

Zeos 8.0 Release Notes

Jan Baumgarten

April 7, 2024

1 Zeos 8.0

The Zeos Team is proud to announce the availability of Zeos 8.0 as a stable release. This is the newest stable version of Zeos. It deprecates Zeos 7.2 and all prior versions of Zeos. Zeos 8.0 has seen tons of bug fixes and improvements. We urge all people still using older versions of Zeos to upgrade. If you have any problems with Zeos 8.0, please get in contact with us on the forums (<http://zeoslib.sourceforge.io>) or on the bugtracker (<https://sourceforge.net/p/zeoslib/tickets/>).

2 General Changes

2.1 Supported compilers

Zeos 8.0 supports Delphi versions from Delphi 7 to 12. Besides the x86 and x86_64 compilers, Zeos now also supports the Nextgen compilers (starting from Delphi XE2), and so supports all available platforms on Delphi. So on Delphi Windows, macOS / Mac OS X, iOS and Android are supported now. The Free Pascal compiler is supported version 3.0 to version 3.2.2. Older versions might work but don't get official support by us anymore.

2.2 New Drivers

Zeos 8.0 now comes with some new drivers:

- OleDB

- ODBC
- Firebird with new interface based API
- Webservice Proxy Driver

For a description of these new drivers please see the respective chapters.

2.3 no more driver versioning

We deprecated protocol names with server versions during the life time of Zeos 7.2 already. These protocols have been removed from Zeos 8.0. So protocol names like "firebird-3.0" will not work anymore. Please use protocol names without version numbers instead, like "firebird".

2.4 Property Documentation

Zeos drivers support a multitude of parameters via the "Properties" property of our components. These parameters are now collected in the new file ZDbcProperties.pas and also in ZPropertiesEditor.pas. Based on the latter file, we will release a more detailed documentation for Zeos driver properties.

2.5 Support for connection loss

Zeos 8.0 now supports the programmer in reacting to a connection loss. If Zeos detects a connection loss, it will

- close all handles
- close all results and datasets
- fire the OnLost event of TZConnection

Some drivers are not supported for this feature:

- WebserviceProxy
- SQLite
- ASA (Adaptive Server Anywhere)

On the bridge drivers (ADO, OLEDB and ODBC) the feature is not fully tested. If you use one of them, we suggest, you do some tests for your application.

2.6 New prepare logic

Some drivers don't prepare statements on first execution automatically. Instead they prepare statements after some executions now. This is for cases where statements get used once only. Preparing these statements would be a waste of resources in that case. This behavior is controlled by the `MinExecCountBeforePrepare` parameter. Setting this parameter to a value greater than zero, means that the statement will only be prepared after that many executions. Setting the parameter to zero means that the statement will be prepared before the first execution. Setting this parameter to a negative value will turn off preparing completely.

Currently the following drivers support this parameter:

- PostgreSQL
- MySQL / MariaDB

For ODBC, OLEDB and ADO the `DeferPrepare` property can be used.

Finally there is `PreferPrepared`. `PreferPrepared` gets mapped to `DeferPrepared` for OleDB, ODBC and ADO.

2.7 DBC layer: changed `GetPWideChar` / `GetPAnsiChar Len` parameter

The `Len` parameter of the `GetPWideChar` and `GetPAnsiChar` method is no longer a pointer. A `var` parameter is used instead.

2.8 Support for `TBCDField` / `TFMTBCDField`

For correct support of `NUMERIC` and `DECIMAL` field types, Zeos now supports the use of `TBCDField` and `TFMTBCDField`. The DBC layer has been extended accordingly. If you use static field definitions in your forms, you will need to adapt them before compiling your application for Zeos 8.0.

Note: TBCDField doesn't usually round values. Assigning 1.065 works for a numeric(15,2) database field. This value usually will also be presented to the user (TField.AsString). To remedy situations where the database would round the value and Zeos and the database have a different view of the fields value, Zeos will round assigned values to the scale defined by the underlying database fields before writing them to the field buffer. So Zeos will round 1.065 to 1.07 and then send 1.07 to the database.

2.9 doPreferPrepared defaults to on now

The option doPreferPrepared defaults to on now. If this option is enabled, the MinExecutionCountBeforePrepare parameter gets set to 0. For more information see 2.6 New prepare logic, page 3.

