SQL Manager.net™
EMS® Software Development

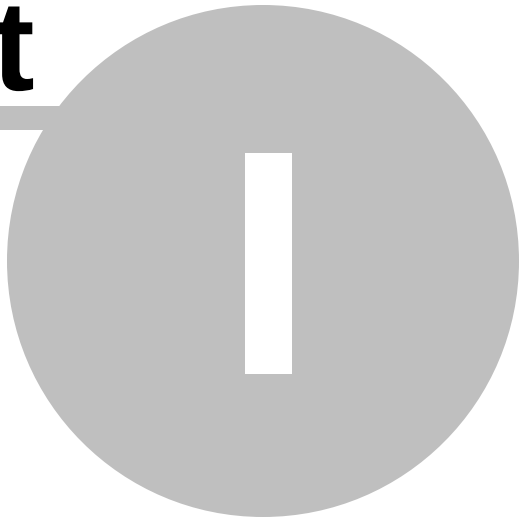
Advanced Data Export for RAD Studio VCL User's Manual

Advanced Data Export for RAD Studio VCL
User's Manual

Document generated on: 22.11.2023

Table of Contents

Part



1 Welcome to Advanced Data Export for RAD Studio VCL!

1.1 Overview

Advanced Data Export for RAD Studio VCL is a component that allows you to save your data in the most popular data formats for the future viewing, modification, printing or web publication. You can export data into MS Access, MS Excel, MS Word (RTF), Open XML Format, Open Document Format (ODF), HTML, XML, PDF, TXT, DBF, CSV and more! There will be no need to waste your time on tiresome data conversion - Advanced Data Export will do the task quickly and produce the result in the desired format.

Visit our web-site for details: http://www.sqlmanager.net/

Key features

- Data export into 17 most popular formats: MS Access, MS Excel, MS Word, Open XML Format, Open Document Format (ODF), RTF, HTML, XML, PDF, TXT, DBF, CSV, SYLK, DIF, LaTeX, SQL and Windows Clipboard
- Export of Unicode data. Manually preset text encoding for exported data (UTF-8, UTF-16/UCS-2, UTF-32/UCS-4, Latin1, Latin2, Latin5, Latin7 and more)
- Saving data for future viewing, modification, printing or web publication
- Easy-to-use wizard allows your end-users to export data quickly
- Powerful export options for each data format
- 100% native Delphi code
- No additional libraries or software required to operate
- Detailed help system and demo application
- Powerful components and properties editors
- Delphi 2010, XE-XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11 Alexandria and C++ Builder 2010, XE-XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11 Alexandria, 12 Athens
- 64-bit Windows platform support
- Setting the user formats for each field separately
- Multilanguage support

Product information

Homepage	http://sqlmanager.net/products/tools/advancedexport
Support Ticket System	http://www.sqlmanager.net/support
Register on-line at	http://sqlmanager.net/products/tools/advancedexport/buy

1.2 What's new

Version	Release date
Advanced Data Export for RAD Studio VCL 4.18	November 22, 2023

What's new in Advanced Data Export for RAD Studio VCL 4.18?

- Support for RAD Studio 12 Athens implemented.
- New types TqeFont, TqeFontStyle, TqeColor were added.
- Fixed using of GREEK_CHARSET for fonts of TQExport4PDF component.
- The temporary folder is now removed after an export in TQExport4Xlsx component.
- The resourse STRINGTABLE defined as language independent.
- Now the QExport4IniFiles.TQIniFile class allows to set up an arbitrary encoding on creation.
- Fixed default format string for Datetime cell in TQExport4Xlsx component.
- Fixed number of exported records when using "Export only" and "Skip records" together in TQExport4XLS component.
- Fixed paths for 32-bit Clang compiler in RAD Studio options.

See also:

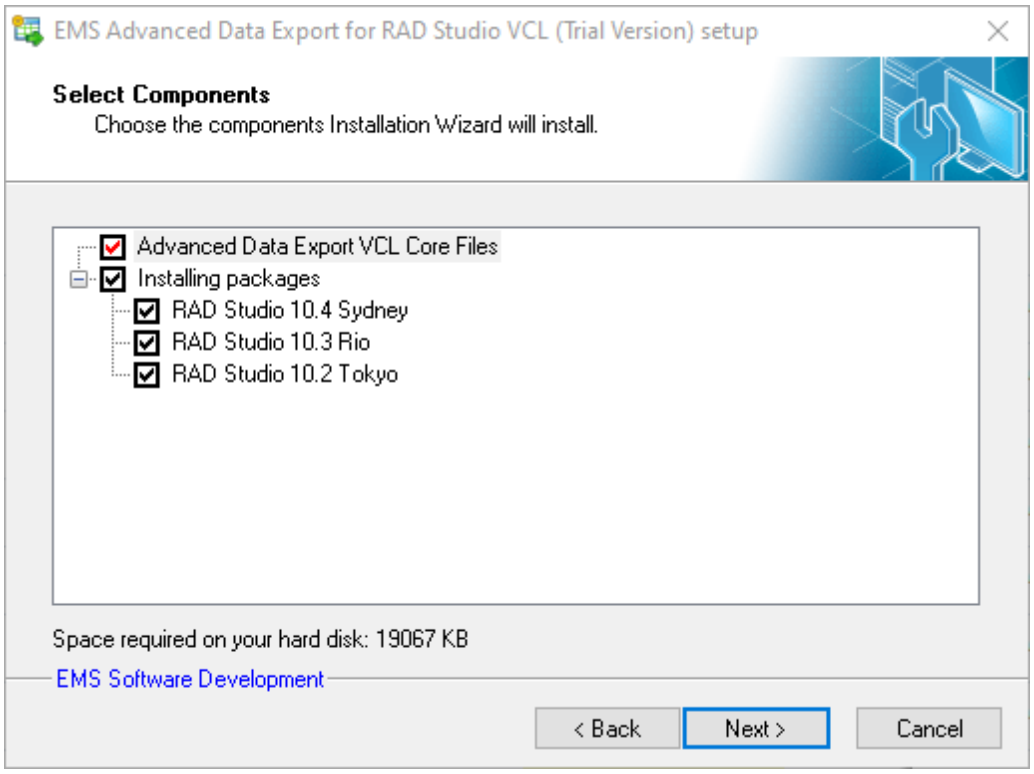
Version history

1.3 Installation

To install the **trial version** of **Advanced Data Export for RAD Studio VCL** onto your system:

- download the distribution package of **Advanced Data Export for RAD Studio VCL** from the download page available at our website;
- unzip the downloaded file to any local directory, e.g. *C:\unzipped*;
- close all currently opened Delphi and/or C++ Builder IDEs, if any;
- run the executable setup file from the local directory and follow the instructions of the installation wizard.

During the installation you will need to select the packages to install and set options that will take effect **only for installed IDE**:



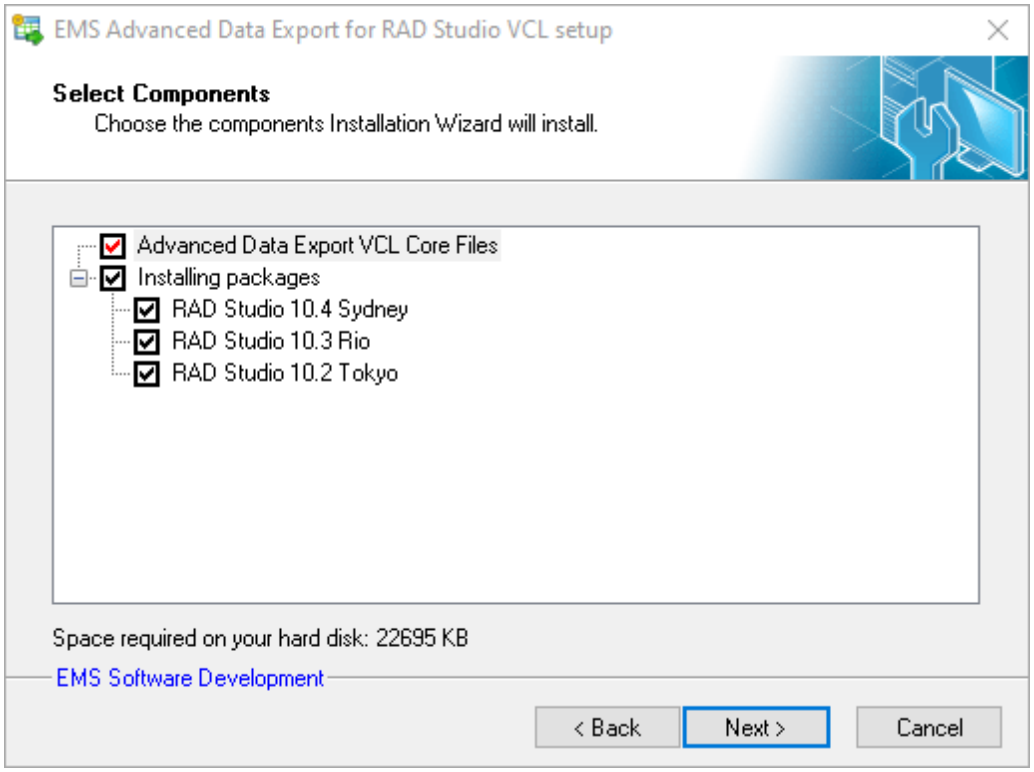
When you are done, you can finish installation of the **trial version** of **Advanced Data Export for RAD Studio VCL**.

To install the **full version** of **Advanced Data Export for RAD Studio VCL** onto your system:

- download the distribution package of **Advanced Data Export for RAD Studio VCL** from the download page available at our website;
- unzip the downloaded file to any local directory, e.g. *C:\unzipped*;
- close all currently opened Delphi and/or C++ Builder IDEs, if any;
- run the executable setup file from the local directory and follow the instructions of the installation wizard.

Enter valid registration information in the appropriate boxes: **Registration name** and **Registration Key**. See details on getting this information.

During the installation you will need to select the packages to install and set options that will take effect **only for installed IDE**:



When you are done, you can finish installation of the **full version** of **Advanced Data Export for RAD Studio VCL**.

Note: If the above given instructions have been insufficient for successful installation of the component suite, please refer to the *readme.1st* file distributed with the product.

1.4 Registration

All purchases are provided by **Digital River** registration service. The **Digital River** order process is protected via a secure connection and makes on-line ordering by credit/debit card quick and safe.

Digital River is a global e-commerce provider for software and shareware sales via the Internet. It accepts payments in US Dollars, Euros, Pounds Sterling, Japanese Yen, Australian Dollars, Canadian Dollars or Swiss Franks by Credit Card (Visa, MasterCard/ EuroCard, American Express, Diners Club), Bank/Wire Transfer, Check or Cash.

If you want to review your order information, or you have questions about ordering or payments please visit our Customer Care Center, provided by **Digital River.**

Please note that all of our products are delivered via ESD (Electronic Software Delivery) only. After purchase you will be able to immediately download the registration keys or passwords. Also you will receive a copy of registration keys or passwords by email. Please make sure to enter a valid email address in your order. If you have not received the keys within 2 hours, please, contact us at sales@sqlmanager.net.

Product distribution	MyCommerce/Digital River
Advanced Data Export for RAD Studio VCL Full version (with sources) + 1-Year Maintenance*	Register Now!
Advanced Data Export for RAD Studio VCL Full version (with sources) + 2-Year Maintenance*	
Advanced Data Export for RAD Studio VCL Full version (with sources) + 3-Year Maintenance*	
Advanced Data Export for RAD Studio VCL Trial version	Download Now!

***EMS Maintenance Program** provides the following benefits:

- Free software bug fixes, enhancements, updates and upgrades during the maintenance period
- Free unlimited communications with technical staff for the purpose of reporting Software failures
- Free reasonable number of communications for the purpose of consultation on operational aspects of the software

After your maintenance expires you will not be able to update your software or get technical support. To protect your investments and have your software up-to-date, you need to renew your maintenance.

You can easily reinitiate/renew your maintenance with our on-line, speed-through Maintenance Reinstatement/Renewal Interface. After reinitiating/renewal you will receive a confirmation e-mail with all the necessary information.

1.5 How to register Advanced Data Export

To register your newly purchased copy of **EMS Advanced Data Export for RAD Studio VCL**, perform the following steps:

- receive the notification letter from **Share-it!** with the registration info;
- enter the **Registration Name** and the **Registration Key** from this letter while installing the **full version** of the product.

See also:

Registration

1.6 Version history

Product name	Version	Release date
Advanced Data Export for RAD Studio VCL	**Version 4.17**	September 28, 2021
Advanced Data Export for RAD Studio VCL	**Version 4.16**	July 6, 2020
Advanced Data Export for RAD Studio VCL	**Version 4.15**	December 13, 2018
Advanced Data Export for RAD Studio VCL	**Version 4.14.1**	April 18, 2017
Advanced Data Export for RAD Studio VCL	**Version 4.14**	March 29, 2017
Advanced Data Export for RAD Studio VCL	**Version 4.13.2**	May 18, 2016
Advanced Data Export for RAD Studio VCL	**Version 4.13**	April 27, 2016
Advanced Data Export for RAD Studio VCL	**Version 4.12**	October 13, 2015
Advanced Data Export for RAD Studio VCL	**Version 4.11**	May 28, 2015
Advanced Data Export for RAD Studio VCL	**Version 4.10**	October 23, 2013
Advanced Data Export for RAD Studio VCL	**Version 4.9**	May 16, 2013
Advanced Data Export for RAD Studio VCL	**Version 4.8**	August 24, 2012
Advanced Data Export for RAD Studio VCL	**Version 4.7**	March 2, 2012
Advanced Data Export for RAD Studio VCL	**Version 4.6**	October 10, 2011
Advanced Data Export for RAD Studio VCL	**Version 4.5**	July 18, 2011
Advanced Data Export for RAD Studio VCL	**Version 4.4**	December 27, 2010
Advanced Data Export for RAD Studio VCL	**Version 4.3**	November 11, 2010
Advanced Data Export for RAD Studio VCL	**Version 4.2**	January 11, 2010
Advanced Data Export for RAD Studio VCL	**Version 4.1**	September 24, 2008
Advanced Data Export for RAD Studio VCL	**Version 4.0**	April 4, 2008

Full version history is available at https://www.sqlmanager.net/products/tools/advancedexport/news

Version 4.17

- Support for RAD Studio 11 Alexandria added.
- The Captions option published for TQExport4SQL to change column names in SQL script.

