
// in this example it is Oracle,
// but can also be Sybase, Informix, DB2
// SQLServer, InterBase, SQLBase and ODBC

dim server as string = "192.168.1.80:3306@test"

con.Connect(server,"root","",SQLConnectionMBS.kMySQLClient)

MsgBox "We are connected!"

// Disconnect is optional
// autodisconnect will occur in destructor if needed
con.Disconnect

msgbox "We are disconnected!"

catch r as RuntimeException
MsgBox r.message

// SAConnection::Rollback()
// can also throw an exception
// (if a network error for example),
// we will be ready
try

// on error rollback changes
con.Rollback

catch rr as runtimeexception
MsgBox rr.message
end try
end try

```

Notes: Supported databases: Oracle, Microsoft SQL Server, DB2, Sybase, Informix, InterBase/Firebird, SQLBase, MySQL, PostgreSQL and ODBC and SQLite

2.16.1 Methods

AutoCommit as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Whether autocommit is enabled or disabled for the current connection.

Notes:

If autocommit is on, the database is committed automatically after each SQL command. Otherwise, transaction is committed only after Commit calling.
(Read and Write computed property)

Client as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The current DBMS client assigned for the connection.

Notes: (Read and Write computed property)

ClientVersion as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Gets the DBMS client API version number.

Notes:

The higher word contains the major client version (the XX value in the XX.YY version number); the lower word contains the minor client version (the YY value in the XX.YY version number).

If an DBMS client was not set calling ClientVersion method will throw an exception.

Commit

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Saves any changes and ends the current transaction.

Notes:

Use Commit method to write transaction changes permanently to a database. It commits the work of all commands that associated with that connection.

All changes to the database since the last commit are made permanent and cannot be undone. Before a commit, all changes made since the start of the transaction can be rolled back using Rollback method.

Connect(DBString as string, UserID as string, Password as string, client as integer=0)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Opens the connection to a data source.

Example:

```
dim con as SqlConnectionMBS// your connection

// some calls for MS SQL Server:
con.Connect("srv2@pubs","","",SqlConnectionMBS.kSQLServerClient)
con.Connect("@pubs","","",SqlConnectionMBS.kSQLServerClient)
con.Connect("BEDLAM\SQL2005EX_EN@pubs","","",SqlConnectionMBS.kSQLServerClient)
con.Connect("BEDLAM\SQLEXPRESS@master","","",SqlConnectionMBS.kSQLServerClient)

// for MySQL:
con.Connect("192.168.1.80:3306@test","root","password",SqlConnectionMBS.kMySQLClient)

// for Postgre SQL:
con.Connect("somedb","name","password",SqlConnectionMBS.kPostgreSQLClient)

// for SQLite:
con.Connect("/test.db","","",SqlConnectionMBS.kSQLiteClient)
```

Notes:

- DBString: Name of database this connection will connect to (see Server specific notes).
- UserID: A string containing a user name to use when establishing the connection (see Server specific notes).
- Password: A string containing a password to use when establishing the connection.
- client: Optional. One of the following values from k*Client constants.

Using the Connect method on a SACConnection object establishes the physical connection to a data source.

After this method successfully completes, the connection is live and you can issue commands against it and process the results.

If you use the default value of Client parameter, you should set Client before using Connect.

To check whether a connection established use isConnected method. To check whether a connection is broken or not use isAlive method.

see also for server specific notes:

http://www.sqlapi.com/OnLineDoc/Connection_Connect.html

Disconnect

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Disconnects the connection from the database.

isAlive as boolean

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the database server connection status for a particular connection object.

Notes:

Returns true if the database server is active and accessible; otherwise false.

This method uses the safe query execution for most supported DBMS-es. The query uses the well known database table or procedure (mysql_ping is used for MySQL). If the query fails the method returns false.

isConnected as boolean

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the connection state for a particular connection object.

Notes: Returns true if connected; otherwise false.

IsolationLevel as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The transaction isolation level.

Notes:

Use the kReadCommitted, kReadUncommitted, kRepeatableRead, kSerializable and kLevelUnknown constants.

(Read and Write computed property)

NativeAPI as SQLAPIMBS

Plugin Version: 9.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns a set of functions of native DBMS client API.

Option(name as string) as string

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A string value of a specific connection or command option.

Notes:

see also:

http://www.sqlapi.com/OnLineDoc/Connection_ Option.html

(Read and Write computed property)

Rollback

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Cancels any changes made during the current transaction and ends the transaction.

Notes:

Rollback method rolls back the database to the state it was in at the completion of the last commit operation. All uncommitted work is undone.

Rollback method rolls back the work of all commands that associated with that connection.

To commit all changes made since the start of the transaction use Commit method.

ServerVersion as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Gets the currently connected DBMS server version number.

Notes: The higher word contains the major server version (the XX value in the XX.YY version number); the lower word contains the minor server version (the YY value in the XX.YY version number).

ServerVersionString as string

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Gets the currently connected DBMS server version string.

Notes: A server version string may contain some useful information about server brand, configuration and so on. It is a good idea to display this information in all your applications.

SetFileOption(name as string, file as folderitem)

Plugin Version: 10.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets an option with passing a file path.

Example:

```
dim db as new SQLConnectionMBS
```

```
// where is the library?
```

```
db.SetFileOption SQLConnectionMBS.kOptionLibraryMySQL, SpecialFolder.UserHome.Child("libmysqlclient.dylib")
```

Notes: Makes sure the path is correct and you have a 32bit library. 64 bit libraries will not work with Real Studio.

SQLExecute(command as string, CommandType as integer=0)

Plugin Version: 10.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Executes a SQL command and ignores result.

Notes:

This is a convenience function.

Internally it creates a SQLCommandMBS with the given command and calls Execute.

All text strings sent to the plugin must have a defined encoding. Else the internal text encoding conversions will fail.

SQLExecuteMT(command as string, CommandType as integer=0)

Plugin Version: 10.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Executes a SQL command and ignores result.

Notes:

This is a convenience function.

Internally it creates a SQLCommandMBS with the given command and calls Execute.

The work is performed on an extra thread, so this function can yield time to other Real Studio threads. And it calls the Working event regularly. For best user experience run this command on a Real Studio thread, so your GUI stays responsive.

All text strings sent to the plugin must have a defined encoding. Else the internal text encoding conversions will fail.

SQLSelect(command as string, CommandType as integer=0) as string

Plugin Version: 10.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Executes a SQL command and returns the first field's string value.

Notes:

This is a convenience function.

Internally it creates a SQLCommandMBS with the given command and calls Execute.

If the result is a record set, the first field from the first row is returned.

This is basically useful for commands like "select sqlite_ version()".

All text strings sent to the plugin must have a defined encoding. Else the internal text encoding conversions will fail.

SQLSelectMT(command as string, CommandType as integer=0) as string

Plugin Version: 10.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Executes a SQL command and returns the first field's string value.

Notes:

This is a convenience function.

Internally it creates a SQLCommandMBS with the given command and calls Execute.

If the result is a record set, the first field from the first row is returned.

This is basically useful for commands like "select sqlite_ version()".

The work is performed on an extra thread, so this function can yield time to other Real Studio threads. And it calls the Working event regularly. For best user experience run this command on a Real Studio thread, so your GUI stays responsive.

All text strings sent to the plugin must have a defined encoding. Else the internal text encoding conversions will fail.

2.16.2 Events

Working

Plugin Version: 10.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The event called while the SQLExecuteMT and SQLSelectMT methods are running.

2.16.3 Constants

kANSIlevel0 = 0

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the ANSI level constants.

kANSIlevel1 = 1

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the ANSI level constants.

kANSIlevel2 = 2

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the ANSI level constants.

kANSIlevel3 = 3

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the ANSI level constants.

kAutoCommitOff = 0

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the constants for the autocommit property.

Notes: Autocommit is off.

kAutoCommitOn = 1

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the constants for the autocommit property.

Notes: Autocommit is on.

kAutoCommitUnknown = -1

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the constants for the autocommit property.

Notes: Autocommit unknown

kClientNotSpecified = 0

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the database client constants.

Notes: Client is not specified.

kDB2Client = 6

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the database client constants.

Notes: DB2 client.

kFirebirdClient = 4

Plugin Version: 10.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the database client constants.

Notes: InterBase/Firebird client.

kInformixClient = 7

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the database client constants.

Notes: Informix client.

kInterBaseClient = 4

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the database client constants.

Notes: InterBase/Firebird client.

kLevelUnknown = -1

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the isolation level constants.

Notes: Unknown

kMySQLClient = 9

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the database client constants.

Notes: MySQL client.

kODBCClient = 1

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the database client constants.

Notes: ODBC client.

kOptionAPPNAME = "APPNAME"

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A constant for the options.

kOptionLibraryDB2 = "DB2CLI.LIBS"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the library with the SetFileOption method.

Notes: for DB2.

kOptionLibraryFirebird = "IBASE.LIBS"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the library with the SetFileOption method.

Notes: for Firebird.

kOptionLibraryInformix = "INFCLI.LIBS"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the library with the SetFileOption method.

Notes: for Informix.

kOptionLibraryInterbase = "IBASE.LIBS"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the library with the SetFileOption method.

Notes: for Interbase.

kOptionLibraryMySQL = "MYSQL.LIBS"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the library with the SetFileOption method.

Notes: for MySQL. Library extension on Mac is ".dylib", on Linux ".so" and on Windows ".dll". You get this library with the MySQL download on their homepage.

kOptionLibraryODBC = "ODBC.LIBS"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the library with the SetFileOption method.

Notes: for ODBC.

kOptionLibraryOracle = "OCI8.LIBS"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the library with the SetFileOption method.

Notes: for Oracle.

kOptionLibraryPostgreSQL = "LIBPQ.LIBS"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the library with the SetFileOption method.

Notes: for PostgreSQL.

kOptionLibrarySeparator = ":"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the platform specific path separator.

Notes:

Use with SetFileOption to specify multiple file paths for a library.
Has a different value on the different platforms.

kOptionLibrarySQLBase = "SQLBASE.LIBS"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the library with the SetFileOption method.

Notes: for SQLbase.

kOptionLibrarySQLite = "SQLITE.LIBS"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the library with the SetFileOption method.

Notes: for SQLite. Can also be the spatialite library.

kOptionLibrarySybaseComm = "SYBCOMN.LIBS"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the library with the SetFileOption method.

Notes: for Sybase.

kOptionLibrarySybaseCS = "SYBCS.LIBS"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the library with the SetFileOption method.

Notes: for Sybase.

kOptionLibrarySybaseCT = "SYBCT.LIBS"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the library with the SetFileOption method.

Notes: for Sybase.

kOptionLibrarySybaseIntl = "SYBINTL.LIBS"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the library with the SetFileOption method.

Notes: for Sybase.

kOptionLibrarySybaseTCL = "SYBTCL.LIBS"

Plugin Version: 10.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constant to specify the library with the SetFileOption method.

Notes: for Sybase.

kOptionWSID = "WSID"

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A constant for the options.

kOracleClient = 2

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the database client constants.

Notes:

Oracle client.

For Windows the file is "oci.dll", for Linux libclntsh.so and for Mac OS X libclntsh.dylib.

kPostgreSQLClient = 10

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the database client constants.

Notes: PostgreSQL client.

kReadCommitted = 1

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the isolation level constants.

Notes: Read committed.

kReadUncommitted = 0

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the isolation level constants.

Notes: Read uncommitted.

kRepeatableRead = 2

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the isolation level constants.

Notes: Repeatable read.

kSerializable = 3

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the isolation level constants.

Notes: Serializable.

kSQLBaseClient = 5

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the database client constants.

Notes: SQLbase client.

kSQLiteClient = 11

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the database client constants.

Notes: SQLite client. Or spatialite.

kSQLServerClient = 3

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the database client constants.

Notes:

Mircosoft SQL Server client.

You may need to download the client packages for accessing SQL Server. Files like the SQLNCLI.dll may be missing. You can download for example the Feature Pack for Microsoft SQL Server 2005 from the microsoft download page.

kSybaseClient = 8

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the database client constants.

Notes: Sybase client.

2.17 class SQLBLobMBS

class SQLBLobMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class for a blob.
Notes:

Basically this is a `SQLStringMBS` which is always marked to contain binary data. You only need this class to use the constructor with `dataprovder` to stream data to the database.

Subclass of the `SQLLongOrLobMBS` class.

2.17.1 Methods

Constructor(data as SQLStringMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new blob object from a string object.

See also:

- 2.17.1 Constructor(dataProvider as SQLDataProviderMBS, BlockSize as UInt32) 81

Constructor(dataProvider as SQLDataProviderMBS, BlockSize as UInt32)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new blob object from a data provider.

Notes:

The blocksize specifies in which sizes data is requested from the provider.

You must make sure that the data provider and this new blob object life long enough. Because the actual data is requested later when you do the update on the database.

If BlockSize is 0, the default block size is used.

See also:

- 2.17.1 Constructor(data as SQLStringMBS) 81

2.18 class SQLAPIMBS

class SQLAPIMBS

Plugin Version: 9.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** This is a class for the native APIs.

Notes: The plugin does not implem

2.18.1 Properties

Connection as `SQLConnectionMBS`

Plugin Version: 9.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The database connection this API is used with.

Notes: (Read only property)

2.19 class `SQLCLOBMBS`

class `SQLCLOBMBS`

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class for a clob (character large object).

Notes:

Basically this is a `SQLStringMBS` which is always marked to contain text. You only need this class to use the constructor with `dataProvider` to stream data to the database.

Subclass of the `SQLLongOrLobMBS` class.

2.19.1 Methods

Constructor(`data` as `SQLStringMBS`)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new clob object from a string object.

See also:

- 2.19.1 Constructor(`dataProvider` as `SQLDataProviderMBS`, `BlockSize` as `UInt32`)

82

Constructor(`dataProvider` as `SQLDataProviderMBS`, `BlockSize` as `UInt32`)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new clob object from a data provider.

Notes:

The blocksize specifies in which sizes data is requested from the provider. You must make sure that the data provider and this new clob object life long enough. Because the actual data is requested later when you do the update on the database.

If BlockSize is 0, the default block size is used.
See also:

- 2.19.1 Constructor(data as SQLStringMBS)

82

2.20 class SQLBytesMBS

class SQLBytesMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class for a string of bytes.

Notes: Subclass of the SQLStringMBS class.

2.20.1 Methods

Constructor(data as SQLStringMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new bytes object based on the given string object.

2.21 class PostgreSQLAPIMBS

class PostgreSQLAPIMBS

Plugin Version: 9.8 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class for Postgre SQL specific functions.

Notes: Subclass of the SQLAPIMBS class.

2.21.1 Methods

DB(conn as SQLConnectionMBS) as string

Plugin Version: 9.8 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The database name used to create the connection.

ErrorMessage(conn as SQLConnectionMBS) as string

Plugin Version: 9.8 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The last error message.

Field(cmd as SQLCommandMBS, RecordIndex as integer, FieldIndex as integer) as string

Plugin Version: 9.8 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Queries a field by index for the row with the RecordIndex.

See also:

- 2.21.1 Field(cmd as SQLCommandMBS, RecordIndex as integer, FieldName as string) as string 84

Field(cmd as SQLCommandMBS, RecordIndex as integer, FieldName as string) as string

Plugin Version: 9.8 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Queries a field by name for the row with the RecordIndex.

See also:

- 2.21.1 Field(cmd as SQLCommandMBS, RecordIndex as integer, FieldIndex as integer) as string 84

FieldCount(cmd as SQLCommandMBS) as integer

Plugin Version: 9.8 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The number of fields in the result.

Host(conn as SqlConnectionMBS) as string

Plugin Version: 9.8 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The host used to create the connection.

Options(conn as SqlConnectionMBS) as string

Plugin Version: 9.8 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The options used to create the connection.

Password(conn as SqlConnectionMBS) as string

Plugin Version: 9.8 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The password used to create the connection.

Port(conn as SqlConnectionMBS) as string

Plugin Version: 9.8 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The port used to create the connection.

RecordCount(cmd as SqlCommandMBS) as integer

Plugin Version: 9.8 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The number of records in the result.

TTY(conn as SqlConnectionMBS) as string

Plugin Version: 9.8 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The tty used to create the connection.

User(conn as SQLConnectionMBS) as string

Plugin Version: 9.8 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The user name used to create the connection.

2.22 class SQLCommandMBS**class SQLCommandMBS**

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** This is the central class for the using the SQL database access.

Example:

```

dim con as SQLConnectionMBS
dim cmd as SQLCommandMBS

try

con = new SQLConnectionMBS // connection object
cmd = new SQLCommandMBS // create command object

// where is the library?
con.SetFileOption con.kOptionLibraryMySQL, SpecialFolder.UserHome.Child("libmysqlclient.dylib")

// connect to database (mySQL in our example)
// server: 192.168.1.80
// port: 3306
// database: test
// name: root
// no password
con.Connect("192.168.1.80:3306@test","root","",",SQLConnectionMBS.kMySQLClient)
// associate a command with connection
// connection can also be specified in SACommand constructor
cmd.Connection=con

// create table
cmd.setCommandText("Create table test_ tbl(fid integer, fvarchar20 varchar(20), fblob blob)")
cmd.Execute

// insert value
cmd.setCommandText("Insert into test_ tbl(fid, fvarchar20) values (1, 'Some string (1)')")
cmd.Execute

```

```
// commit changes on success
con.Commit

MsgBox("Table created, row inserted!")

catch r as SQLExceptionMBS
// SACConnection::Rollback()
// can also throw an exception
// (if a network error for example),
// we will be ready
try

// on error rollback changes
if con<>nil then
con.rollback
end if
catch x as SQLExceptionMBS
// ignore
end try