2.10 Ordinal position of return value for stored procedures

For database systems, where a return value is defined, like MS SQL Server, the return value has moved to ordinal position 0. This change affects dblib based drivers (MS SQL Server / Sybase Adaptive Server Enterprise) as well Oracle, MySQL / MariaDB, OLEDB, ODBC and ADO. Please check your applications.

2.11 support for aborting long running operations

Zeos now supports the abortion of long running operations. To do so, call TZConnection.AbortOperation from another thread. This is supported on MySQL, MariaDB, Oracle, dblib (MS SQL Server, SAP Adaptive Server Enterprise), PostgreSQL, Firebird and SQLite.

2.12 new method StartTransaction on the DBC layer

Until Zeos 8.0 the DBC layer didn't have a way to start transactions besides leaving AutoCommit mode. Zeos 8.0 introduces a new StartTransaction method. This can save API calls - especially on databases that default to using auto commit because it doesn't start unnecessary transactions after Commit / Rollback.

2.13 Nested transactions

Zeos now supports nested transactions. `TZConnection.StartTransaction` on the component layer and `IZConnection.StartTransaction` will return the current transaction nesting level. If a transaction already has been started, Zeos will use save points to create nested transactions automatically. When issuing a Commit or Rollback, Zeos will act accordingly by releasing save points or rolling back to save points.

2.14 `stBigDecimal` and `stCurrency` have new meaning

The two data types `stBigDecimal` and `stCurrency` are not used for describing `TExtendedFields`. They now are used for describing decimal and numeric field types.

2.15 Zeos Field Classes

Zeos 8 introduces its own field classes. They derive from the original Fields, so can be used almost seamlessly. The Zeos field classes can work directly on Zeos internal structures and by this way are way faster then normal TField descendants. Also Zeos implements some field classes to support features of databases, that are not supported by the Delphi / Free Pascal TField implementations or to enable the use of database features on older Delphi versions where the RTL does not provide some field types. Examples are the `TZUInt64Field` and Fields that support MySQLs dates with the day or month being set to 0.

The new Zeos fields can be disabled, if necessary, by setting the `DisableZFields` property of a Zeos Dataset to True. In this case Zeos will not generate `TZFieldDefs` and will subsequently not generate the new Zeos fields.

2.15.1 `TZDateTimeField`

`TZDateTimeField` is a replacement for `TDateTimeField`. It allows for greater flexibility when it comes to displaying and editing Dates and Times. An example are dates with 0 components in them on MySQL and MariaDB. Also it allows a more accurate display of timestamps - up to 1 microsecond.

2.16 New parameters for handling of exceptions and warnings

Zeos 8.0 introduces two new parameters for the handling of exceptions and warnings: `AddLogMsgToExceptionOrWarningMsg` and `RaiseWarningMessages`

2.16.1 AddLogMsgToExceptionOrWarningMsg

This is a boolean option. If set to true (default) The offending SQL will be added to exceptions generated by the server. If set to false, SQL statements will not be added.

On the DBC layer this option is reflected by the `SetAddLogMsgToExceptionOrWarningMsg` method of the `IZConnection` Interface.

2.16.2 RaiseWarningMessages

This is a boolean option. If set to true, warnings by the SQL server will be raised as exceptions. Default is false.

On the DBC layer this option is reflected by the `SetRaiseWarningMessages` method of the `IZConnection` Interface.

2.17 Zeos now respects TField.ProviderFlags

Until Zeos 7.2 Zeos could update queries only if they queried a single table and all fields were writable. Starting with Zeos 8.0 Zeos uses information from `TField.ProviderFlags` and `TField.ReadOnly`. This allows us to have more queries be updateable automatically.

Fields that don't have the `pfInUpdate` flag set will not get updated by Zeos.

Fields that don't have the `pfInWhere` flag set will not get used for constructing the where clause of an DML statement.