Version 4.16

- Support for RAD Studio 10.4 Sydney implemented.
- Invalid characters are not allowed in result files now.
- Margins can be now customized in TQExport4Xlsx and TQExport4Docx components.
- Data can be exported without calling the dialog using the TQExport4Dialog.Execute (const ConfigFileName: string) method.
- The XLSX file couldn't be opened if SheetName property contained spaces. Fixed now.
- Redundant attributes removed from the pageSetup tag in the XLSX result file.
- The 'Could not convert variant of type (Null) into type (OleStr)' error fixed for exporting TQuery data containing NULL values into XLS.
- Configuration files are now loaded faster.
- End of support for RAD Studio 2009 and older versions.
- Other improvements and bugfixes.

Version 4.15

- Implemented support of RAD Studio 10.3 Rio.
- Encoding for XML files has been added.
- The error occurred on exporting data into XLSX with custom export source and AutoCalcColWidth property set to True. Fixed now.
- Invalid XLS file structure was created with AutoCalcColWidth property set to True. Fixed now.

- XLS. The error occurred if Sheets property contained only one element. Fixed now.
- XLS. Temporary files were created in the application folder, which resulted in access errors in some cases. Fixed now.
- DoExport method didn't work correctly if more than 65536 rows were processed. Fixed now.
- Headers and footers for print page in XLS were not displayed correctly. Fixed now.
- EDatabaseError exception occurred on executing the TCustomSQLDataSet. GetRecordCount method. Fixed now.
- Custom values for ThousandSeparator property were not saved in the configuration file. Fixed now.
- ODS. Date and time values were not exported correctly. Fixed now.
- Many other improvements and fixes.

Version 4.14.1

- Support of RAD Studio 10.2 Tokyo added.
- The possibility to set a start topic of the help file in the TQExport4Dialog.HelpTopic property added.
- The TXlsFieldFormat.Width property did not work. Fixed now.
- Some other small bug fixes.

Version 4.14

- Support for 64-bit Windows target platform has been added.
- Support for Unicode configuration files added.
- Now it is possible to export data into XLSX files, containing Unicode symbols in names.
- ODS and ODT files with Unicode symbols, generated by the component, were opened with MS Excel and MS Word with errors. Fixed now.
- Now all components except for TADO_QExport4Access and TQExport4DBF export BLOB and binary data as Base64 and HEX strings.
- Now you can set column width in ColumnWidths property of the TQExport4XLS component with TqeCustomSource4 as source and no fields defined.
- AutoCalcColWidth property is now applied correctly in XLS files.
- XLSX files were opened with errors if data cell styles were changed in OnGetDataParams event handler. Fixed now.
- The Access violation error occurred on getting size and precision for TFmtBCDField field types. Fixed now.
- Now the formatting strings for TXlsxCellStyle.NumericFormat property in TQExport4Xlsx component create new styles in the document correctly.
- The errors occurred in TQExport4ODS and TQExport4ODT components on exporting data without adding any style settings. Fixed now.
- Many other fixes and improvements.

Version 4.13.2

- RAD Studio 10.1 Berlin support added
- A few installation errors have been fixed

Version 4.13

- The possibility of exporting string data as hyperlinks has been added.
- The possibility of formatting TBCDField field values as Currency field type has been implemented.
- All 64-bit Integer values with more than 18 digits were converted to Float type with exponential notation. Fixed now.
- Errors on 64-bit run-time QExport4RT_C23.cbproj package compilation have been fixed.
- String height is now adjusted to cell font and data size.

- The invalid byte was added after every line break when exporting to TXT, CSV, DIF, SYLK. Fixed now.
- Integer overflow error has been fixed.
- Images are exported to XLS correctly now.
- The "Cannot load package 'QEXport4RT_C11' error has been fixed.
- The "Not enough timers available" error when using the components inside of IntraWeb application has been fixed.
- Now the 'Destination Directory' value is correctly saved to the template.
- The Database Home list was empty in some cases for Oracle server. Fixed now.
- Now '0000-00-00' zero date is exported correctly from MySQL server.
- Other minor improvements and bugfixes.

Version 4.12

- Added support of RAD Studio 10 Seattle
- There were double fields count when going back to the first step in ImportWizard. Fixed now.
- Fields mapping was cleared on loading template, even if there were no changes in fields information. Fixed now.
- Some graphic artefacts in ImportWizard in Win64 build. Fixed now.

Version 4.11

- RAD Studio XE8 support implemented.
- Export to XLS. Ability to set page properties for the printed page has been implemented.
- Export to PDF. The AutoBreakString property is renamed to BreakString. The way it works has been changed.
- The LoadPropertiesFromStream and SavePropertiesToStream methods have been added.
- Export to Access. Microsoft.ACE.OLEDB.15.0 support is added.
- Multiple bugs related to decimal separator setting have been fixed.
- Export to PDF. The line breaks in MEMO-fields were deleted. Fixed now.

Version 4.10

- Support of RAD Studio XE5 is added.
- QExport4XML. Added export to MS Access XML format files.
- QExportDocx. Added BLOB fields export.
- Other improvements and bug fixes.

Version 4.9

- Added support of Embarcadero RAD Studio XE4.
- Added the possibility to append date/time to the exported file name. The *AppendDateTimeToFileName* and *AppendDateTimeToDatabaseName* (TADO_QExport4Access) properties. Use the *ExportedFileName* and *ExportedDatabaseName* (TADO_QExport4Access) properties to get the name of the file to which the export is performed.
- QExport4SQL. Added the possibility to generate a single INSERT statement. The *MultipleInsert* property.
- TQExport4XML. Now the BLOB filed data is uploaded using base64 encoding. (For Delphi 2005/CBuilder 2006 versions and older).
- QExport4XLS. Now the exported data can be splitted into Excel worksheets by the specified number of rows. The *SplitByRows* property.
- When changing delimiters in the *Formats* property, the delimiters in the corresponding format masks are changed automatically.
- QExport4XLS. Now when trying to export more than 255 columns, an error message occurs.

- QExport4XLS. Fixed the errors of the format mask usage.
- If format masks correspond to system masks, they were not saved in the **.dfm* file. Fixed now.
- When removing a delimiter in the *Formats* property, it was replaced by #0 in the corresponding format masks. Fixed now.
- QExport4DBF. While exporting the *TWideMemoField* fields were recorded in the **.dbt* file in Unicode. Fixed now.
- QExport4SQL. When unloading currency data types, a currency symbol was added to the values. Fixed now.
- Other improvements and bug fixes.

Version 4.8

- QExport4Xlsx. Added the RightToLeftExcelMode property. If RightToLeftExcelMode = True, opening the exported file makes Excel switch to the 'Right to Left' display mode.
- QExport4Xlsx. Now string data with non-printable characters (characters with codes up to # 31) are exported correctly.
- QExport4Xlsx. The values of Date/Time fields were exported as numbers. Fixed now.
- Now all fields, including Blob fields, are exported by default.
- QExport4Dialog. Now it is impossible to export, if none of the fields are selected.
- QExport4ASCII. The Null character was added to the end of each row. Fixed now.
- QExport4ASCII. Export to CSV. The format mask was not applied to real numeric values. Fixed now.
- QExport4ASCII. Export to CSV. The column width is now calculated only for the fields, where the field width is not specified.
- QExport4ASCII. Export to CSV. The export of multibyte characters caused the error. Fixed now.
- QExport4ASCII. Now column width parameters are read from a configuration file correctly.
- QExport4XLS. Values were formatted incorrectly. Fixed now.
- QExport4XLS. The format masks for standard styles were modified while being exported. Fixed now.

Version 4.7

1. QExport4XLS:
   - Added the Aggregates property. This collection allows to define aggregate functions for the range of cells.
   - The row height settings were not applied. Fixed now.
   - The images stretched along the width of data output. Fixed now.
2. TQExport4ASCII:
   - Added the TrimCaptionsToDataLength property. When TrimCaptionsToDataLength=True, the caption width will be aligned along the data columns width.
   - When DataSet=nil and AutoCalcColWidth=True, the data export caused the error. Fixed now.
3. TQExport4Xlsx.
   - In the designer the editor of the StripStylesLis properties could not be called. Fixed now.
   - The font settings were not applied. Fixed now.
   - When processing OnGetDataParams, the resulting file was created incorrectly. Fixed now.
4. QExport4XLS. TQExport4Xlsx. A number of properties were not saved in the configuration file. Fixed now.
5. TQExport4DBF. When using a Chinese locale, the file was created incorrectly. Fixed

now.
6. XML-Based. The data export caused the error. Fixed now.
7. In OnGetExportText the values for null-fields could not be overridden. Fixed now.
8. Other improvements and bug fixes.

Version 4.6

- Added support of Embarcadero RAD Studio XE2.

Version 4.5

- Added the support of MS Access 2007. When the FileVersion property of the TADO_QExport4Access component is equal to afvAccess2007, the export is performed to the MS Access 2007 format.
- TQExport4Xlsx. Now it is possible to set a view of each cell individually using the new OnGetDataParams event.
- TQExport4Xlsx. Added the possibility to determine and set the columns width automatically. The new AutoCalcColWidth property is responsible for this.
- TQExport4Xlsx. The data formatting could not be applied. Fixed now.
- TQExport4Xlsx. Values of the Formats and UserFormats properties were not saved in the configuration file. Fixed now.
- TQExport4Xls. When opening a file with multiple sheets, generated by the component, the cells actions (sorting, filtering, etc.) became inaccessible in Excel. Fixed now.
- TQExport4Xls. In some cases the data export caused the program hang-up. Fixed now.
- TQExport4Dialog. When exporting to MS Access, the "Print file after export" and "Open file after export" options did not work (it was not possible to open and print the file). Fixed now.
- ADO_QExport4Access. When PrintFile=True and ShowFile=True, the ACCESS process remained in the process list after the export is finished. Fixed now.
- Some other improvements and bugfixes.

Version 4.4

- QExport4XLS. The possibility to automatically add new sheets when exporting more than 65536 lines is added.AutoAddSheet property is responsible for this. When AutoAddSheet = True, the new sheets are added automatically. When AutoAddSheet = False - upon reaching the maximum number of the lines the error message on the sheet appears.
- QExport4XLS. The AutoTruncateValue property is added. If the value of more than 32767 symbols is inserted in the cell, the length of the line will be truncated automatically to the enabled when AutoTruncateValue = True. When AutoTruncateValue = False, the error message "String too large" will appear.
- QExport4Dialog. The ability to Drag and Drop fields to the specified position is added. The field was added to the end of the list before.
- When exporting the value over 4110 symbols to Excel the file did not open. Fixed now.
- QExport4Dialog. The error of exporting the empty sheets to PDF occurred when setting the PDFOptions->PageOptions->Units = unMillimeter property. Fixed.
- The error "Index of bounds" occurred with the null Map property. Fixed now.
- QExport4XLS. The error preventing the column width automated setting is fixed.
- QExport4XLS. The error of the line breaking is fixed.
- QExport4XLS. The error causing the application hanging occurred at the following conditions:
  - ✓ the export is set for several sheets;
  - ✓ on one or several sheets the property Exported = False;

Fixed now.
- Other small improvements and bugfixes.

Version 4.3
- The support of Embarcadero RAD Studio XE is added.
- QExport4XLS. The "Out of Memory" error occurred when modifying the Options->FooterFormat->Color property. Fixed now.
- QExport4XLS. The "Data may have been lost" error occurred when exporting data using the AddStringCell function. Fixed now.
- QExport4. The formatting of the Currency type did not work for Docx and XML formats. Fixed now.
- A compile error occurred when compiling the QXMLWriter.pas file in Delphi5. Fixed now.
- Export4Xlsx. If the CustomSource data source contained columns with the String data type, they were not exported. Fixed now.
- QExport4Xlsx. Null values for the DateTime fields were exported as 0. Fixed now.
- QExport4Xlsx. The component didn't run under Windows Vista OS. Fixed now.
- QEXport4XLS. Added a new event handler - OnAdvancedGetExportData.
- TQExport4Dialog. The ExportType property is added.
- Added three demo projects for C++ Builder 2010.
- The installer now creates shortcuts for demo and help files.
- Some other improvements and bugfixes.

Version 4.2
- RAD Studio 2010 support is added.
- Portuguese Brazilian, Turkish, Spanish and Dutch localizations are added.
- When exporting to XLSX, the integer and float types are exported as numeric.
- It is now possible to display DisplayLabel instead of FiledName for exported fields in the TQExport4Dialog.
- Memory leaks are eliminated.
- Errors connected with strip styles occurred when exporting data to DOCX and XLSX. Fixed now.
- Resolved the type naming conflict which occurred when compiling component packages in C++ Builder.
- Other small improvements and bug fixes.