// show error message
MsgBox r.message
end try
```

2.22.1 Methods

Cancel

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Attempts to cancel the pending result set, or current statement execution.

Notes:

Cancel can cancel the following types of processing on a statement:

A function running asynchronously on the statement.

A function running on the statement on another thread.

After an application calls a function asynchronously, it checks repeatedly to determine whether it has finished processing. While the function is processing, an application can call Cancel to cancel the function.

In a multithread application, the application can cancel a function that is running synchronously on a statement.

see also

http://www.sqlapi.com/OnLineDoc/Command_Cancel.html

Close

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Closes the specified command object.

Notes:

Use the Close method to close the command explicitly.

A command will be implicitly closed in destructor, so you don't have to call Close method explicitly.

CommandText as string

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Gets the command text associated with the SACommand object.

Example:

```
dim s as new SQLCommandMBS(nil, "select * from test")
```

```
MsgBox s.CommandText
```

Notes:

Use the CommandText method to return the command text declared in SACommand constructor or set-CommandText method.

All text strings sent to the plugin must have a defined encoding. Else the internal text encoding conversions will fail.

CommandType as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Gets the command type currently associated with the SACommand object.

Notes:

One of the following values from SACommandType_ t enum:

* kCommandTypeUnknown Command type is not defined. Library will detect command type automatically when needed.

kCommandTypeSQLStmt Command is an SQL statement.

kCommandTypeSQLStmtRaw Command is an SQL statement that mustn't be interpreted by SQLAPI++.

kCommandTypeStoredProc Command is a stored procedure or a function.

Remarks

The command type can be explicitly set in SACommand constructor and setCommandText method, but it's not necessary to do it.

The CommandType method returns the command type value that was specified in SACommand constructor or setCommandText method. If you declared the command type value as kCommandTypeUnknown (the default value) then command type is detected by the Library and the CommandType method returns this detected value.

Connection as SQLConnectionMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The connection for the command.

Notes:

When you set the connection on a command object that already has associated connection, the previous association will be correctly discarded (with closing opened command if needed) and new connection will be set.

If you attempt to call any method on a SACommand object that requires database access with no valid connection, an error occurs.

(Read and Write computed property)

Constructor

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new command object with no connection and no command text.

See also:

- 2.22.1 Constructor(connection as SqlConnectionMBS, SqlCommand as String, CommandType as integer=0) 90

Constructor(connection as SqlConnectionMBS, SqlCommand as String, CommandType as integer=0)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** This constructor initializes a new SqlCommandMBS object.

Notes:

Connection: the connection to associated with the command.

SqlCommand: A string which represents command text string (an SQL statement or a stored procedure name). If it is an empty string, no command text is associated with the command, and you have to call setCommandText method later.

CommandType: The type of command like kCommandTypeUnknown, kCommandTypeSQLStatement, kCommandTypeSQLStatementRaw or kCommandTypeStoredProcedure.

All text strings sent to the plugin must have a defined encoding. Else the internal text encoding conversions will fail.

See also:

- 2.22.1 Constructor 90

CreateParam(name as string, ParamType as integer, DirType as integer=0) as SqlParameterMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates parameter associated with the specified command.

Notes:

Parameters

Returns a new SqlParameterMBS object on success or nil on any error.

name: A string representing the name of parameter.
 ParamType: Type of the parameter's value. Use the kDataType constants.
 ParamSize: An integer value represents parameter's value size.
 ParamPrecision: An integer value represents parameter's value precision.
 ParamScale: An integer value represents parameter's value scale.
 DirType: Type of the parameter. Use the kParamDirType* constants.

Normally you should not create parameters by yourself. The Library automatically detects whether the command has parameters in terms of the command text and implicitly creates a set of SAParam objects.

Nevertheless, if you call CreateParam explicitly you have to delete all SAParam objects created automatically by the Library before. Use DestroyParams method before the first call of CreateParam method.

See also:

- 2.22.1 CreateParam(name as string, ParamType as integer, NativeType as integer, ParamSize as integer, ParamPrecision as integer, ParamScale as integer, DirType as integer=0) as SQLParamMBS

CreateParam(name as string, ParamType as integer, NativeType as integer, ParamSize as integer, ParamPrecision as integer, ParamScale as integer, DirType as integer=0) as SQLParamMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates parameter associated with the specified command.

Notes:

Parameters

name: A string representing the name of parameter.
 ParamType: Type of the parameter's value. Use the kDataType constants.
 ParamSize: An integer value represents parameter's value size.
 ParamPrecision: An integer value represents parameter's value precision.
 ParamScale: An integer value represents parameter's value scale.
 DirType: Type of the parameter. Use the kParamDirType* constants.

Returns a new SQLParamMBS object on success or nil on any error.

Normally you should not create parameters by yourself. The Library automatically detects whether the command has parameters in terms of the command text and implicitly creates a set of SAParam objects.

Nevertheless, if you call `CreateParam` explicitly you have to delete all `SAParam` objects created automatically by the Library before. Use `DestroyParams` method before the first call of `CreateParam` method.

See also:

- 2.22.1 `CreateParam(name as string, ParamType as integer, DirType as integer=0)` as `SQLParamMBS`
90

DestroyParams

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Destroys all parameters associated with the specified command.

Notes:

`DestroyParams` method destroys all parameters either created automatically by the Library or by user.

Normally you should not create and delete parameters by yourself. The Library automatically detects whether the command has parameters, implicitly creates a set of `SAParam` objects and then deletes them in `SACommanddestructor`. But if you have some reason to create parameters explicitly use `CreateParam` method and then call `DestroyParams` method to delete all parameters after your work with parameters is over.

Execute

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Executes the current command.

Notes:

Use the `Execute` method to execute the query or stored procedure specified in the command text. `Execute` method calls `Prepare` method implicitly if needed. If the command has input parameters, they should be bound before calling `Execute` method. Input parameters represented by `SAParam` object. To bind input variables assign a value to `SAParam` object returning by `Param` or `ParamByIndex` methods.

A command (an SQL statement or procedure) can have a result set after executing. To check whether a result set exists use `isResultSet` method. If result set exists, a set of `SAField` objects is created after command execution. Rows from the result set can be fetched one by one using `FetchNext` method. To get field description or value use `Field` method.

Output parameters represented by `SAParam` objects. They are available after command execution. To get parameter description or value use `Param` or `ParamByIndex` methods.

ExecuteCommand(SQLCommand as string, CommandType as integer=0)

Plugin Version: 10.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Executes the given command.

Notes:

This is a convenience function.

Internally it calls setCommandText with the given command and calls Execute.

All text strings sent to the plugin must have a defined encoding. Else the internal text encoding conversions will fail.

ExecuteCommandMT(SQLCommand as string, CommandType as integer=0)

Plugin Version: 10.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Executes the given command.

Notes:

This is a convenience function.

Internally it calls setCommandText with the given command and calls Execute.

The work is performed on an extra thread, so this function can yield time to other Real Studio threads. And it calls the Working event regularly. For best user experience run this command on a Real Studio thread, so your GUI stays responsive.

All text strings sent to the plugin must have a defined encoding. Else the internal text encoding conversions will fail.

ExecuteMT

Plugin Version: 10.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Executes the current command.

Notes:

Use the Execute method to execute the query or stored procedure specified in the command text. Execute method calls Prepare method implicitly if needed. If the command has input parameters, they should be bound before calling Execute method. Input parameters represented by SAParam object. To bind input variables assign a value to SAParam object returning by Param or ParamByIndex methods.

A command (an SQL statement or procedure) can have a result set after executing. To check whether a result set exists use isResultSet method. If result set exists, a set of SAField objects is created after command execution. Rows from the result set can be fetched one by one using FetchNext method. To get field description or value use Field method.

Output parameters represented by SAParam objects. They are available after command execution. To get parameter description or value use Param or ParamByIndex methods.

The work is performed on an extra thread, so this function can yield time to other Real Studio threads. And it calls the Working event regularly. For best user experience run this command on a Real Studio thread, so your GUI stays responsive.

FetchFirst as boolean

Plugin Version: 11.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Fetches first row from a result set.

Notes:

Same as FetchNext, but jumps to the first row.

Not supported for Interbase and SQLite.

FetchLast as boolean

Plugin Version: 11.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Fetches last row from a result set.

Notes:

Same as FetchNext, but jumps to the last row.

Not supported for Interbase and SQLite.

FetchNext as boolean

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Fetches next row from a result set.

Notes:

Returns true if the next row was fetched; otherwise false .

Use FetchNext method to fetch row by row from the result set.

Each column of fetched row is represented by SAField object. If a result set exists after the last command execution, a set of SAField objects is created implicitly. To check whether a result set exists use isResultSet method. FetchNext method updates value parts of SAField objects.

To get field description or value use Field method.

FetchPrior as boolean

Plugin Version: 11.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Fetches previous row from a result set.

Notes:

Same as FetchNext, just going back inside the result set.

Not supported for Interbase and SQLite.

Field(index as integer) as SQLFieldMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the column specified by its position in the result set.

Notes:

index: A one-based field number in a result set.

Use Field method to access a field by its name or position in the result set.

Using an index smaller than 1 and greater then the value returned by FieldCount method will result in a failed assertion.

A set of SAField objects creates implicitly after the command execution if the result set exists.SAField object contains full information about a column: name, type, size, value.

See also:

- 2.22.1 Field(name as string) as SQLFieldMBS 96

Field(name as string) as SQLFieldMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the column specified by its name in the result set.

Notes:

name: A string that represents a name of the requested field.

Returns a reference to a SAField object.

Use Field method to access a field by its name or position in the result set.

Using a non-existent field name will throw an exception.

A set of SAField objects creates implicitly after the command execution if the result set exists.SAField object contains full information about a column: name, type, size, value.

See also:

- 2.22.1 Field(index as integer) as SQLFieldMBS 95

FieldCount as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the number of fields (columns) in a result set.

Notes:

FieldCount method returns the number of fields created implicitly after the command execution if a result set exists.

A field is represented by SAField object. You can get field value and description using Field method.

isExecuted as boolean

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Whether this command was already executed.

isOpened as boolean

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns true if the SACommand object is opened; otherwise false.

isResultSet as boolean

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Tests whether a result set exists after the command execution.

Notes: Returns true if the result set exists; otherwise false.

Open

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Opens the specified command object.

Notes:

Use the `Open` method to open the command explicitly.

A command will be implicitly opened by any method that needs an open command, therefore you don't have to call it explicitly.

To test whether a command is opened use `isOpened` method.

Option(name as string) as string

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A string value of a specific command option.

Notes:

see also:

http://www.sqlapi.com/OnLineDoc/Command_Option.html

(Read and Write computed property)

Param(ID as integer) as SQLParamMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the command parameter specified by its position.

Notes:

ID: A position of parameter specified in the command text. Normally position is a number stated in the command text after a colon (for example, 1 for :1, 5 for :5).

Returns a reference to a `SAParam` object which is only valid as long as the `param` object is not deleted by the library.

Use `Param` method to access a parameter by its name or position (in SQL statement). If, for example, you want to walk through all the parameters use `ParamByIndex` method.

If parameters were not created before calling `Param` method the Library creates them implicitly (can query native API if needed and therefore can throw exception on error) and then returns the specified parameter.

Passing a value of name or position which does not specified in the command text will throw an exception.

SAParam object contains full information about a parameter: name, type, size, etc. Values for the input parameters can be assigned to SAParam object.

See also:

- 2.22.1 Param(name as string) as SQLParamMBS

99

Param(name as string) as SQLParamMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the command parameter specified by its name.

Notes:

Name: A string that represents a name of the requested parameter. Normally name is a string stated in the command text after a colon (for example, 'city' for :city, 'my city' for :”my city”) or a parameter name in a stored procedure or function.

Returns a reference to a SAParam object which is only valid as long as the param object is not deleted by the library.

Use Param method to access a parameter by its name or position (in SQL statement). If, for example, you want to walk through all the parameters use ParamByIndex method.

If parameters were not created before calling Param method the Library creates them implicitly (can query native API if needed and therefore can throw exception on error) and then returns the specified parameter.

Passing a value of name or position which does not specified in the command text will throw an exception.

SAParam object contains full information about a parameter: name, type, size, etc. Values for the input parameters can be assigned to SAParam object.

See also:

- 2.22.1 Param(ID as integer) as SQLParamMBS

98

ParamByIndex(index as integer) as SQLParamMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the command parameter specified by index.

Notes:

Index: A zero-based index of the requested parameter in the array of SAParam objects. It must be greater than or equal to 0 and 1 less than the value returned by ParamCount method.

Returns a reference to a SAParam object.

Normally you should use Param method to access a parameter by its name or position (in SQL statement). ParamByIndex method can be used if, for example, you want to walk through all the parameters.

If parameters were not created before calling ParamByIndex method the Library creates them implicitly (can query native API if needed and therefore can throw exception on error) and then returns the specified parameter.

Passing a negative value of index or a value greater or equal than the value returned by ParamCount method will result in a failed assertion.

SAParam object contains full information about a parameter: name, type, size, etc. Values for the input parameters can be assigned to SAParam object.

ParamCount as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the number of parameters associated with the SACommand object.

Notes:

ParamCount method returns the number of parameters created explicitly by using CreateParam method or (if parameters were not created before) creates them implicitly (can query native API if needed and therefore can throw exception on error) and returns the number of created parameters.

Command parameter is represented by SAParam object. You can look SAParam objects through and assign their values with Param and ParamByIndex methods.

Prepare

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Prepares command before execution.

Notes:

Prepare method compiles the command, but does not execute it. The method detects syntax errors in command text and verifies the existence of database objects.

Execute method calls Prepare method implicitly if needed, therefore you don't have to call it explicitly.

RowsAffected as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the number of rows affected by the last insert/update/delete command execution.

setCommandText(SQLCommand as string, CommandType as integer=0)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets the command text.

Example:

```
dim s as new SQLCommandMBS
s.setCommandText "select * from test"
MsgBox s.CommandText
```

Notes:

SQLCommand: A string which represents command text string (an SQL statement or a stored procedure name).

CommandType: The type of command like kCommandTypeUnknown, kCommandTypeSQLStatement,

`kCommandTypeSQLStatementRaw` or `kCommandTypeStoredProcedure`.

It's not necessary to set a command type explicitly, because it is defined automatically in terms of command text string. But if you still have any reason to do it, use one of the `kCommandType*` constants. To get command type use `CommandType` method.

2.22.2 Events

Working

Plugin Version: 10.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The event called while the `ExecuteMT` and `ExecuteCommandMT` methods are running.

2.22.3 Constants

`kCommandTypeSQLStatement=1`

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the constants for the command type.

Notes: Command is an SQL statement.

`kCommandTypeSQLStatementRaw=2`

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the constants for the command type.

Notes: Command is an SQL statement that mustn't be interpreted by SQLAPI.

`kCommandTypeStoredProcedure=3`

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the constants for the command type.

Notes: Command is a stored procedure or a function.

kCommandTypeUnknown=0

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the constants for the command type.

Notes: Used by default. Library detects command type automatically.

kOptionPreFetchRows="PreFetchRows"

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the option constants.

Example:

```
dim cmd as new SQLCommandMBS
// do something

dim nBulkSize as integer = 1000
cmd.Option("PreFetchRows") = str(nBulkSize)
```

kParamDirTypeInput=0

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the parameter direction type constants.

Notes: Input parameter.

kParamDirTypeInputOutput=1

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the parameter direction type constants.

Notes: Input/output parameter.

kParamDirTypeOutput=2

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the parameter direction type constants.

Notes: Output parameter.

kParamDirTypeReturn=3

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the parameter direction type constants.

Notes: Returning parameter.

2.23 class SQLDateTimeMBS

class SQLDateTimeMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The SQL date/time value class.

Example:

```
dim d as new SQLDateTimeMBS(2008, 3, 4, 23, 10, 20)
```

```
MsgBox d.StringValue // shows "2008-03-04T23:10:20"
```

2.23.1 Methods

Constructor(other as SQLDateTimeMBS)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a copy of the SQL Date Time.

Example:

```
dim d as new SQLDateTimeMBS(2008, 3, 4, 23, 10, 20)
```

```
dim e as new SQLDateTimeMBS(d)
```

```
MsgBox e.StringValue // shows "2008-03-04T23:10:20"
```

See also:

- 2.23.1 Constructor(value as double) 105
- 2.23.1 Constructor(Year as integer, Month as integer, Day as integer, Hour as integer, Minute as integer, Second as integer) 105

Constructor(value as double)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new SQL date time value based on the double value.

Example:

```
dim d as new SQLDateTimeMBS(2008, 3, 4, 23, 10, 20)
dim e as new SQLDateTimeMBS(d.DoubleValue+1) // clone with one day more

MsgBox e.StringValue // shows "2008-03-05T23:10:20"
```

See also:

- 2.23.1 Constructor(other as SQLDateTimeMBS) 104
- 2.23.1 Constructor(Year as integer, Month as integer, Day as integer, Hour as integer, Minute as integer, Second as integer) 105

Constructor(Year as integer, Month as integer, Day as integer, Hour as integer, Minute as integer, Second as integer)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new SQL Datetime with the given values.

Example:

```
dim d as new SQLDateTimeMBS(2008, 3, 4, 23, 10, 20)