2.18 output parameter support on TZQuery

`TZQuery` now supports output paramaters. To use this feature, you need to disable the `ParamCheck` property of `TZQuery` and register the parameters

manually. Types and sizes of parameter need to be set accordingly. See the following example:

```
Query.ParamCheck := False;  
Query.SQL.Text := 'CALL abtest(?, ?, ?, ?, ?)';  
Query.Params.CreateParam(ftInteger, 'P1', ptInPut);  
Query.Params[0].AsInteger := 10;  
Query.Params.CreateParam(ftInteger, 'P2', ptInPut);  
Query.Params[1].AsInteger := 20;  
Query.Params.CreateParam(ftString, 'P3', ptInPut);  
Query.Params[2].Precision := 10;  
Query.Params[2].AsString := 'xx';  
Query.Params.CreateParam(ftInteger, 'P4', ptOutPut);  
Query.Params.CreateParam(ftString, 'P5', ptOutPut);  
Query.Params[4].Precision := 20;  
Query.ExecSQL;
```

2.19 smaller memory footprint

By optimizing access in TDataset descendants, Zeos is able to save on memory now if TZFields are used, which is the default for Zeos 8.0.

2.20 TZParam

Zeos now implements its own parameter type TZParam. This will lead to problems when parameters are stored in dfm/lfm files. The new TZParam became necessary to support batch DML statements on the component layer - with TZQuery and TZReadOnlyQuery.

2.21 new component TZTransaction

Zeos 8.0 introduces a new TZTransaction object. This enables Zeos datasets to fetch data from one transaction and write changed data to another transaction. Also it allows Zeos to have more than one transaction per connection - if the underlying driver supports it. To support these new possibilities, TZReadOnlyQuery introduces the new Transaction property. Using this property one can tell TZReadOnlyQuery which connection to use for fetching its data. TZQuery and TZTable introduce an UpdateTransaction property, which can be used if changed data should be written on a separate transaction.

When using TZTransaction, it makes sense to avoid using the StartTransaction, Commit and Rollback methods and to use the according methods from TZTransaction instead.

On Firebird and Interbase Zeos can use their support for having multiple concurrent transactions per connection. On all other database systems Zeos will create a separate connection per transaction. This can lead to problems if your license restricts the connection count or if one uses one time passwords to secure databases.

A simple example how to use the new TZTransaction can be seen in the test case TZGenericTransactionTestCase.TestQueryTransaction.

3 Firebird

3.1 Support for TGUID-Field

Zeos now supports different ways to store GUIDs in Firebird and map them to TGUIDField:

- SetGUIDByType: all CHAR(16) CHARACTER SET OCTETS type columns will be treated as GUID columns. Can be used on Connection and DataSet.
- GUIDDomains: list of database domains that are to be treated as GUID columns. List separators are ';' and ','. Can be used on connection only.
- GUIDFields: list of fields that should be treated as GUID type fields. List separators are ';' and ','. Can be used on Dataset only.

3.2 Support for timeouts on Firebird 4.0

Zeos now supports more timeouts, as introduced with Firebird 4.0:

- StatementTimeOut
- SessionIdleTimeOut

3.3 Usage of new interface based API

The firebird driver uses the new interface based API that was introduced with Firebird 3.0. If the client library doesn't support this new API, it will automatically fall back to the ISC API. If you want to force the usage of the old API for some reason, you can set the option "FirebirdAPI" on the connection properties. If you set this property to "interface", the new interface based API will get used. If you set this property to "legacy", the legacy ISC API will get used.

4 MySQL / MariaDB

4.1 MySQL_FieldType_Bit_1_IsBoolean is enabled by default

The parameter `MySQL_FieldType_Bit_1_IsBoolean` is enabled by default now if your server version is equal to or greater than 5.0.3. This means that fields declared as `BIT(1)` will be treated as boolean fields. The old assumption that an `enum('Y','N')` is a boolean field is disabled if this parameter is enabled. If this parameter is enabled, `enum('Y','N')` will be a string field. Other enums behave as before, they will be mapped to a string field in any case.

4.2 Support for batch DML on MariaDB

Zeos now supports Batch DML with `INSERT` and `DELETE` on MariaDB.

4.3 changed parameter names

Some parameter names have changed to make them comply with the MySQL manual. See <https://dev.mysql.com/doc/refman/8.0/en/mysql-options.html>

old name	new name
<code>MYSQL_SSL_KEY</code>	<code>MYSQL_OPT_SSL_KEY</code>
<code>MYSQL_SSL_CERT</code>	<code>MYSQL_OPT_SSL_CERT</code>
<code>MYSQL_SSL_CA</code>	<code>MYSQL_OPT_SSL_CA</code>
<code>MYSQL_SSL_CAPATH</code>	<code>MYSQL_OPT_SSL_CAPATH</code>
<code>MYSQL_SSL_CYPHER</code>	<code>MYSQL_OPT_SSL_CIPHER</code>

In general it makes sense to check if your parameters match the MySQL / MariaDB documentation now.