Version 4.1
- Support of RAD Studio 2009 is added
- Minor improvements and bug-fixes

Version 4.0
- Four new data export components have been added:
  - ✓ QExport4Xlsx component allows you to export data to the MS Excel 2007 sheets
  - ✓ QExport4Docx component allows you to export data to the MS Word 2007 tables
  - ✓ QExport4ODS component allows you to export data to the OpenDocument Spreadsheet files (Open Document Format)
  - ✓ QExport4ODT component allows you to export data to the OpenDocument Text files (Open Document Format)
- Unicode support: now you are able to export Unicode data (UTF-8, UTF-16/UCS-2, UTF-32/UCS-4)
- Now you can select Data Packet or Access as a document type (TQExportXMLType = (xtDatapacket2, xtAccess)) for the QExport4XML component
- Due to the new installer the components will be installed and registered in Delphi\C++

Builder environment automatically
- Support of BDS 2006, RAD Studio 2007, Delphi 2007 and C++ Builder 2007 is added
- Other minor improvements and bug-fixes

Scroll to top

See also:

What's new

1.7 Other EMS Products

Quick navigation


MySQL


Microsoft SQL Server


PostgreSQL


InterBase / FireBird


Oracle


IBM DB2


Tools & components

MySQL


SQL Management Studio for MySQL
EMS SQL Management Studio for MySQL is a complete solution for database administration and development. SQL Studio unites the must-have tools in one powerful and easy-to-use environment that will make you more productive than ever before!


SQL Manager for MySQL
Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.


Data Export for MySQL
Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more.


Data Import for MySQL
Import your data from MS Access, MS Excel and other popular formats to database tables via user-friendly wizard interface.


Data Pump for MySQL
Migrate from most popular databases (MySQL, PostgreSQL, Oracle, DB2, InterBase/Firebird, etc.) to MySQL.


Data Generator for MySQL
Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.


DB Comparer for MySQL
Compare and synchronize the structure of your databases. Move changes on your development database to production with ease.


DB Extract for MySQL
Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.


SQL Query for MySQL
Analyze and retrieve your data, build your queries visually, work with query plans, build charts based on retrieved data quickly and more.


Data Comparer for MySQL
Compare and synchronize the contents of your databases. Automate your data migrations from development to production database.

Scroll to top

Microsoft SQL Server

SQL Management Studio for SQL Server
EMS SQL Management Studio for SQL Server is a complete solution for database administration and development. SQL Studio unites the must-have tools in one powerful and easy-to-use environment that will make you more productive than ever before!

EMS SQL Backup for SQL Server
Perform backup and restore, log shipping and many other regular maintenance tasks on the whole set of SQL Servers in your company.

SQL Administrator for SQL Server
Perform administrative tasks in the fastest, easiest and most efficient way. Manage maintenance tasks, monitor their performance schedule, frequency and the last execution result.

SQL Manager for SQL Server
Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.

Data Export for SQL Server
Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more

Data Import for SQL Server
Import your data from MS Access, MS Excel and other popular formats to database tables via user-friendly wizard interface.

Data Pump for SQL Server
Migrate from most popular databases (MySQL, PostgreSQL, Oracle, DB2, InterBase/Firebird, etc.) to Microsoft® SQL Server™.

Data Generator for SQL Server
Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.

DB Comparer for SQL Server
Compare and synchronize the structure of your databases. Move changes on your development database to production with ease.

DB Extract for SQL Server
Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.

SQL Query for SQL Server
Analyze and retrieve your data, build your queries visually, work with query plans, build charts based on retrieved data quickly and more.

Data Comparer for SQL Server
Compare and synchronize the contents of your databases. Automate your data migrations from development to production database.

Scroll to top

PostgreSQL

SQL Management Studio for PostgreSQL
EMS SQL Management Studio for PostgreSQL is a complete solution for database administration and development. SQL Studio unites the must-have tools in one powerful and easy-to-use environment that will make you more productive than ever before!

EMS SQL Backup for PostgreSQL
Creates backups for multiple PostgreSQL servers from a single console. You can use automatic backup tasks with advanced schedules and store them in local or remote folders or cloud storages

SQL Manager for PostgreSQL
Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.

Data Export for PostgreSQL
Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more

Data Import for PostgreSQL
Import your data from MS Access, MS Excel and other popular formats to database tables via user-friendly wizard interface.

Data Pump for PostgreSQL
Migrate from most popular databases (MySQL, SQL Server, Oracle, DB2, InterBase/Firebird, etc.) to PostgreSQL.

Data Generator for PostgreSQL
Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.

DB Comparer for PostgreSQL
Compare and synchronize the structure of your databases. Move changes on your development database to production with ease.

DB Extract for PostgreSQL
Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.

SQL Query for PostgreSQL
Analyze and retrieve your data, build your queries visually, work with query plans, build charts based on retrieved data quickly and more.

Data Comparer for PostgreSQL
Compare and synchronize the contents of your databases. Automate your data migrations from development to production database.

Scroll to top

InterBase / Firebird

SQL Management Studio for InterBase/Firebird
EMS SQL Management Studio for InterBase and Firebird is a complete solution for database administration and development. SQL Studio unites the must-have tools in one powerful and easy-to-use environment that will make you more productive than ever before!

SQL Manager for InterBase/Firebird
Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.

Data Export for InterBase/Firebird
Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more

Data Import for InterBase/Firebird
Import your data from MS Access, MS Excel and other popular formats to database tables via user-friendly wizard interface.

Data Pump for InterBase/Firebird
Migrate from most popular databases (MySQL, SQL Server, Oracle, DB2, PostgreSQL, etc.) to InterBase/Firebird.

Data Generator for InterBase/Firebird
Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.

DB Comparer for InterBase/Firebird
Compare and synchronize the structure of your databases. Move changes on your development database to production with ease.

DB Extract for InterBase/Firebird
Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.

SQL Query for InterBase/Firebird
Analyze and retrieve your data, build your queries visually, work with query plans, build charts based on retrieved data quickly and more.

Data Comparer for InterBase/Firebird
Compare and synchronize the contents of your databases. Automate your data migrations from development to production database.

Scroll to top

Oracle

SQL Management Studio for Oracle
EMS SQL Management Studio for Oracle is a complete solution for database administration and development. SQL Studio unites the must-have tools in one powerful and easy-to-use environment that will make you more productive than ever before!

SQL Manager for Oracle
Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.

Data Export for Oracle
Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more.

Data Import for Oracle
Import your data from MS Access, MS Excel and other popular formats to database tables via

user-friendly wizard interface.


Data Pump for Oracle
Migrate from most popular databases (MySQL, PostgreSQL, MySQL, DB2, InterBase/Firebird, etc.) to Oracle


Data Generator for Oracle
Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.


DB Comparer for Oracle
Compare and synchronize the structure of your databases. Move changes on your development database to production with ease.


DB Extract for Oracle
Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.


SQL Query for Oracle
Analyze and retrieve your data, build your queries visually, work with query plans, build charts based on retrieved data quickly and more.


Data Comparer for Oracle
Compare and synchronize the contents of your databases. Automate your data migrations from development to production database.

Scroll to top

IBM DB2


SQL Manager for DB2
Simplify and automate your database development process, design, explore and maintain existing databases, build compound SQL query statements, manage database user rights and manipulate data in different ways.


Data Export for DB2
Export your data to any of 20 most popular data formats, including MS Access, MS Excel, MS Word, PDF, HTML and more.


Data Import for DB2
Import your data from MS Access, MS Excel and other popular formats to database tables via user-friendly wizard interface.


Data Pump for DB2
Migrate from most popular databases (MySQL, PostgreSQL, Oracle, MySQL, InterBase/Firebird, etc.) to DB2


Data Generator for DB2
Generate test data for database testing purposes in a simple and direct way. Wide range of data generation parameters.


DB Extract for DB2
Create database backups in the form of SQL scripts, save your database structure and table data as a whole or partially.


SQL Query for DB2
Analyze and retrieve your data, build your queries visually, work with query plans, build charts

based on retrieved data quickly and more.

Scroll to top

Tools & components

Advanced Data Export for RAD Studio VCL
Advanced Data Export for RAD Studio VCL allows you to save your data in the most popular office programs formats.

Advanced Data Export .NET
Advanced Data Export .NET is a component for Microsoft Visual Studio .NET that will allow you to save your data in the most popular data formats for the future viewing, modification, printing or web publication. You can export data into MS Access, MS Excel, MS Word (RTF), PDF, TXT, DBF, CSV and more! There will be no need to waste your time on tiresome data conversion - Advanced Data Export will do the task quickly and will give the result in the desired format.

Advanced Data Import for RAD Studio VCL
Advanced Data Import for RAD Studio VCL will allow you to import your data to the database from files in the most popular data formats.

Advanced PDF Generator for RAD Studio
Advanced PDF Generator for RAD Studio gives you an opportunity to create PDF documents with your applications written on Delphi or C++ Builder.

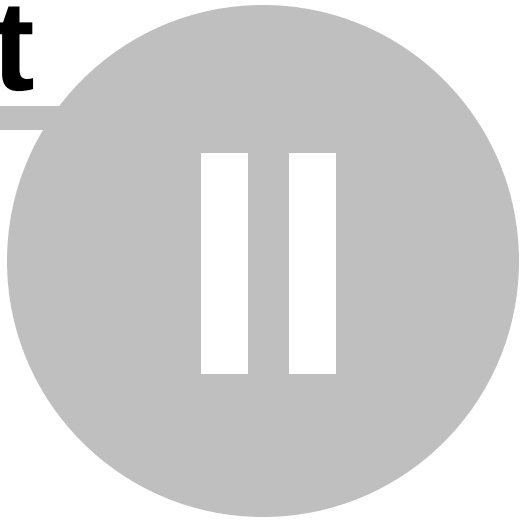
Advanced Query Builder for RAD Studio VCL
Advanced Query Builder for RAD Studio VCL is a powerful component for Delphi and C++ Builder intended for visual building SQL statements for the SELECT, INSERT, UPDATE and DELETE clauses.

Advanced Excel Report for RAD Studio
Advanced Excel Report for RAD Studio is a powerful band-oriented generator of template-based reports in MS Excel.

Advanced Localizer for RAD Studio VCL
Advanced Localizer for RAD Studio VCL is an indispensable component for Delphi for adding multilingual support to your applications.

Scroll to top

Part



2 Advanced Data Export

EMS Advanced Data Export for RAD Studio VCL represents a set of tools for exporting data from any Command objects, DataTable and ListView descendants to different popular formats, such as Microsoft Excel, RTF, HTML, PDF, LaTeX, CSV, DIFF, SYLK, Plain text, Windows Clipboard, XML, DBF, SQL and Microsoft Access format. During the export process none of the mechanisms of cooperation between applications in Microsoft Windows environment (DDE, OLE) is used (except Microsoft Access export), which ensures an extremely high speed of work.

From a programmer's point of view the suite represents a homomorphic hierarchy of classes with the common ancestor TQExport4. Beside the basic properties, methods and events, some specific characteristics are included in descendant classes.

The TQExport4Dialog component allows you to define all the export settings and export your data to any of available formats within a dialog window. Using TQExport4Dialog you can add all the functionality of **Advanced Data Export for RAD Studio VCL** to your application within a single line of code!

Advanced Data Export for RAD Studio VCL provides a collection of the following components:

Component	Brief description
TQExport4	Common data export component
TADO_QExport4Access	Provides data export to MS Access
TQExport4ASCII	Provides data export to CSV, DIFF, SYLK, Plain text formats
TQExport4DBF	Provides data export to DBF Format
TQExport4Clipboard	Provides data export to Windows Clipboard
TQECustomSource	Provides data export from any custom source
TQExport4Dialog	Provides data export to all formats within a dialog window
TQExport4Docx	Provides data export to MS Word format
TQExport4HTML	Provides data export to HTML format (compatible with HTML 4.0 specification)
TQExport4LaTeX	Provides data export to LaTeX format versions 2.09 and 2e
TQExport4ODS	Provides data export to OpenDocument Spreadsheet format
TQExport4ODT	Provides data export to OpenDocument Text format
TQExport4PDF	Provides data export to PDF format
TQExport4RTF	Provides data export to RTF format (compatible with Microsoft Word)
TQExport4SQL	Provides data export to SQL script file (as a set of INSERT statements)
TQExport4XLS	Provides data export to MS Excel 97-2003 format
TQExport4Xlsx	Provides data export to MS Excel format
TQExport4XML	Provides data export to XML format

2.1 TQExport4

2.1.1 TQExport4 Reference

Unit
QExport4

Description
The *TQExport4* class is the basic class of the collection. In this class the basic properties and events for all the descendant classes are determined, and so are the two basic methods - Execute and Abort - which are later overridden in each of the components.

2.1.2 Properties

Run-time only Key properties

- _Version
- Aborted
- About
- AllowCaptions
- AutoCalcColWidth
- AutoCalcStrType
- CaptionRow
- Captions
- ColumnsAlign
- ColumnsWidth
- CurrentRecordOnly
- CustomSource
- DataSet
- DBGrid
- ExportedFields
- ExportEmpty
- ExportRecCount
- ExportSource
- Footer
- Formats
- GoToFirstRecord
- Header
- ListView
- OnlyVisibleFields
- SkipRecCount
- StringGrid
- UserFormats

2.1.2.1 _Version

```
property _Version: string;
```

Description
The *_Version* property is used in all TQExport4 descendant properties and shows the number of the current *Advanced Data Export* version. If you click to change it, you will see the 'About' window with more detailed information.

See also:
About property

2.1.2.2 Aborted

```
property Aborted: boolean;
```

Description
The *Aborted* property shows if method Abort was used. This property is read-only.