MsgBox d.StringValue // shows "2008-03-04T23:10:20"
```

See also:

- 2.23.1 Constructor(other as SQLDateTimeMBS) 104
- 2.23.1 Constructor(value as double) 105

DoubleValue as double

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The double value of this date/time.

Notes:

Use these operators to get current date/time value using standard double representation. Days are represented by whole number increments starting with 30 December 1899, midnight as time zero. Hour values are expressed as the absolute value of the fractional part of the number.

Date and time	Representation
30 December 1899, midnight	0.00
1 January 1900, midnight	2.00
4 January 1900, midnight	5.00
4 January 1900, 6 A.M.	5.25
4 January 1900, noon	5.50
4 January 1900, 9 P.M.	5.875

Fraction as UInt32

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the value of the fraction of second (0 to 999,999,999) this SQLDateTime object represents.

Notes:

The value of the fraction field is the number of billionths of a second and ranges from 0 through 999,999,999 (1 less than 1 billion). For example, the value of the fraction field for a half-second is 500,000,000, for a thousandth of a second (one millisecond) is 1,000,000, for a millionth of a second (one microsecond) is 1,000, and for a billionth of a second (one nanosecond) is 1.

GetDay as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the day this SADateTime object represents (1 31).

GetDayOfWeek as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the day of the week this SDateTime object represents (Sunday = 1).

GetDayOfYear as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the day of the year this SDateTime object represents (Jan 1 = 1).

GetHour as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the hour this SDateTime object represents (0 23).

GetMinute as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the minute this SDateTime object represents (0 59).

GetMonth as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the month this SDateTime object represents (1 12).

GetSecond as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the second this SDateTime object represents (0 59).

GetYear as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the year this SADateTime object represents.

StringValue as string

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The string value for this date/time.

2.24 class SQLDataProviderMBS

class SQLDataProviderMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class for a data provider.

Notes: Use this to set a blob/clob object with streaming data. For example if you want to add a 1 GB big file to the database without loading it into RAM in one piece, you can use this class to read it in small chunks.

2.24.1 Events

Read(byref PieceType as integer, Length as UInt32) as string

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The event called whenever new data is needed.

Notes:

PieceType is kOnePiece, kFirstPiece, kLastPiece or kNextPiece.

If your stream is on the end, you set this to kLastPiece.

Return the raw data in a string.

Length is the number of bytes.

2.24.2 Constants

kFirstPiece=1

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the piece type constants.

Notes: The first piece is processed. You may setup everything you need to handle the data.

kLastPiece=3

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the piece type constants.

Notes: The last piece is processed. You can close files/network connections.

kNextPiece=2

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the piece type constants.

Notes: The next piece is processed. Not the first one or the last one, but one between.

kOnePiece=4

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the piece type constants.

Notes: The whole data stream is delivered in one call of the event.

2.25 class SQLGlobalsMBS

class SQLGlobalsMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class for the global methods of the SQL plugin.

2.25.1 Methods

GetVersionBuild as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The build number of the SQLAPI library.

GetVersionMajor as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The major version number of the SQLAPI library.

GetVersionMinor as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The minor version number of the SQLAPI library.

SetLicenseCode(n as string, enddate as integer, v1 as integer, v2 as integer)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Registers the SQL plugin and library.

Notes: Once you ordered a license, you receive details on how to call this method.

Setlocale(category as integer, locale as string)

Plugin Version: 10.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets the locale to use.

Notes:

The Setlocale function sets the C library's notion of natural language formatting style for particular sets of routines. Each such style is called a 'locale' and is invoked using an appropriate name passed as a C string.

The setlocale() function recognizes several categories of routines. These are the categories and the sets of

routines they select:

LocaleAll	Set the entire locale generically.
LocaleCollate	Set a locale for string collation routines. This controls alphabetic ordering in <code>strcoll()</code> and <code>strxfrm()</code> .
LocaleCTYPE	Set a locale for the <code>ctype</code> and multibyte functions. This controls recognition of upper and lower case, alphabetic or non-alphabetic characters, and so on.
LocaleMessages	Set a locale for message catalogs, see <code>catopen</code> function.
LocaleMonetary	Set a locale for formatting monetary values; this affects the <code>localeconv()</code> function.
LocaleNumeric	Set a locale for formatting numbers. This controls the formatting of decimal points in input and output of floating point numbers in functions such as <code>printf()</code> and <code>scanf()</code> , as well as values returned by <code>localeconv()</code> .
LocaleTime	Set a locale for formatting dates and times using the <code>strftime()</code> function.

Only three locales are defined by default: the empty string "" (which denotes the native environment) and the "C" and "POSIX" locales (which denote the C-language environment). By default, C programs start in the "C" locale.

2.25.2 Constants

LocaleAll=0

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the locale category constants for `SetLocale`.

Notes: Set the entire locale generically.

LocaleCollate=1

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the locale category constants for `SetLocale`.

Notes: Set a locale for string collation routines. This controls alphabetic ordering in `strcoll()` and `strxfrm()`.

LocaleCType=2

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the locale category constants for SetLocale.

Notes: et a locale for the ctype(3) and multibyte(3) functions. This controls recognition of upper and lower case, alphabetic or non-alphabetic characters, and so on.

LocaleMessages=6

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the locale category constants for SetLocale.

Notes: Set a locale for message catalogs, see catopen(3) function.

LocaleMonetary=3

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the locale category constants for SetLocale.

Notes: Set a locale for formatting monetary values; this affects the localeconv() function.

LocaleNumeric=4

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the locale category constants for SetLocale.

Notes: Set a locale for formatting numbers. This controls the formatting of decimal points in input and output of floating point numbers in functions such as printf() and scanf(), as well as values returned by localeconv().

LocaleTime=5

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the locale category constants for SetLocale.

Notes: Set a locale for formatting dates and times using the strftime() function.

2.26 class `SQLExceptionMBS`

class `SQLExceptionMBS`

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The error exception class to report SQL errors.

Notes:

The `SQLDatabaseMBS` class sets its error properties on an error. All other SQL classes raise exceptions where you can check the message property.
Subclass of the `RuntimeException` class.

2.27 class `SQLFieldMBS`

class `SQLFieldMBS`

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** This is the class for a SQL field in a record.

Notes:

Be aware that field objects exists only as long as their `SQLCommand` exists.
Subclass of the `SQLValueReadMBS` class.

2.27.1 Methods

FieldNativeType as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns native type code of the field.

FieldPrecision as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns precision of the field value (the total number of allowable digits).

FieldScale as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns scale of the field value (the number of digits to the right of the decimal point).

FieldSize as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns field data size.

Notes: (Read and Write computed property)

FieldType as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns field data type.

Notes:

Value is one of the kDataType* constants.
(Read and Write computed property)

isFieldRequired as boolean

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Shows if it is possible for the field value to be null.

Notes: Returns true if the field value can be null; false otherwise.

Name as string

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns name of the field.

Option(name as string) as string

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The string value of a specific field option.

Notes:

See for more details:

http://www.sqlapi.com/OnLineDoc/Field_setOption.html

(Read and Write computed property)

Pos as integer

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns a one-based position of the field in a result set.

ReadLongOrLob(consumer as *SQLDataConsumerMBS*, BlockSize as integer)

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Starts reading of Long or BLOB(CLOB) value using the given data consumer.

Notes:

BlockSize: Size of piece of data you want to get to the consumer event.

After a command execution all output parameters are updated by their values, including Long and BLOB(CLOB) parameters. If you want to control piecewise reading of Long or BLOB(CLOB) data you should do the following:

Before a command execution set `kLongOrLobReaderManual` reading mode (see `LongOrLobReaderMode`) for Long or BLOB(CLOB) parameters you want to process by a data consumer. After that `SQLAPI++` will skip reading output Long and BLOB(CLOB) parameters that you set to be read manually.

After command execution use `ReadLongOrLob` method for each output parameter defined to be read manually.

Note, that if the command has result set(s) (it is possible in some servers, see Server specific notes) then output parameters are available only after all result sets are completely processed using `FetchNext` method.

2.28 class SQLDataConsumerMBS

class SQLDataConsumerMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The class for a data consumer.

Notes: If you query a clob/blob value, that value may not fit into memory, so you may prefer to get a callback for data and write it to a file in small chunks.

2.28.1 Events

Write(PieceType as integer, data as string, Length as UInt32, BlobSize as UInt32)

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The event called to process data.

Notes:

PieceType is kOnePiece, kFirstPiece, kLastPiece or kNextPiece.

Data is the raw data in a binary string.

Length is the number of bytes and BlobSize the size of data blocks used.

2.28.2 Constants

kFirstPiece=1

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the piece type constants.

Notes: The first piece is processed. You may setup everything you need to handle the data.

kLastPiece=3

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the piece type constants.

Notes: The last piece is processed. You can close files/network connections.

kNextPiece=2

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the piece type constants.

Notes: The next piece is processed. Not the first one or the last one, but one between.

kOnePiece=4

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** One of the piece type constants.

Notes: The whole data stream is delivered in one call of the event.

2.29 class SQLiteDatabaseMBS

class SQLiteDatabaseMBS

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The database class for the SQL plugin

Notes:

You can use the SQL plugin without using REALbasic built in database classes if you use the SQLConnectionMBS and SQLCommandMBS classes.

Or you use the SQLiteDatabaseMBS class which is a subclass of the database class and can be used with REALbasics RecordSet class. The current implementation is far from complete. You can connect with passing the database URL in the DatabaseName property of the SQLiteDatabaseMBS class. You prefix this URL with the database type you are using.

You can use Execute and Select to run SQL statements. Errors can be queries with the lasterror properties. For the RecordSet, you can get the column count, the column names and values and move to the next row. All the other methods like deleting a record or updating a value are not yet implemented and you may use SQL commands to do this.

Supported databases: Oracle, Microsoft SQL Server, DB2, Sybase, Informix, InterBase/Firebird, SQLBase, MySQL, PostgreSQL and ODBC and SQLite.

As field and table schema functions are not implemented, you can't use this database with the database browser features in the Real Studio IDE.

Subclass of the Database class.

2.29.1 Methods

Connect as boolean

Plugin Version: 9.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Connects to the database.

Example:

```
dim db as new SQLDatabaseMBS

// where is the library?
db.SetFileOption SQLConnectionMBS.kOptionLibraryMySQL, SpecialFolder.UserHome.Child("libmysqlclient.dylib")

// connect to database
// in this example it is MySQL,
// but can also be Sybase, Informix, Oracle, DB2
// SQLServer, InterBase, SQLBase and ODBC

db.DatabaseName="mysql:192.168.1.80:3306@test"
db.UserName="root"
db.Password=""

if db.Connect then

MsgBox "We are connected!"

MsgBox "Server Version: "+db.GetConnection.ServerVersionString

// Disconnect is optional
// autodisconnect will occur in destructor if needed

else
MsgBox db.ErrorMessage
end if
```

Notes:

Returns true on success and false on failure.

Please set the DatabaseName, UserName and Password properties.

The database name must contain the complete information and a prefix for the kind of database.

Use this prefixes: "ODBC:" , "Oracle:" , "SQLServer:" , "InterBase:" "SQLBase:" , "DB2:" "Informix:" , "Sybase:" , "MySQL:" , "PostgreSQL:" or "SQLite:".

GetConnection as SqlConnectionMBS

Plugin Version: 9.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The connection for this database used in the background.

Notes: Note that methods on this connection object can raise exceptions while methods on the SQL-DatabaseMBS class sets the error properties.

Option(name as string) as string

Plugin Version: 10.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets an option for the connection.

Notes: (Read and Write computed property)

SetFileOption(name as string, file as folderitem)

Plugin Version: 10.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Sets an option with passing a file path.

Example:

```
dim db as new SQLDatabaseMBS
```

```
// where is the library?
```

```
db.SetFileOption SqlConnectionMBS.kOptionLibraryMySQL, SpecialFolder.UserHome.Child("libmysqlclient.dylib")
```

Notes: Makes sure the path is correct and you have a 32bit library. 64 bit libraries will not work with Real Studio.

Chapter 3

List of all classes

• PostgreSQLAPIMBS	83
• SQLAPIMBS	81
• SQLBLobMBS	80
• SQLBytesMBS	83
• SQLCLobMBS	82
• SQLCommandMBS	86
• SQLConnectionMBS	63
• SQLDatabaseMBS	117
• SQLDataConsumerMBS	116
• SQLDataProviderMBS	108
• SQLDateTimeMBS	104
• SQLErrorExceptionMBS	113
• SQLFieldMBS	113
• SQLGlobalsMBS	109
• SQLIntervalMBS	29
• SQLite3MBS	33
• SQLLongBinaryMBS	36
• SQLLongCharMBS	37

• SQLLongOrLobMBS	39
• SQLNotInitializedExceptionMBS	38
• SQLNullMBS	38
• SQLNumericMBS	34
• SQLParamMBS	25
• SQLPositionMBS	13
• SQLPreparedStatementMBS	14
• SQLStringMBS	19
• SQLUnsupportedExceptionMBS	55
• SQLValueMBS	55
• SQLValueReadMBS	39