5 PostgreSQL

5.1 Support for batch DML

Zeos now supports BATCH DML on PostgreSQL. Supported statements are INSERT and DELETE statements.

5.2 usage of binary protocol

The PostgreSQL driver now uses the binary protocol for talking to the database. This allows faster fetches for the dataset. We also prevent rounding errors for TFloatField and escape the "parsing hell" for date and time values. The parameters DateReadFormat, DateWriteFormat, TimeReadFormat, TimeWriteFormat, DateTimeReadFormat and DateTimeWriteFormat don't get used anymore. This can lead to behavior changes between Zeos 7.2 and Zeos 8.0.

Note: Zeos still supports using text format result sets. Setting Binary-WireResultMode to false will revert to using text results and should allow one to use the more unusual data types of PostgreSQL.

5.3 compatibility option BindDoubleAsString

A lot of our users use the AsFloatMethod to set parameter values. Unfortunately PostgreSQL will generate an error if we send Float values for usage in NUMERIC fields. Using this option will send Float (Double) values as Strings to the database and circumvent this behavior. For more information on the use case see <https://sourceforge.net/p/zeoslib/tickets/282/>.

6 SQLite

6.1 SQLite tokenizer now recognizes backticks as quotes

The SQLite tokenizer now correctly recognizes backticks as quotes. Even though not standard SQL, backticks are supported by SQLite for compatibility. See (https://www.sqlite.org/lang_keywords.html). This should make using backticks in SQL safe with SQLite on Zeoslib.

6.2 better error descriptions

The SQLite driver now supports the proper use of `sqlite3_errstr` which leads to better error descriptions. If you use the Error message of exceptions, you should check your code if it needs to be updated. (See <https://sourceforge.net/p/zeoslib/tickets/260/>)

Note: In Zeos 7.2 we introduced the new parameter "ExtendedErrorMessage" for enabling this new behavior. In Zeos 7.3 the "ExtendedErrorMessage" is removed and the new behavior cannot be disabled.

6.3 support for GUID fields

The sqlite driver now supports GUID fields. Fields that have GUID in their type description will be treated as TGUIDField. Fields that are not 16 bytes wide will be treated as being null. Zeos supports reading GUIDs that are in a string representation with and without curly brackets.

6.4 support for TBCDFields

Any Field with a fielddef of `*INT*(18,4)` will be treated as a TBCDField.

Mode	Value
OCI_DEFAULT	0
OCI_MIGRATE	1
OCI_SYSDBA	2
OCI_SYSOPER	4
OCI_PRELIM_AUTH	8

Table 1: Values for OCIAuthenticateMode

7 Oracle

7.1 UTF8 is aliased

The character set UTF8 is now aliased to AL32UTF8 because the Oracle UTF8 is broken. It is CESU-8 in reality which Zeos can't convert reliably. This might lead to problems on older servers.

7.2 New parameter OCIAuthenticateMode

We introduced a new parameter OCIAuthenticateMode for Oracle authentication. This parameter maps to the mode parameter of the OCISessionBegin function (https://docs.oracle.com/cd/B10501_01/appdev.920/a96584/oci15r13.htm). OCIAuthenticateMode needs to be set as an integer because the flags it represents can be combined in some cases. Possible values can be seen in table 1.

7.3 Support for old installations

Zeos does not support Oracle installations older than Oracle 9i.

8 dblib (MS SQL Server with freetds / SAP Adaptive Server Enterprise)

8.1 discontinued support for ntwdblib.dll

We discontinued the support for ntwdblib.dll. If you try to use ntwdblib.dll, Zeos will raise an error. Please use freetds instead.

8.2 new parameter TDSVersion

FreeTDS allows to select the proper TDS protocol version for your server. See <https://www.freetds.org/userguide/ChoosingTdsProtocol.html> The new parameter TDSProtocolVersion reflects this possibility. Possible values are "4.2", "5.0", "7.0", "7.1", "7.2", "7.3" and "7.4".