See also:
Abort method

2.1.2.3 About

```
property About: string;
```

Description
The *About* property is used in all TQExport4 descendant components and contains information about the current *Advanced Data Export* version and the contact information of its developers. Click the property button to display the 'About' window.

See also:
Version property

2.1.2.4 AllowCaptions

```
property AllowCaptions: boolean;
```

Description
If *AllowCaptions* is True, then Captions are visible in the result file.

See also:
Captions property

2.1.2.5 AutoCalcColWidth

```
property AutoCalcColWidth: boolean;
```

Description
The *AutoCalcColWidth* property is used in TQExport4HTML, TQExport4XLS, TQExport4Xlsx, TQExport4RTF and TQExport4ASCII components. If this property is true, then width of each column in the result file is set automatically depending on the maximum number of symbols in the column cells.

See also:
AutoCalcStrType property

2.1.2.6 AutoCalcStrType

```
property AutoCalcStrType: boolean;
```

Description

The *AutoCalcStrType* property works only if ExportSource = esListView or esStringGrid. If this property is true, then after exporting the first record of the source column *Advanced Data Export* tries to determine the data type of the column, and pass this type to the result file. E.g. if the first record value is '1234.78', and AutoCalcStrType = True, then all the column is treated as Float. It makes sense if all the values of the column have the same format, and you export data, e.g. to MS Excel.

2.1.2.7 CaptionRow

```
property CaptionRow: integer;
```

Description
Property *CaptionRow* is used only when ExportSource is *esStringGrid*. It indicates the number of the row, containing captions for the result columns. The default value is -1, that means that property is not applied. To set the first row of the string grid as caption row, set *CaptionRow* to 0, to use second row, set *CaptionRow* to 1, etc.

See also:
Captions property

2.1.2.8 Captions

```
property Captions: TStrings;
```

Description
The *Captions* property is used in all TQExport4 descendant components, except TQExport4DBF and TQExport4SQL, and determines the column titles in the exported file. The name for each field is set by a separate string which looks as follows

```
<field_name>=<column_title>
```

where *<field_name>* is the name of one of the source fields, and *<column_title>* is the text string without such restrictions as quotation marks and apostrophes. The <field_name> is case-insensitive, and the spaces around the mark of equality are not taken into consideration. If the corresponding string for the field is not specified, the the default caption will be used as a column title (e.g. property Field.DisplayLabel, if ExportSource=esDataSet, or Column.Caption, if ExportSource=esListView). The Captions property works only if *AllowCaptions* is true.

See also:
ExportedFields property
CaptionRow property

2.1.2.9 ColumnsAlign

```
property ColumnsAlign: TStrings;
```

Description
Property *ColumnsAlign* is used in TQExport4HTML, TQExport4RTF and TQExport4ASCII (if ExportType=etTXT), and defines the alignment of the exported cells. Set the alignment in the following format: <Column_Name>=Left|Center|Right, e.g. if COLUMN_1=Center, then column COLUMN_1 will aligned center.

See also:
ColumnsWidth property

2.1.2.10 ColumnsWidth

```
property ColumnsWidth: TStrings;
```

Description
Property *ColumnsWidth* is used in TQExport4XLS, TQExport4RTF and TQExport4ASCII (if ExportType=etTXT), and allows you to set the width of the result table columns in the following way: <field_name>=<value>, where <field_name> is the source field name, and <value> is the corresponding column width in symbols.

See also:
ColumnsAlign property

2.1.2.11 CurrentRecordOnly

```
property CurrentRecordsOnly: boolean;
```

Description
If the *CurrentRecordOnly* property is true, then only one record is exported from the DataSet - current records at the moment of the export start (properties GoToFirstRecord and SkipRecCount are taken into consideration). When the export finishes, current DataSet record doesn't change.

See also:
GoToFirstRecord property
SkipRecCount property

2.1.2.12 CustomSource

```
property CustomSource: TqeCustomSource;
```

Description
The *CustomSource* property allows you to export data from any source you choose. This property contains properties, methods and events which describe operations with any data source.

See also:
TqeCustomSource object

2.1.2.13 DataSet

```
property DataSet: TDataSet;
```

Description
Use the *Dataset* property to determine the exported dataset, if the ExportSource property is esDataSet. The dataset must be opened (Dataset.Active = true) before the export begins (before the Execute method is invoked).

See also:
DBGrid property
ListView property
StringGrid property
ExportSource property

2.1.2.14 DBGrid

```
property DBGrid: TDBGrid;
```

Description
Use the *DBGrid* property to determine the exported database grid, if the ExportSource property is esDBGrid. The DataSource property must be defined and the source dataset must be opened (Dataset.Active = true) before the export begins (before the Execute method is invoked).

See also:
DataSet property
ListView property
StringGrid property
ExportSource property

2.1.2.15 ExportedFields

```
property ExportedFields: TStrings;
```

Description
The *ExportedFields* property is used in all the TQExport4 descendant components and contains the list of the exported fields. If this list is empty (as default), then all the table fields will be exported, except BLOB ones. You can also edit the list of the exported fields by right-clicking the component and choosing "Fields editor..." from the popup menu.

See also:
Captions property

2.1.2.16 ExportEmpty

```
property ExportEmpty: boolean;
```

Description
If property *ExportEmpty* is true then no error occurs even if the exported dataset is empty. In this case the proper component correctly starts and finishes export without writing a single record to the destination file.

2.1.2.17 ExportRecCount

```
property ExportRecCount: integer;
```

Description
The *ExportRecCount* property determines the number of records, exported from the source table. If ExportRecCount = 0, then all the records are exported.

See also:
SkipRecCount property
OnExportedRecord

2.1.2.18 ExportSource

```
type
  TQExportSource = (esDataSet, esListView, esDBGrid, esStringGrid, esCustom);

property ExportSource: TQExportSource;
```

Description
The *ExportSource* property determines the data source for export. The following values are available: esDataset - export from DataSet, esDBGrid - export from DBGrid, esListView - export from ListView, esStringGrid - export from StringGrid. Use TQExport properties to determine the instanse of the selected source component: DataSet, DBGrid, ListView or StringGrid in accordance. The esCustom value allows you to export data from any source you want. See the CustomSource property for details.

See also:
CustomSource property
DataSet property
DBGrid property
ListView property
StringGrid property

2.1.2.19 Footer

```
property Footer: TStrings;
```

Description
The *Footer* property is used in all the descendant components, except TQExport4XML and TQExport4DBF, and can include a text that will be placed in the export file (FileName) directly *after* the exported data.

See also:
Header property

2.1.2.20 Formats

```
property Formats: TQExportFormats;
```

Description

The complex property *Formats* is used in all the descendant components, except TQExport4DBF, and determines the type of the exported data of the types most frequently used, for which a formatted output might be needed. For the representation of each of these types there exist the string properties IntegerFormat, FloatFormat, CurrencyFormat and DateTimeFormat which have the settings specified in the operation system by default. There are also properties for representation of Boolean and Null values. The settings of these properties will be applied to all source fields of the corresponding types. To set a special format for a separate field (fields), use UserFormats property.

2.1.2.21 GoToFirstRecord

```
property GoToFirstRecord: boolean;
```

Description

The *GoToFirstRecord* property is used in all TQExport4 descendant components. If this property is true then the export is started from the first table record, otherwise from the current one.

2.1.2.22 Header

```
property Header: TStrings;
```

Description
The *Header* property is used in all the descendant components, except TQExport4XML and TQExport4DBF, and can include a text that will be placed in the export file (FileName) directly *before* the exported data.

See also:
Footer property

2.1.2.23 ListView

```
property ListView: TListView;
```

Description
Use *ListView* property to determine the exported list view, if the ExportSource property is esListView.

See also:
DataSet property
DBGrid property
StringGrid property
ExportSource property

2.1.2.24 OnlyVisibleFields

```
property OnlyVisibleFields: boolean;
```

Description

The *OnlyVisibleFields* property is used in all the TQExport4 descendant components. If OnlyVisibleFields is true, and ExportedFields is empty, then all the fields with Visible=True will be exported (except BLOBs). If OnlyVisibleFields is true, and ExportedFields is not empty, then only fields, included into ExportedFields, with Visible=True will be exported.

2.1.2.25 SkipRecCount

```
property SkipRecCount: integer;
```

Description
The *SkipRecCount* property determines the number of records which are not exported. If SkipRecCount = 0, then all the records are exported.

See also:
ExportRecCount property
OnSkippedRecord event

2.1.2.26 StringGrid

```
property StringGrid: TStringGrid;
```

Description
Use *StringGrid* property to determine the exported string grid, if the ExportSource property is esStringGrid.

See also:
DataSet property
DBGrid property
ListView property
ExportSource property

2.1.2.27 UserFormats

```
property UserFormats: TStrings;
```

Description
The *UserFormats* property is used in all TQExport4 descendant components, except TQExport4DBF and TQExport4SQL, and allows to set a special output format for any source field. At the same time, the setting of the strings being a part of the property is set in the following way

```
<field_name>=<output_format>
```

where *<field_name>* is the name of one of the source fields, and *<output_format>* is the output format for this field. For more details, see the Delphi/C++ Builder documentation on FormatFloat, FormatCurrency, FormatDateTime functions. The <field_name> is case-insensitive. Note that there must not be any spaces to the left and to the right of the equality sign.

Use UserFormats property in this and only in this case when you need to set different output formats for two and more fields of the same type, otherwise Formats property should be used.

Note: If you want to add measurement units or any other comment string to format string on export to XLS you need to keep in mind that symbols "d", "e, "h", "m", "s" and "y" are used for formatting purposes. If you need to use these symbols in the format string, add back slash before the symbol (e.g. "", "").

Example:

```
field1 = ###.# kilo
field2 = ##
```

See also:
Formats property

2.1.3 Methods

Key methods

- Abort
- Execute
- ExportToStream
- LoadPropertiesFromFile
- SavePropertiesToFile

2.1.3.1 Abort

```
procedure Abort; virtual;
```

Description
The *Abort* method stops the current process of export initiated by the Execute method invocation. At the same time, the OnStopExport event takes place.

See also:
Aborted property
Execute method
OnStopExport event

2.1.3.2 Execute

```
procedure Execute; virtual;
```

Description
The *Execute* method is the central method of the collection. It executes the export of source data to file. Of course, for each descendant class the method is defined in its own way, but the logics of the work remains the same for all the components. In case of the invocation, the function checks if the FileName and ExportSource (and the corresponding DataSet, DBGrid, ListView, StringGrid or CustomSource) properties are correct; in case of an error, the exceptions are raised and the method stops executing, returning false as a result. If all the required properties are set correctly, the method starts its work with generating the OnBeginExport event. Then the data export to file begins, and after the export of each record the event OnExportedRecord event takes place. On completion of the process, the event OnEndExport is invoked. The export can be interrupted by Abort method invocation, in this case the OnStopExport event handler is invoked. At the same time, it is guaranteed that even in case of export interruption the output file will still exist in a readable form for the corresponding software product.

See also:
Abort method
ExportToStream method
OnBeginExport event

2.1.3.3 ExportToStream

```
procedure ExportToStream(AStream: TStream);
```

Description
Use *ExportToStream* to export data to the stream specified by the *AStream* parameter. This can be any *TStream* descendant. This method works in all the QExport4 descendants, except TQExport4XLS.

See also:
Execute method

2.1.3.4 LoadPropertiesFromFile

```
procedure LoadPropertiesFromFile(const FileName: string);
```

Description
Use this method to restore object properties from file defined by the *FileName* variable previously saved with the SavePropertiesToFile method.

2.1.3.5 SavePropertiesToFile

```
procedure SavePropertiesToFile(const FileName: string);
```

Description
Use this method to save all object properties to file defined by the *FileName* variable. You can restore previously saved properties by calling the LoadPropertiesFromFile method.

2.1.4 Events

Key events

- OnBeforeExportRow
- OnBeginExport
- OnEndExport
- OnExportedRecord
- OnGetCellParams
- OnGetExportText
- OnSkippedRecord
- OnStopExport

2.1.4.1 OnBeforeExportRow

```
type
  TBeforeExportRowEvent = procedure(Sender: TObject; Row: TQExportRow; var Accept: boo

property OnGetCellParams: TBeforeExportRowEvent;
```

Description

The *OnBeforeExportRow* event allows you to manage data during the export process. The *Row* parameter allows you to access data in the current row, and the *Accept* parameter allows you to export the row or not. If *Accept* is set to *true*, the row will be exported.

2.1.4.2 OnBeginExport

```
property OnBeginExport: TNotifyEvent;
```

Description
The *OnBeginExport* event takes place directly before the export beginning.

See also:
Execute method
OnEndExport event
OnStopExport event

2.1.4.3 OnEndExport

```
property OnEndExport: TNotifyEvent;
```

Description
The *OnEndExport* event takes place when the export is complete if it wasn't interrupted by the Abort method. Otherwise the event OnStopExport is invoked.

See also:
OnBeginExport event
OnStopExport event

2.1.4.4 OnExportedRecord

```
property OnExportedRecord: TExportedRecordEvent;
```

Description
The *OnExportedRecord* event takes place after the export of each source record. It is most frequently used to transfer to the user the information on the export execution process, such as: display of the number of records exported, the increase of the *ProgressBar* position, etc.

See also:
ExportRecCount property
OnSkippedRecord event
TExportedRecordEvent type

2.1.4.5 OnGetCellParams event

```
property OnGetCellParams: TGetCellParamsEvent;
```

Description
The *OnGetCellParams* event takes place after the parameters of the record (including record value) are received. Depending on the record and column number, and on the value, you can change the format of the result table cell: alignment, font, and background color.

See also:
OnGetExportText event
TGetCellParamsEvent type

2.1.4.6 OnGetExportText

```
property OnGetExportText: TGetExportTextEvent;
```

Description
This event takes place after the export of each source string. Using this event you can replace some definite text of the exported table with another text.

See also:
OnGetCellParams event
TGetExportTextEvent type

2.1.4.7 OnSkippedRecord

```
property OnSkippedRecord: TExportedRecordEvent;
```

Description
The *OnSkippedRecord* event takes place when one of the first source records is skipped. The number of records to skip is determined by the SkipRecCount property.

See also:
SkipRecCount property
OnExportedRecord event
TExportedRecordEvent type

2.1.4.8 OnStopExport

```
property OnStopExport: TQExportStopEvent;
```

Description
The *OnStopExport* event takes place when the export process begun by the Execute method is interrupted with the Abort method invocation.

See also:
Abort method
OnBeginExport event
OnEndExport event
TQExportStopEvent type

2.2 TADO_QExport4Access

2.2.1 TADO_QExport4Access Reference

Unit
ADO_QExport4Access

Description
The *TADO_QExport4Access* component is intended for exporting data to MS Access format trough an ADO connection. To work with this component you should have ADO installed onto your system.

2.2.2 Properties

Run-time only Key properties

- AppendDateTimeToDatabaseName
- AutoCreateDatabase
- AutoCreateTable
- DatabaseName
- ExportedDatabaseName
- FileVersion
- PrintFile
- ShowFile
- TableName

2.2.2.1 AppendDateTimeToDatabaseName

```
property AppendDateTimeToDatabaseName : boolean;
```

Description
Setting the *AppendDateTimeToDatabaseName* property to *True* adds the export date and time to the name of the result Access database file.

See also:
ExportedDatabaseName property

2.2.2.2 AutoCreateDatabase

```
property AutoCreateDatabase : boolean;
```

Description
Setting the *AutoCreateDatabase* property to *True* creates the result Access database automatically in case it does not exists.

See also:
AutoCreateTable property

2.2.2.3 AutoCreateTable

```
property AutoCreateTable : boolean;
```

Description
Setting the *AutoCreateTable* property to *True* creates the result Access table automatically in case it does not exists.

See also:
AutoCreateDatabase property

2.2.2.4 DatabaseName

```
property DatabaseName : string;
```

Description
The *DatabaseName* property defines the name of the result Access file to export data to.

See also:
TableName property
ExportedDatabaseName property

2.2.2.5 ExportedDatabaseName

```
property ExportedDatabaseName : string;
```

Description
The *ExportedDatabaseName* property defines the result Access database file name considering the value of the AppendDateTimeToDatabaseName property. If this property is set to *True* then the result database name consists of the value defined in the DatabaseName property and exported date and time.

See also:
AppendDateTimeToDatabaseName property
DatabaseName property

2.2.2.6 FileVersion

```
property FileVersion : TAccessFileVersion;
TAccessFileVersion = (afvAccess2003, afvAccess2007);
```

Description

The *FileVersion* property defines the format of the result file: Access2003 (*.mdb) or Access2007 (*.accdb).

2.2.2.7 PrintFile

```
property PrintFile : boolean;
```

Description
Setting the *PrintFile* property to *True* sends the result Access file to printing after the export is finished.

See also:
ShowFile property

2.2.2.8 ShowFile

```
property ShowFile : boolean;
```

Description
Setting the *ShowFile* property to *True* opens the result file in MS Access after the export is finished.

See also:
PrintFile property

2.2.2.9 TableName

```
property TableName;
```

Description
The *TableName* property defines the name of the result Access table to export data to.

See also:
DatabaseName property

2.3 TQExport4ASCII

2.3.1 TQExport4ASCII Reference

Unit
QExport4ASCII

Description
The *TQExport4ASCII* component is used for the data export to formats that are usually used as working or interchange formats, i.e. Comma Separated Value (CSV), Data Interchange File Format (DIFF), Symbolic Links (SYLK) and Plain Text Format.
The output file format is set by the ExportType property.

2.3.2 Properties

Run-time only Key properties

AutoCalcColWidth
ColumnsAlign
ColumnsWidth
CSVComma
CSVQuote
CSVQuoteStrings
ExportType
TXTSpacing

2.3.2.1 CSVComma

```
property CSVComma: char;
```

Description
The *CSVComma* property is used only during the export to *.csv* format (ExportType = etCSV) and is created to install the symbol-delimiter in the result file. The default property value is the current Windows list separator.

See also:
CSVQuoteStrings property

2.3.2.2 CSVQuote

```
property CSVQuote: char;
```

Description
The *CSVQuote* property is used only during the export to *.csv* format (ExportType = etCSV) and is created to install the quotation symbol in the result file. This symbol is used only if the CSVQuoteStrings property is true. The default property value is ".

See also:
CSVQuoteStrings property

2.3.2.3 CSVQuoteStrings

```
property CSVQuoteStrings: boolean;
```

Description

The *CSVQuoteStrings* property is used only during the export to *.csv* format (ExportType = etCSV). If this property is true, then all the source strings will be exported as quotations, and the apostrophes will be doubled.

For example, string

QuickExport is a nice set of components, isn't it?

is exported as

'QuickExport is a nice set of components, isn''t it?'

See also:

CSVComma property

CSVQuote property

2.3.2.4 ExportType

```
property ExportType: TQExport4ASCIIType;
```

Description
The *ExportType* property sets the format of the output export file. The relations between the property setting and the file type are as follows:

Property setting	File type	File extension
etCSV	Comma Separated Values	*.csv*
etDIF	Data Interchange File Format	*.dif*
etSYLK	Symbolic Links Format	*.slk*
etTxt	Plain Text Format	*.txt*

2.3.2.5 TXTSpacing

```
property TXTSpacing: integer;
```

Description

The *TXTSpacing* property sets the distance (in symbols) between the data columns in the result file. The default value is 1.

2.3.3 Methods

Key methods

Abort

2.4 TQExport4DBF

2.4.1 TQExport4DBF Reference

Unit
QExport4DBF

Description
The *TQExport4DBF* component allows you to export your data to the DBF format. The result file can be opened in Delphi Database Desktop or in any database application.

2.4.2 Properties

Run-time only Key properties

- ColumnsPrecision
- DefaultFloatSize
- DefaultFloatDecimal
- NullValue

2.4.2.1 ColumnsPrecision

```
property ColumnsPrecision: TStrings;
```

Description
The *ColumnsPrecision* property determines the column precisions in the exported file. The name for each field is set by a separate string which looks as follows

<field_name>=<column_precision>

where *<field_name>* is the name of one of the source fields, and *<column_precision>* is the column precision. The *<field_name>* is case-insensitive, and the spaces around the mark of equality are not taken into consideration.

See also:
DefaultFloatSize property
DefaultFloatDecimal property

2.4.2.2 DefaultFloatSize

```
property DefaultFloatSize: integer;
```

Description
The *DefaultFloatSize* property defines the default size for float fields. This value is used when float fields has no defined size.

See also:
ColumnsPrecision property
DefaultFloatDecimal property

2.4.2.3 DefaultFloatDigital

```
property DefaultFloatDecimal: integer;
```

Description
The *DefaultFloatDecimal* property defines the default size for the fractional part of float fields. This value is used when float fields has no defined size of the fractional part.

See also:
ColumnsPrecision property
DefaultFloatSize property

2.4.2.4 NullValue

```
property NullValue: string;
```

Description
This property contains the string value that is to be inserted to the output DBF file if record has the fields with NULL values.

See also:
ColumnsPrecision property
DefaultFloatDecimal property

2.5 TQExport4Clipboard

2.5.1 TQExport4Clipboard Reference

Unit
QExport4Clipboard

Description
Use *TQExport4Clipboard* component to export your data to the Windows clipboard. Two export formats are available in this component. See ExportType property for details.

2.5.2 Properties

▸ Run-time only Key properties

- AllowCaptions
- Captions
- ClipboardViewer
- ExportType
- Footer
- Formats
- Header
- Separator
- Spacing
- UserFormats

2.5.2.1 ClipboardViewer

```
property ClipboardViewer: string;
```

Description

The *ClipboardViewer* property determines the clipboard-viewer application to display your data after export. The default executable file is *clipbrd.exe*, located at C:\ drive. You can change it, if necessary. This property is used only if property ShowFile = True.

2.5.2.2 ExportType

```
property ExportType: TQClipboardExportType;
```

Description
Use this property to select the way of separating columns in the exported table. If *ExportType=etFixed* then each column of the exported table will have its own fixed size, defined by the Spacing property. If *ExportType=etSeparated* then the columns of the table will be separated by a certain character defined by the Separator property.

See also:
Separator property
Spacing property

2.5.2.3 Separator

```
property Separator: Char;
```

Description
This property determines the character used for separating the exported table columns if property ExportType = etSeparated. The default value is the current Windows list separator.

See also:
ExportType property
Spacing property

2.5.2.4 Spacing

```
property Spacing: Integer;
```

Description
This property determines the interval between the exported table columns, if property ExportType = etFixed. The default interval is 2 'space' characters.

See also:
ExportType property
Separator property

2.6 TQECustomSource

2.6.1 TQECustomSource Reference

Unit
QExport4CustomSource

Description
The *TqeCustomSource* object contains properties, methods and events which allow you to export data from any custom source. To export data from custom source, you need to define the OnGetColumnValue and OnGetNextRecord event handlers.

2.6.2 Properties

Run-time only Key properties

- ColCount
- Columns
- Eof
- RecNo

2.6.2.1 ColCount

```
property ColCount: integer;
```

Description
The *ColCount* property returns the columns count in the custom source. The property value is equivalent to the *Columns.Count* value. This property is read-only.

See also:
Columns property

2.6.2.2 Columns

```
property Columns: TqeCustomColumns;
```

Description
The *Columns* property is a collection of TqeCustomColumn objects which describe the columns in the custom source. Use this property to access the culumns' parameters by accessing each object in this collection and setting its parameters.

See also:
TqeCustomColumns object
TqeCustomColumn object

2.6.2.3 Eof

```
property Eof: boolean;
```

Description

The *Eof* property indicates whether the end of export source is reached or not. If its value is True, the TQExport4 class or its descendant which owns this object, finishes the export process. The OnGetNextRecord event sets the property value to the value, returned after the event execution.

2.6.2.4 RecNo

```
property RecNo: integer;
```

Description
The *RecNo* property returns the current record number. This property is read-only.

2.6.3 Methods

Key methods

- ColumnByName
- First
- Next

2.6.3.1 ColumnByName

```
function ColumnByName(const ColumnName: string): TqeCustomColumn;
```

Description
This method returns the TqeCustomColumn object which corresponds to the column with name equivalent to *ColumnName*.

See also:
TqeCustomColumn object

2.6.3.2 First

```
procedure First;
```

Description
The *First* method sets the RecNo property to 1.

2.6.3.3 Next

```
procedure Next;
```

Description
The *Next* method moves the export along the data source. It calls the *OnGetNextRecord* event, increases the *RecNo* property value, and sets the *Eof* property to *True* if the event handler returned *True*.

2.6.4 Events

Key events

- OnGetColumnValue
- OnGetNextRecord

2.6.4.1 OnGetColumnValue

```
type
  TGetColumnValueEvent = procedure(Sender: TObject; RecNo: integer; Column: TqeCustomC
of object;

property OnGetColumnValue: TGetColumnValueEvent;
```

Description

The *OnGetColumnValue* event returns Value which is the value of the record by its number and column which are defined by the *RecNo* and *Column* parameters. Use this event to export data from the custom data source by selecting values depending on the column and current record number.

2.6.4.2 OnGetNextRecord

```
type
  TGetNextRecordEvent = procedure(Sender: TObject; RecNo: integer; var Eof: boolean) o

property OnGetNextRecord: TGetNextRecordEvent;
```

Description
Use this event to determine when the end of export source is reached. If the *Eof* variable is true after the event execution, the export process stops.

2.7 TQExport4Dialog

2.7.1 TQExport4Dialog Reference

Unit
QExport4Dialog

Description
The *TQExport4Dialog* component allows you to define all the export settings and export your data to any of available formats within a dialog window.

2.7.2 Properties

Run-time only Key properties

- _Version
- About
- AllowCaptions
- AllowedExports
- AutoCalcStrType
- AutoChangeFileExt
- AutoLoadOptions
- AutoSaveOptions
- Captions
- ColumnsAlign
- ColumnsWidth
- CommonOptions
- ConfirmAbort
- CSVOptions
- DataSet
- DBGrid
- ExportedFields
- ExportRecCount
- ExportSource
- FileName
- Footer
- Formats
- GoToFirstRecord
- Header
- HTMLMultiFileOptions
- HTMLPageOptions
- HTMLTableOptions
- ListView
- OnlyVisibleFields
- OptionsFileName
- PDFOptions
- PrintFile
- RTFOptions
- SaveLoadButtons
- ShowFile
- SkipRecCount
- SQLOptions
- StringGrid
- TXTOptions
- UserFormats
- XLSOptions
- XMLOptions

2.7.2.1 _Version

```
property _Version: string;
```

Description
The *_Version* property shows the number of the current *Advanced Data Export* version. If you click to change it you will see the 'About' window with more detailed information.

See also:
About property

2.7.2.2 About

```
property About: string;
```

Description
The *About* property contains information about the current *Advanced Data Export* version and the contact information of its developers. Click the property button to display the 'About' window.

See also:
Version property

2.7.2.3 AllowCaptions

```
property AllowCaptions: boolean;
```

Description
If *AllowCaptions* is true, then Captions will be visible in the result file.