8.3 changed behavior for empty Strings and TZStoredProc

The dblib API doesn't allow to send empty strings to stored procedures. This is a limitation of the dbRpcParam function of the API. Zeos now exhibits this behavior when TZStoredProc gets used. Previous versions of Zeos inserted the NUL (#0) character instead.

9 OleDB

Zeos now supports OleDB for Delphi as well as for Free Pascal. It is meant to replace the ADO driver and is faster than ADO as well. To use this driver, insert the OLE DB Connection string into the Database property of TZConnection. The host name property doesn't work with this driver. This driver was mostly tested with Microsoft SQL Servers during development.

9.1 parameters

paramater name	description
internal_buffer_size	Drivers like ODBC, OleDB, Oracle, ASE(SACAPI) do allow block-fetches to reduce roundtrips. Define memory in bytes for the block buffer. Default is 128Kb Zeos will do a minimum allocation for one row.
Provider	The OleDB Provider if not specified in the connection string.
Trusted_Connection	Use trusted connection?

DeferPrepare	Defer prepare? If not set Zeos tries to prepare update, delete, insert and select statements immediately. Also Zeos tries to determine the parameter types, allocates the parameter buffers once only and does not use late binding of parameters. Thus it is faster if NO defer prepare is used and the statement gets used more than once. Some servers might fail to prepare the statements (MS products are masters at failing including unknown exceptions). Turn it off on the DataSet or Statement level if you run into that issue.
StatementTimeout	Execution timeout of a statement in seconds.
MarsConn	Enables MARS (Multiple Active Result Sets) on Microsoft SQL Server drivers.
Initial Catalog	The initial catalog to use for the connection.

9.2 Note for users of Microsoft SQL Server

Don't forget to add the "MarsConn=True" parameter to the connection string unless UseMetadata is turned off. Otherwise Zeos will not be able to get metadata from the server while it reads a result set. For more information see (<https://sourceforge.net/p/zeoslib/tickets/434/#9e48>).

10 ADO

ADO is deprecated. Use OleDb instead.

11 ODBC

Zeos now supports ODBC for Delphi as well as for Free Pascal. The driver registers two drivers, namely `odbc_a` and `odbc_w`. This allows to use Wide Char drivers as well as ANSI drivers. To use this driver, insert the ODBC Connection string into the Database property of `TZConnection`. For examples see (<https://www.connectionstrings.com/>). The host name property doesn't work with this driver. This driver was mostly tested with Microsoft SQL Servers during development. Support for data source names is implemented in that the driver checks if there is an equals sign (=) in the database

property. If there is no equals sign, the driver assumes that only the name of a DSN was given and internally converts to a connection sting in the form "DSN=<Database>".

11.1 parameters

paramater name	description
internal_buffer_size	Drivers like ODBC, OleDb, Oracle, ASE(SACAPI) do allow block-fetches to reduce roundtrips. Define memory in bytes for the block buffer. Default is 128Kb Zeos will do a minimum allocation for one row.
Trusted_Connection	Use trusted connection?
DeferPrepare	Defer prepare? If not set Zeos tries to prepare update, delete, insert and select statements immediately. Also Zeos tries to determine the parameter types, allocates the parameter buffers once only and does not use late binding of parameters. Thus it is faster if NO defer prepare is used and the statement gets used more than once. Some servers might fail to prepare the statments (MS products are masters at failing including unknown exceptions). Turn it off on the DataSet or Statement level if you run into that issue.
StatementTimeOut	Execution timeout of a statement in seconds.
DriverCompletion	Refer to ODBC manual for details. Possible values are: SQL_DRIVER_COMPLETE, SQL_DRIVER_PROMPT, SQL_DRIVER_COMPLETE_REQUIRED
Server	The Server to connect to.

CharacterSet	Specifies the character set used to interact with the driver. If you are accessing a CharacterSet "NONE" Firebird / Interbase database you should always use the character set "NONE" as the connection character set. In addition it is recommended to specify which character set the "NONE" represents by adding the Charset_NONE_Alias parameter. Example: Properties.Values['Charset_NONE_Alias'] := 'WIN1251'. For odbc_a and ole_db(raw longvarchar only) it is implemented as: Set a custom characterset to notify Zeos about conversion routines. Note for odbc_a: This CodePage must be equal for all fields(ODBC). Otherwise use the odbc_w driver. Here it is defined as: <Character Set Alias>:<Codepage>/<MaxBytes>where "Character Set Alias" is the character set alias, "Codepage" is the Windows code page number to be used and "MaxBytes" is the maximum number of bytes in the character set. Example: CharacterSet=latin1:1252/1 or CharacterSet=utf8:65001/4
DRIVER	Specifies the ODBC driver to be used on this connection.