See also:
Captions property

2.7.2.4 AllowedExports

```
property AllowedExports: TAllowedExports;
```

Description
The *AllowedExports* property defines the export types available in the dialog window. It is a set of boolean subproperties each of which corresponds to a certain export type. The following subproperties are availavble in *TAllowedExports*:
aeXLS
aeWord
aeRTF
aeHTML
aeXML
aeDBF
aeTXT
aeCSV
aeDIFF
aeSylk
aeLaTeX
aeSQL
aeClipboard

All the subproperties are default true. If you don't want any of these types to appear in the dialog window set its subproperty to false.

See also:
AutoChangeFileExt property
CommonOptions property

2.7.2.5 AutoCalcStrType

```
property AutoCalcStrType: boolean;
```

Description

The *AutoCalcStrType* property works only if ExportSource = esListView or esStringGrid. If this property is true, then after exporting the first record of the source column *Advanced Data Export* tries to determine the data type of the column, and pass this type to the result file. E.g. if the first record value is '1234.78', and AutoCalcStrType = True, then all the column is treated as Float. It makes sense if all the values of the column have the same format, and you export data, e.g. to MS Excel.

2.7.2.6 AutoChangeFileExt

```
property AutoChangeFileExt: boolean;
```

Description
If the *AutoChangeFileExt* property is true then on choosing the export type in the dialog window the file extension will be changed automatically in the 'Destination file' edit field.

See also:
AllowedExports property
FileName property

2.7.2.7 AutoLoadOptions

```
property AutoLoadOptions: boolean;
```

Description
If the *AutoLoadOptions* property is true then the export settings are automatically loaded from the default **.cfg* file. To create this file your should enable the AutoSaveOptions property or use the button 'Save Export Options' in the dialog window and save export options to the default file at least once. The file is defined in the OptionsFileName property.

See also:
AutoSaveOptions property
OptionsFileName property
SaveLoadButtons property

2.7.2.8 AutoSaveOptions

```
property AutoSaveOptions: boolean;
```

Description
If the *AutoSaveOptions* property is true then the export options you set in the dialog window are automatically saved to the default *.cfg file. The file is defined in the OptionsFileName property.

See also:
AutoLoadOptions property
OptionsFileName property
SaveLoadButtons property

2.7.2.9 Captions

```
property Captions: TStrings;
```

Description
The *Captions* property determines the column titles in the exported file (except DBF and SQL). The name for each field is set by a separate string which looks as follows:

<field_name>=<column_title>

where *<field_name>* is the name of one of the source fields, and *<column_title>* is the text string without such restrictions as quotation marks and apostrophes. The <field_name> is case-insensitive, and the spaces around the mark of equality are not taken into consideration. If the corresponding string for the field is not specified, the the default caption will be used as a column title (e.g. property Field.DisplayLabel, if ExportSource=esDataSet, or Column.Caption, if ExportSource=esListView). The Captions property works only if AllowCaptions is true.

See also:
AllowCaptions property
ExportedFields property

2.7.2.10 ColumnsAlign

```
property ColumnsAlign: TStrings;
```

Description
Property *ColumnsAlign* is used in export to HTML, Word, RTF, TXT and defines the alignment of the exported cells. Set the alignment in the following format: <Column_Name>=Left|Center|Right, e.g. if COLUMN_1=Center, then column COLUMN_1 will aligned center.

See also:
ColumnsWidth property

2.7.2.11 ColumnsWidth

```
property ColumnsWidth: TStrings;
```

Description
Property *ColumnsWidth* is used in export to Excel, Word, RTF, TXT and allows you to set the width of the result table columns in the following way: <field_name>=<value>, where <field_name> is the source field name, and <value> is the corresponding column width in symbols.

See also:
ColumnsAlign property

2.7.2.12 CommonOptions

```
property CommonOptions: TCommonOptions;
```

Description

The *CommonOptions* property defines the option tabs available in the dialog window for all export types. It is a set of boolean subproperties each of which corresponds to the certain option tab. The following subproperties are available in TCommonOptions: coFields, coFormats, coColons, coCaptions, coOptions. All the subproperties are default true. If you don't want any of these tabs to appear in the dialog window set its subproperty to false.

See also:

OptionsFileName property
AllowedExports property

2.7.2.13 ConfirmAbort

```
property ConfirmAbort: boolean;
```

Description
If the *ConfirmAbort* property is true then on aborting the export process will requre confirmation. Note that aborting export is possible only if export size is large enough.

2.7.2.14 CSVOptions

```
property CSVOptions: TQExportCSVOptions;
```

Description
The *CSVOptions* property allows you to define all the CSV export options in the design-time. The properties of the *TQExportCSVOptions* correspond to the properties of the TQExportASCII.

2.7.2.15 DataSet

```
property DataSet: TDataSet;
```

Description
Use the *Dataset* property to determine the exported dataset, if the ExportSource property is esDataSet. The dataset must be opened (Dataset.Active = true) before the export begins (before the Execute method is invoked).

See also:
DBGrid property
ListView property
StringGrid property
ExportSource property

2.7.2.16 DBGrid

```
property DBGrid: TDBGrid;
```

Description
Use the *DBGrid* property to determine the exported database grid, if the ExportSource property is *esDBGrid*. The *DataSource* property must be defined and the source dataset must be opened (*Dataset.Active = true*) before the export begins (before the Execute method is invoked).

See also:
DataSet property
ListView property
StringGrid property
ExportSource property

2.7.2.17 ExportedFields

```
property ExportedFields: TStrings;
```

Description
The *ExportedFields* property contains the list of the exported fields. If this list is empty (as default), then all the table fields will be exported, except BLOB ones. You can also edit the list of the exported fields by right-clicking the component and choosing "Fields editor..." from the popup menu.

See also:
Captions property

2.7.2.18 ExportRecCount

```
property ExportRecCount: integer;
```

Description
The *ExportRecCount* property determines the number of records, exported from the source table. If *ExportRecCount = 0*, then all the records are exported.

See also:
SkipRecCount property
OnExportedRecord event

2.7.2.19 ExportSource

```
property ExportSource: TQExportSource;
```

Description
The *ExportSource* property determines the data source for export. The following values are available:
esDataset - export from DataSet
esDBGrid - export from DBGrid
esListView - export from ListView
esStringGrid - export from StringGrid

Use *TQExport4Dialog* properties to determine the instance of the selected source component: DataSet, DBGrid, ListView or StringGrid respectively.

See also:
DataSet property
DBGrid property
ListView property
StringGrid property

2.7.2.20 FileName

```
property FileName: string;
```

Description
The *FileName* property determines the name of the result export file. The property must not be empty; if it is empty in the moment of the Execute method invocation, the component will stop being executed having generated an exception with the message 'Invalid File Name!'.

See also:
ShowFile property
PrintFile property

2.7.2.21 Footer

```
property Footer: TStrings;
```

Description
The *Footer* property can include a text that will be placed in the result file directly *after* the exported data (except XML and DBF).

See also:
Header property
OnGetFooter event

2.7.2.22 Formats

```
property Formats: TQExportFormats;
```

Description
The complex property *Formats* determines the type of the exported data of the types most frequently used, for which a formatted output, such as ftInteger, ftFloat, ftCurrency and ftDateTime might be needed (except DBF and SQL). For the representation of each of these types there exist the string properties IntegerFormat, FloatFormat, CurrencyFormat and DateTimeFormat accordingly, which have the settings specified in the operation system by default. There are also properties for representating boolean and Null values. The settings of these properties will be applied to all source fields of the corresponding types. To set a special format for a separate field (fields), use the UserFormats property.

See also:
UserFormats property
TQExportFormats object

2.7.2.23 GoToFirstRecord

```
property GoToFirstRecord: boolean;
```

Description
If the *GoToFirstRecord* property is *true* then the export is started from the first table record, otherwise from the current one.

2.7.2.24 Header

```
property Header: TStrings;
```

Description
The *Header* property can include a text that will be placed in the result file directly before the exported data (except XML and DBF).

See also:
Footer property
OnGetHeader event

2.7.2.25 HTMLMultiFileOptions

```
property HTMLMultiFileOptions: TQExportHTMLMultiFileOptions;
```

Description

The *HTMLMultiFileOptions* property allows you to tune options of exporting data to several HTML pages in the design time. The properties of the TQExportHTMLMultiFileOptions object correspond to the properties of the TQExport4HTML and TQExportHTMLNavigation.

2.7.2.26 HTMLPageOptions

```
property HTMLPageOptions: TQExportHTMLPageOptions;
```

Description

The *HTMLPageOptions* property allows you to customize each exported HTML page in the design time. The properties of the TQExportHTMLPageOptions object correspond to the properties of the TQExport4HTML and THTMLOptions.

2.7.2.27 HTMLTableOptions

```
property HTMLTableOptions: TQExportHTMLTableOptions;
```

Description

The *HTMLTableOptions* property allows you to customize tables in the result HTML documents in the design time. The properties of the TQExportHTMLTableOptions object correspond to the properties of the TQExport4HTML and TTableOptions.

2.7.2.28 ListView

```
property ListView: TListView;
```

Description
Use *ListView* property to determine the exported list view, if the ExportSource property is *esListView*.

See also:
DataSet property
DBGrid property
StringGrid property
ExportSource property

2.7.2.29 OnlyVisibleFields

```
property OnlyVisibleFields: boolean;
```

Description

If *OnlyVisibleFields* property is true, and ExportedFields is empty, then all the fields with *Visible=True* will be exported (except BLOBs). If *OnlyVisibleFields* is true, and ExportedFields is not empty, then only fields, included into ExportedFields, with *Visible=True* will be exported.

2.7.2.30 OptionsFileName

```
property OptionsFileName: string;
```

Description

The *OptionsFileName* property defines the default filename to save and to load the export options when using the buttons 'Save Export Options' and 'Load Export Options' in the dialog window. This property also defines the filename used when AutoSaveOptions or AutoLoadOptions property is enabled. If you will not define the complete filename Windows will write this file to the directory, that is current, or to C:\ (it depends on the Windows version).

See also:

AutoLoadOptions property
AutoSaveOptions property
SaveLoadButtons property

2.7.2.31 PDFOptions

```
property PDFOptions: TQExportPDFOptions;
```

Description
The *PDFOptions* property allows you to define all the PDF export options in the design-time. The properties of the *TQExportPDFOptions* correspond to the properties of the TPDFOptions.

2.7.2.32 PrintFile

```
property PrintFile: boolean;
```

Description
If *PrintFile* is *true* then the result file will be sent to printing right after export.

See also:
FileName property
ShowFile property

2.7.2.33 RTFOptions

```
property RTFOptions: TQExportRTFOptions;
```

Description
The *RTFOptions* property allows you to define all the RTF export options in the design-time. The properties of the *TQExportRTFOptions* correspond to the properties of the TRTFOptions.

2.7.2.34 SaveLoadButtons

```
property SaveLoadButtons: boolean;
```

Description
If the *SaveLoadButtons* property is true then buttons 'Save Export Options...' and 'Load Export Options' are available in the dialog window. Using these buttons you can save the export options defined in the *SaveOptions* property and load them during the next export session. The default filename to store the export options is defined in the OptionsFileName property.

See also:
AutoLoadOptions property
AutoSaveOptions property
OptionsFileName property

2.7.2.35 ShowFile

```
property ShowFile: boolean;
```

Description
If the *ShowFile* property is true then the result file will be opened in the appropriate program right after export.

See also:
FileName property
PrintFile property

2.7.2.36 SkipRecCount

```
property SkipRecCount: integer;
```

Description
The *SkipRecCount* property determines the number of records which are not exported. If *SkipRecCount = 0*, then all the records are exported.

See also:
ExportRecCount property
OnSkippedRecord event

2.7.2.37 SQLOptions

```
property SQLOptions: TQExportSQLOptions;
```

Description
The *SQLOptions* property allows you to define all the SQL export options in the design-time. The properties of the *TQExportSQLOptions* correspond to the properties of the TQExport4SQL.

2.7.2.38 StringGrid

```
property StringGrid: TStringGrid;
```

Description
Use the *StringGrid* property to determine the exported string grid, if the ExportSource property is *esStringGrid*.

See also:
DataSet property
DBGrid property
ListView property
ExportSource property

2.7.2.39 TXTOptions

```
property TXTOptions: TQExportTXTOptions;
```

Description
The *TXTOptions* property allows you to define all the TXT export options in the design-time. The properties of the *TQExportTXTOptions* correspond to the properties of the TQExport4ASCII.

2.7.2.40 UserFormats

```
property UserFormats: TStrings;
```

Description
The *UserFormats* property allows to set a special output format for any source field (except DBF and SQL). At the same time, the setting of the strings being a part of the property is set in the following way:

<field_name>=<output_format>

where *<field_name>* is the name of one of the source fields, and *<output_format>* is the output format for this field. For more details, see the Delphi/C++ Builder documentation on FormatFloat, FormatCurrency, FormatDateTime functions. The <field_name> is case-insensitive. Note that there must not be any spaces to the left and to the right of the equality sign.

Use the *UserFormats* property in this and only in this case when you need to set different output formats for two and more fields of the same type, otherwise Formats property should be used.

See also:
Formats property

2.7.2.41 XLSOptions

```
property XLSOptions: TQExportXLSOptions;
```

Description
The *XLSOptions* property allows you to define all the XLS export options in the design-time. The properties of the *TQExportXLSOptions* correspond to the properties of the TQExport4XLS and TXLSOptions.

2.7.2.42 XMLOptions

```
property XMLOptions: TQExportXMLOptions;
```

Description
The *XMLOptions* property allows you to define all the XML export options in the design-time. The properties of the *TQExportXMLOptions* correspond to the properties of the TXMLOptions.

2.7.3 Methods

- Key methods
  - Execute

2.7.3.1 Execute

```
procedure Execute;
```

Description
The *Execute* method executes the export of source data to file. In case of the invocation, the function checks if the FileName and ExportSource (and the corresponding DataSet, DBGrid, ListView or StringGrid) properties are correct; in case of an error, the exceptions are raised and the method stops executing, returning false as a result. If all the required properties are set correctly, the method starts its work with generating the OnBeginExport event. Then the data export to file begins, and after the export of each record the event OnExportedRecord event takes place. On completion of the process, the event OnEndExport is invoked.

See also:
OnBeginExport event

2.7.4 Events

Key events

- OnBeginExport
- OnEndExport
- OnExportedRecord
- OnGetExportText
- OnGetFooter
- OnGetHeader
- OnSkippedRecord

2.7.4.1 OnBeginExport

```
property OnBeginExport: TQExportEvent;
```

Description
The *OnBeginExport* event takes place directly before the export beginning.

See also:
Execute method
OnEndExport event

2.7.4.2 OnEndExport

```
property OnEndExport: TQExportEvent;
```

Description
The *OnEndExport* event takes place when the export is complete.

See also:
OnBeginExport event

2.7.4.3 OnExportedRecord

```
property OnExportedRecord: TQRecordExportedEvent;
```

Description
The *OnExportedRecord* event takes place after the export of each source record. It is most frequently used to transfer to the user the information on the export execution process, such as: display of the number of records exported, the increase of the ProgressBar position, etc.

See also:
ExportRecCount property
OnSkippedRecord event
TQRecordExportedEvent type

2.7.4.4 OnGetExportText

```
property OnGetExportText: TQGetExportTextEvent;
```

Description
This event takes place after the export of each source string. Using this event you can replace some definite text of the exported table with another text.

See also:
TQGetExportTextEvent type

2.7.4.5 OnGetFooter

```
property OnGetFooter: TQExportGetColonEvent;
```

Description
The *OnGetFooter* event takes place when the document footer is exported.

See also:
Footer property
TQExportGetColonEvent type
OnGetHeader event

2.7.4.6 OnGetHeader

```
property OnGetHeader: TQExportGetColonEvent;
```

Description
The *OnGetHeader* event takes place when the document header is exported.

See also:
Header property
TQExportGetColonEvent type
OnGetFooter event

2.7.4.7 OnSkippedRecord

```
property OnSkippedRecord: TQRecordExportedEvent;
```

Description
The *OnSkippedRecord* event takes place when one of the first source records is skipped. The number of records to skip is determined by the SkipRecCount property.

See also:
SkipRecCount property
OnExportedRecord event
TQRecordExportedEvent type

2.8 TQExport4Docx

2.8.1 TQExportDocx Reference

Unit
QExport4Docx

Description
The *TQExport4Docx* component allows you to export your data to the MS Word sheets format.

2.8.2 Properties

Run-time only Key properties

Options

2.8.2.1 Options

```
property DocxOptions: TQExport4DocxOptions;
```

Description
The *DocxOptions* property is "complex" and contains some subproperties - properties of the *TQExport4DocxOptions* class.

2.9 TQExport4HTML

2.9.1 TQExport4HTML Reference

Unit
QExport4HTML

Description
The TQExportHTML component is used, as it follows from its name, for exporting data to HTML files. The main peculiarities of the component are:

- complete compatibility with HTML 4.0 specification (see http://www.w3.org);
- compatibility with contemporary browsers (Internet Explorer, Mozilla, Opera), starting with version 3;
- external table styles support (see CSSFileName, UsingCSS) which gives the opportunity of editing simultaneously all the files that exist as a result of the export;
- export templates support (HTMLTemplate property), the opportunity of creating your own templates, of saving them to file and loading them from file;
- the opportunity of creating several HTML files with a definite number of records in each (MaxRecords) and the automatic generation of the index file (GenerateIndex);
- the opportunity of setting the align (ColumnsAlign) and the header for each exported dataset field;

...and much more.

2.9.2 Properties

Run-time only Key properties

- BoolAsCheckBox
- ColumnsAlign
- CSSFileName
- GenerateIndex
- HTMLOptions
- HTMLTemplate
- MaxRecords
- Navigation
- TableOptions
- OverwriteCSSFile
- Title
- UsingCSS
- InterpretTags

2.9.2.1 BoolAsCheckBox

```
property BoolAsCheckBox: boolean;
```

Description
If property *BoolAsCheckBox* is *true* then all the boolean fields of the table are exported as checkboxes.

2.9.2.2 CSSFileName

```
property CSSFileName: string;
```

Description
The *CSSFilename* property may be set only when the property UsingCSS is set to usExternal position, and it determines the name of the external style file for the document that is included with the href attribute of the link element, i.e. if you have the setting *CSSFileName = 'Animals.css'* (and you haven't forgotten about *UsingCSS = usExternal*!), the following string will be placed in the result file

```
<link rel="stylesheet" type="text/css" href="Animals.css">
```

More on the use of style tables see the UsingCSS property description, and also the corresponding part of HTML specification at http://www.w3.org/TR/html4/present/styles.html.

See also:
UsingCSS property

2.9.2.3 GenerateIndex

```
property GenerateIndex: boolean;
```

Description
The *GenerateIndex* property is responsible for the automatic creation of an HTML page, containing links to files created during a multifile export (MaxRecords property value is more than zero).

See also:
MaxRecords property

2.9.2.4 HTMLOptions

```
property HTMLOptions: THTMLOptions;
```

Description
The *HTMLOptions* property sets the general parameters of the result HTML document, and it is complex, i.e. it includes several "subproperties" (in fact, the properties of THTMLOptions class). Besides setting each property by itself, you can set them all at once in accordance with one of the ten pre-defined templates (see HTMLTemplate property description).

See also:
HTMLTemplate property
TableOptions property
THTMLOptions object

2.9.2.5 HTMLTemplate

```
property HTMLTemplate: THTMLTemplate;
```

Description
The *HTMLTemplate* property is used for setting the parameters of the result document (HTMLOptions property) and the exported data table (TableOptions property) simultaneously.

See also:
HTMLOptions property
TableOptions property
LoadTemplateFromFile method
SaveTemplateToFile method
THTMLTemplate type

2.9.2.6 MaxRecords

```
property MaxRecords: integer;
```

Description
The *MaxRecord* property sets the maximum number of the source records placed into one HTML file. If it has a setting different from zero, then during the Execute method execution files FileName00.html, FileName01.html, ... , FileNameNN.html will be created, in each of them exactly *MaxRecord* of the source records will be placed (it is assumed that FileName is a setting of the FileName property). If *MaxRexords = 0* (default), all the records will be exported to one HTML file.

See also:
GenerateIndex property

2.9.2.7 Navigation

```
property Navigation: TQExportHTMLNavigation;
```

Description
The *Navigation* property allows you to define various options of the multi-file export.

See also:
GenerateIndex property
MaxRecords property
TQExportHTMLNavigation object

2.9.2.8 TableOptions

```
property TableOptions: TTableOptions;
```

Description
Similar to the HTMLOptions property, the *TableOptions* property is a "complex" property and it includes several subproperties (the properties of the TTableOptions class). Along with setting each property itself, you can set them all at once in accordance with one of the ten pre-defined templates (see the HTMLTemplate property description).

See also:
HTMLOptions property
HTMLTemplate property
TTableOptions object

2.9.2.9 OverwriteCSSFile

```
property OverwriteCSSFile: boolean;
```

Description
If *OverwriteCSSFile* is *true* and UsingCSS is *true* and the file specified in CSSFileName already exists, then it will be rewrittren.

2.9.2.10 Title

```
property Title: string;
```

Description
The *Title* property determines the title of the exported document.

See also:
Footer property
Header property

2.9.2.11 UsingCSS

```
property UsingCSS: TUsingCSS;
```

Description
The *UsingCSS* property sets the position of the style table (inside or outside the HTML file). With the *usInternal* setting the table is placed directly in the HTML file that can look, for example, as follows (Template = htClassic):

```
<Style type="text/css">
BODY  background: #333399; color: #FFFFFF; font-family: Arial;
A:link  color: #69EF7D
A:visited  color: #FF00FF
A:active  color: #00FF00
.ThRows  background-color: #FF0000; color: #FFFFFF; font-weight: bold; text-align: center
.TrRows  background-color: #007AEC; color: #FFFFFF
.TrOdd  background-color: #006BCE; color: #FFFFFF
</Style>
```

If *UsingCSS* = *usExternal*, the text above will be placed in the file specified by the CSSFileName property, and in the HTML file itself there will be only one line

```
<link rel="stylesheet" type="text/css" href="Animals.css">
```

The use of the *usExternal* setting can be very helpful in case you create several HTML files; then to change, for example, the background color of the even table rows in all the created files it is enough to correct only one setting of the style file (in this example it is Animals.css).
More on the use of style tables in HTML documents see http://www.w3.org/TR/html4/present/styles.html.

See also:
CSSFileName property
TUsingCSS type

2.9.2.12 InterpretTags

```
property InterpretTags: boolean;
```

Description
If property *InterpretTags* is true then all special symbols <, >, ", & found in exported data (text) will be replaced with corresponding *< > " &* ones.

See also:
HTMLOptions property
TableOptions property
LoadTemplateFromFile method
SaveTemplateToFile method
THTMLTemplate type

2.9.3 Methods

Key methods

- Abort
- Execute
- LoadTemplateFromFile
- SaveTemplateToFile

2.9.3.1 LoadTemplateFromFile

```
procedure LoadTemplateFromFile(const FileName: string);
```

Description
Method *LoadTemplateFromFile* loads from the *FileName* file the template previously saved by the SaveTemplateToFile procedure.

See also:
HTMLTemplate property

2.9.3.2 SaveTemplateToFile

```
procedure SaveTemplateToFile(const FileName: string);
```

Description
The procedure saves the current value of the TableOptions and HTMLOptions properties to the file specified by the *FileName* parameter. The saved template can be loaded later using the LoadTemplateFromFile procedure.

See also:
HTMLTemplate property

2.9.4 Events

Key events

OnGetCellParams

2.10 TQExport4LaTeX

2.10.1 TQExport4LaTeX Reference

Unit
QExport4LaTeX

Description
The *TQExport4LaTeX* component is used for the data export to LaTeX format which is a popular (especially among mathematicians and physicists) macroextension by of TeX pack developed by D.Knut. For more info on TeX, its various modifications, extensions and Net resources, see TeX Users Group (TUG) at http://www.tug.org.
The Russian-speaking users can also visit the site of the Cyrillic TeX Users Group (CyrTUG) at http://www.cemi.rssi.ru/cyrtug/.

Note: The current component version was tested with MikTeX 1.20e (http://www.miktex.org) - freeware version of TeX and LaTeX for Win32.

2.10.2 Properties

Run-time only Key properties

- Options
- Preambula

2.10.2.1 Options

```
property Options: TLaTeXOptions;
```

Description
The *Options* property is "complex" and contains a number of subproperties - properties of class TLaTeXOptions.

See also:
TLaTeXOptions component

2.10.2.2 Preambula

```
property Preambula: TStrings;
```

Description
The *Preambula* property determines all the additional LaTeX commands that you would like to place before document. The commands already defined by the subproperties of LaTeXOptions property shouldn't be incuded in this propety. For more info on LaTeX commands can be found at TUG.

See also:
Options property

2.10.3 Methods

Key methods

Abort

2.11 TQExport4ODS

2.11.1 TQExport4ODS Reference

Unit
QExport4ODS

Description
The *TQExport4ODS* component allows you to export your data to the OpenDocument Spreadsheet format.

2.11.2 Properties

Run-time only Key properties

- SheetName
- ODSOptions

2.11.2.1 SheetName

```
property SheetName: String;
```

Description
This property allows you to set sheet name of OpenDocument Spreadsheet.

2.11.2.2 ODSOptions

```
property ODSOptions: TQExportODSOptions;
```

Description
The *ODSOptions* property is "complex" and contains some subproperties - properties of the *TQExportODSOptions* class.

2.12 TQExport4ODT

2.12.1 TQExport4ODT Reference

Unit
QExport4ODT

Description
The *TQExport4ODT* component allows you to export your data to the OpenDocument Text format.

2.12.2 Properties

Run-time only Key properties

- TableName
- ODTOptions

2.12.2.1 TableName

```
property TableName: String;
```

Description
This property allows you to set table name of OpenDocument Text.

2.12.2.2 ODTOptions

```
property ODTOptions: TQExport4ODTOptions;
```

Description
The *ODTOptions* property is "complex" and contains some subproperties - properties of the *TQExportODTOptions* class.

2.13 TQExport4PDF

2.13.1 TQExport4PDF Reference

Unit
QExport4PDF

Description
The *TQExport4PDF* component is intended for exporting data to PDF format.

2.13.2 Properties

Options
BreakString

2.13.2.1 Options

```
property Options: TPDFOptions;
```

Description

Use the *Options* property to define customize the result PDF file - tune its fonts, grid, etc.

2.13.2.2 BreakString

```
type
  TBreakStringCriteria = (bscNone, bscWidth, bscData);

property BreakString: TBreakStringCriteria;
```

Description
Use the *BreakString* property to define the rules for breaking lines in the document.

bscNone - do not break lines
bscWidth - fit lines to the column width
bscData - use line breaks in the source line

2.14 TQExport4RTF

2.14.1 TQExport4RTF Reference

Unit
QExport4RTF

Description
The *TQExport4RTF* component is used for data export to RTF (Rich Text Format) format supported by many text processing programs (e.g. Microsoft Word).