11.2 Note for users of Microsoft SQL Server

Don't forget to add the "MarsConn=True" parameter to the connection string unless UseMetadata is turned off. Otherwise Zeos will not be able to get metadata from the server while it reads a result set. For more information see (<https://sourceforge.net/p/zeoslib/tickets/434/#9e48>).

12 Web Service Proxy

The Web Service Proxy driver is meant to work in combination with a special Zeos based web service server. The server can access any database supported by Zeos. It is Web Service based because the server and the client use SOAP over HTTP(S) to communicate with each other. This allows for several scenarios where regular database connections can be problematic:

- Scenarios where the connection may be unstable or where the connection and IP addresses used may change. I.e. access over the internet or mobile networks or connections where the client transitions from mobile connections to wireless lan or even wired connections.
- Work on clients where a native access library is not available. I.e. Delphi Apps for mobile phones.

Also the web service proxy server can add a layer of security because it can add one time password schemes to the authentication used.

For more information on the Web Service Proxy see <https://sourceforge.net/p/zeoslib/wiki/WebServiceProxyDriver/>

13 new unit - TZMethodInThread

TZMethodInThread was the first CONCEPT of a runtime component which allows the users to force long-lasting SQL operations to the background, like a simulated synchronous mode. It is useful when used together with TZConnection.AbortOperation. However TZMethodInThread works, it is only a concept and we highly encourage users to implement their own worker threads of their flavor.

Each TZMethodInThread instance can run one operation at a time, but more instances can be created if necessary.

Usage:

```
Procedure TDummy.OnError(Sender: TObject; Error: Exception);  
Begin
```

```
  //Make sure you do proper synchronization here,  
  //as this method is run from within the context  
  //of the thread!
```

```
    WriteLn(Error.Message);  
End;
```

```
Procedure TDummy.OnFinish(Sender: TObject);  
Begin
```

```
  //Make sure you do proper synchronization here,  
  //as this method is run from within the context  
  //of the thread!
```

```
    WriteLn('Finished!');  
End;
```

```

Procedure OpenInBackground;
Begin
    methodthread := TZMethodInThread.Create;
    methodthread.OnError := TDummy.OnError;
    methodthread.OnFinish := TDummy.OnFinish;
    myquery.SQL.Text := 'SELECT * FROM VeryHugeTable';
    methodthread.Open(myquery);
End;

```

If you are using data-aware components, make sure to unassign any data-sources before (or call `.DisableControls`), as the VCL thread is not blocked and data events will be attempted to be processed at inconsistent states. This will most probably result in access violations or other unwanted behavior. Proper synchronization is also needed at event handlers (like `TZQuery.BeforeOpen`, `.AfterOpen`, `.AfterScroll`, `TZConnection.AfterConnect`, etc.) as if specific actions are called from a worker thread, these event handlers will also run in the worker's context!

A long running operation can be canceled in two ways:

- Graceful: call `ZConnection.AbortOperation`. This is not implemented for all drivers; and implemented, but untested at some. If it's not supported, an `EZUnsupportedException` will be raised. Only use this method to keep data consistency.
- Ungraceful: call `ZMethodInThread.Kill(True)`. With no further questions, this will terminate the background thread leaving the query and the connection in an unknown state. Expect memory leaks, permanent loss of connection to the database, crash of the application or data corruption in the database. Use only at your own risk... or never.

Disclaimer: Please keep in mind that `TZMethodInThread` (and pushing specific actions to background threads) were not tested in every scenario and might cause issues in unique cases. It is still a viable alternate until Zeos implements proper asynchronous functionality.

14 Known Limitations

14.1 Negative Time Intervals on PostgreSQL

Zeos internal structures don't allow handling of negative time intervals. Also there is no field type we can use to display negative intervals. Because of these limitations negative intervals will be returned with all parts (hours, minutes, seconds, microseconds) set to 0 (zero).