2.14.2 Properties

Run-time only Key properties

ColumnsAlign
ColumnsWidth
Options

2.14.2.1 Options

```
property Options: TRTFOptions;
```

Description
The *Options* property is "complex" and contains some subproperties - properties of the TRTFOptions class.

See also:
TRTFOptions object

2.14.3 Methods

Key methods

Abort

2.14.4 Events

Key events

OnGetCaptionStyle
OnGetDataStyle
OnGetFooterStyle
OnGetHeaderStyle

2.14.4.1 OnGetCaptionStyle

```
type
  TrtfGetCaptionStyleEvent = procedure(Sender: TObject; Style: TrtfStyle; ColNo: integ
of object;
  TQExportColAlign = (ecaLeft, ecaCenter, ecaRight);

property OnGetCaptionStyle: TrtfGetCaptionStyleEvent;
```

Description
The *OnGetCaptionStyle* event takes place when the style of the caption is received. Depending on the column number you can edit the caption apperance style. See demo application to learn more about this event.

2.14.4.2 OnGetDataStyle

```
type
  TrtfGetDataStyleEvent = procedure(Sender: TObject; Style: TrtfStyle; ColNo: integer;
of object;
  TQExportColAlign = (ecaLeft, ecaCenter, ecaRight);

property OnGetDataStyle: TrtfGetDataStyleEvent;
```

Description

The *OnGetDataStyle* event takes place when the style of data is received. Depending on the column number you can edit the data appearance style. See demo application to learn more about this event.

2.14.4.3 OnGetHeaderStyle

```
type
  TrtfGetStyleEvent = procedure(Sender: TObject; Style: TrtfStyle) of object;

property OnGetHeaderStyle: TrtfGetStyleEvent;
```

Description

The *OnGetHeaderStyle* event takes place when the style of header is received. Type code here to change the document's header appearance style. See demo application to learn more about this event.

2.14.4.4 OnGetFooterStyle

```
type
  TrtfGetStyleEvent = procedure(Sender: TObject; Style: TrtfStyle) of object;

property OnGetFooterStyle: TrtfGetStyleEvent;
```

Description

The *OnGetFooterStyle* event takes place when the style of footer is received. Type code here to change the document's footer appearance style. See demo application to learn more about this event.

2.15 TQExport4SQL

2.15.1 TQExport4SQL Reference

Unit
QExport4SQL

Description
The *TQExport4SQL* component allows you to export your data to the SQL script file as a set of INSERT statements.

2.15.2 Properties

Run-time only Key properties

- CommitAfterScript
- CommitRecCount
- CommitStatement
- CreateTable
- FormatValues
- MultipleInsert
- NullValue
- StatementTerm
- TableName

2.15.2.1 CommitAfterScript

```
property CommitAfterScript: boolean;
```

Description
This property allows you to insert the commit statement to the bottom line of the output script. You can define this statement using the CommitStatement property.

See also:
CommitRecCount property
CommitStatement property
CreateTable property
StatementTerm property
TableName property

2.15.2.2 CommitRecCount

```
property CommitRecCount: integer;
```

Description
This property allows you to define a number of records after which the commit statement is inserted to the output script. The default number is 0, i.e. no commits are inserted. You can define the commit statement using the CommitStatement.

See also:
CommitAfterScript property
CommitStatement property
CreateTable property
StatementTerm property
TableName property

2.15.2.3 CommitStatement

```
property CommitStatement: string;
```

Description
This property determines the commit statement to insert to the result script after a definite number of records (see CommitRecCount property) or at the end of the script (see CommitAfterScript property).

See also:
CommitAfterScript property
CommitRecCount property
CreateTable property
StatementTerm property
TableName property

2.15.2.4 CreateTable

```
property CreateTable: boolean;
```

Description
If this property is *true*, then the CREATE TABLE statement is inserted to the output script. You can define the table name using the TableName property.

See also:
CommitAfterScript property
CommitRecCount property
CommitStatement property
StatementTerm property
TableName property

2.15.2.5 FormatValues

```
property FormatValues: boolean;
```

Description
This property permits or forbids formatting of exported values according to the content of the Formats property.

See also:
CommitAfterScript property
CommitRecCount property
CommitStatement property
StatementTerm property
TableName property

2.15.2.6 MultipleInsert

```
property MultipleInsert: boolean;
```

Description
This property permits or forbids generating a single INSERT statement. If this property is set to *True* then the result single insert statement contains multiple rows at a time.

2.15.2.7 NullValue

```
property NullValue: string;
```

Description
This property contains the string value that is to be inserted to the output script if record has the fields with NULL values.

See also:
CommitAfterScript property
CommitRecCount property
CommitStatement property
StatementTerm property
TableName property

2.15.2.8 StatementTerm

```
property StatementTerm: char;
```

Description
This property determines the character to denote the end of each statement. The default character is semicolon - ';'.

See also:
CommitAfterScript property
CommitRecCount property
CommitStatement property
CreateTable property
TableName property

2.15.2.9 TableName

```
property TableName: string;
```

Description
This property allows you to set the name of the table to use in the INSERT statement and CREATE TABLE statement, if the CreateTable property is *true*.

See also:
CommitAfterScript property
CommitRecCount property
CommitStatement property
CreateTable property
StatementTerm property

2.15.3 Types conversion

Depending on different SQL standards internal Delphi types TFieldType are converted differently into data types for CREATE TABLE statements.

DB2
Interbase/Firebird
MS SQL Server
MY SQL
Oracle
Postgresql
ANSI SQL

2.15.3.1 DB2

Delphi type	*SQL data type*
ftSmallint	INTEGER
ftBoolean	BOOLEAN
ftFloat	FLOAT
ftCurrency, ftBCD, ftFMTBcd	NUMERIC(24,4)
ftDate	DATE
ftTime	TIME
ftDateTime, ftTimeStamp	TIMESTAMP
ftParadoxOle, ftArray, ftOraBlob, ftOraClob, ftDBaseOle, ftGraphic, ftBytes, ftVarBytes, ftBlob, ftTypedBinary, ftFmtMemo, ftWideString	BLOB
ftMemo, ftVariant	CLOB
ftLargeint	BIGINT

For the rest of types the size of a field is checked. If the field size is less than 255 chars then it is converted into VARCHAR. Otherwise, into BLOB.

2.15.3.2 Interbase/Firebird

Delphi type	*SQL data type*
ftSmallint	SMALLINT
ftBoolean	BOOLEAN
ftFloat	DOUBLE PRECISION
ftCurrency, ftBCD, ftFMTBcd	NUMERIC (15,2)
ftDate	DATE
ftTime	TIME
ftDateTime	TIMESTAMP
ftBlob, ftMemo, ftGraphic, ftFmtMemo, ftParadoxOle, ftDBaseOle, ftOraBlob, ftOraClob, ftBytes, ftVarBytes, ftVariant	BLOB

For the rest of types the size of a field is checked. If the field size is less than 255 chars then it is converted into VARCHAR. Otherwise, into BLOB.

2.15.3.3 MS SQL Server

Delphi type	*SQL data type*
ftSmallint	SMALLINT
ftBoolean	CHAR(1)
ftFloat	FLOAT
ftCurrency, ftBCD, ftFMTBcd	NUMERIC(15,2)
ftDate, ftTime, ftDateTime	DATETIME
ftTime	TIME
ftDateTime, ftTimeStamp	TIMESTAMP
ftOraClob, ftMemo, ftFmtMemo	TEXT
ftOraBlob, ftBlob, ftGraphic	IMAGE
ftGuid	NCHAR
ftWideString	<8000 chars - NVARCHAR >8000 chars - NTEXT
ftBytes	BINARY
ftVarBytes	VARBINARY
ftTimeStamp	TIMESTAMP
ftVariant	SQL_VARIANT

For the rest of types the size of a field is checked. If the field size is less than 255 chars then it is converted into VARCHAR. Otherwise, into TEXT.

2.15.3.4 MY SQL

Delphi type	*SQL data type*
ftSmallint	SMALLINT
ftBoolean	BOOLEAN
ftInteger, ftWord, ftAutoInc	INTEGER
ftLargeint	BIGINT
ftFloat	DOUBLE
ftCurrency, ftBCD, ftFMTBcd	FLOAT(15,2)
ftDate	DATE
ftTime	TIME
ftDateTime, ftTimeStamp	DATETIME
ftBlob, ftMemo, ftGraphic, ftFmtMemo, ftParadoxOle, ftDBaseOle, ftOraBlob, ftOraClob, ftVariant	LONGBLOB
ftMemo, ftVariant	CLOB
ftLargeint	BIGINT

For the rest of types the size of a field is checked. If the field size is less than 2048 chars then it is converted into VARCHAR. Otherwise, into TEXT.

2.15.3.5 Oracle

Delphi type	*SQL data type*
ftWideString	<2000 chars - NVARCHAR2 >2000 chars - BLOB
ftSmallint, ftInteger, ftWord, ftAutoInc, ftLargeint, ftFloat, ftCurrency, ftBCD, ftFMTBcd	NUMBER
ftBoolean	CHAR(1)
ftDate	DATE
ftTime, ftDateTime, ftTimeStamp	DATE
ftDateTime, ftTimeStamp	TIMESTAMP
ftParadoxOle, ftArray, ftOraBlob, ftOraClob, ftDBaseOle, ftGraphic, ftBytes, ftVarBytes, ftBlob, ftTypedBinary, ftFmtMemo	BLOB
ftMemo, ftVariant	CLOB
ftMemo, ftVariant	CLOB
ftLargeint	BIGINT

For the rest of types the size of a field is checked. If the field size is less than 255 chars then it is converted into VARCHAR. Otherwise, into BLOB.

2.15.3.6 Postgresql

Delphi type	*SQL data type*
ftSmallint	SMALLINT
ftInteger, ftWord, ftAutoInc, ftReference	INTEGER
ftLargeint	BIGINT
ftBoolean	BOOLEAN
ftFloat	DOUBLE PRECISION
ftCurrency, ftBCD, ftFMTBcd	NUMERIC(20,4)
ftDate	DATE
ftTime	TIME
ftDateTime, ftTimeStamp	TIMESTAMP
ftParadoxOle, ftDBaseOle, ftTypedBinary, ftCursor, ftBytes, ftVarBytes,ftBlob, ftArray, ftOraBlob, ftDataSet	BYTEA
ftWideString, ftMemo, ftFmtMemo, ftGraphic, ftADT, ftOraClob, ftVariant, ftInterface, ftIDispatch, ftGuid	TEXT

For the rest of types the size of a field is checked. If the field size is less than 255 chars then it is converted into VARCHAR. Otherwise, into BLOB.

2.15.3.7 ANSI SQL

Delphi type	*SQL data type*
ftSmallint	SMALLINT
ftInteger, ftWord, ftAutoInc, ftLargeint	INTEGER
ftBoolean	BOOLEAN
ftFloat	FLOAT
ftCurrency, ftBCD, ftFMTBcd	DOUBLE PRECISION
ftDate	DATE
ftTime	TIME
ftDateTime, ftTimeStamp	TIMESTAMP
ftParadoxOle, ftArray, ftOraBlob, ftOraClob, ftDBaseOle, ftGraphic, ftBytes, ftVarBytes, ftBlob, ftTypedBinary, ftFmtMemo, ftMemo, ftVariant	TEXT

For the rest of types the size of a field is checked. If the field size is less than 255 chars then it is converted into VARCHAR. Otherwise, into TEXT.

2.16 TQExport4XLS

2.16.1 TQExport4XLS Reference

Unit
QExport4XLS

Description
The *TQExport4XLS* component is used to export data to the most popular e-table format - Microsoft Excel. The result files are fully compatible with the versions 95, 97, 2000, XP, and 2003.

2.16.2 Properties

Run-time only Key properties

- AutoAddSheet
- AutoTruncateValue
- Cells
- Charts
- DefColWidth
- DefRowHeight
- ExportStage
- FieldFormats
- FooterRows
- HeaderRows
- HyperLinks
- Images
- MergedCells
- Pictures
- Notes
- Options
- Sheets
- SplitByRows
- StartDataCol
- StripStyles
- StripType

When AutoAddSheet = True, the new sheets are added automatically. When AutoAddSheet = False - upon reaching the maximum number of the lines the error message on the sheet appears.

property is added. If the value of more than 32767 symbols is inserted in the cell, the length of the line will be truncated automatically to the enabled when AutoTrancateValue = True. When AutoTrancateValue = False, the error message "String too large" will appear.

2.16.2.1 AutoAddSheet

```
property AutoAddSheet: boolean;
```

Description
The AutoAddSheet property is valid when exporting more then 65536 lines.
If AutoAddSheet = True a new sheet is added.
If AutoAddSheet = False - the error message appears on the sheet.

2.16.2.2 AutoTruncateValue

```
property AutoTruncateValue: boolean;
```

Description
If the value of more than 32767 symbols is inserted into the cell, the length of the line will be truncated automatically to the enabled when AutoTruncateValue = True. When AutoTruncateValue = False, the error message "String too large" will appear.

2.16.2.3 Header

```
property Header: TqeStrings;
```

If the compiler version is VER140 or higher then `TqeStrings = TWideStrings`, otherwise `TqeStrings`

Description:
The *Header* property is used for outputting text before exported data. In order to output text into sep
use the symbol-separator '#9'.

2.16.2.4 Cells

```
property Cells: TxlsCells;
```

Description
The *Cells* property allows you to insert values to the definite places in the result Excel document. TxlsCells is a collection of TxlsCell objects which define the parameters of each single cell.

See also:
TxlsCells object
Charts property
HyperLinks property
Images property
MergedCells property
Notes property
Pictures property

2.16.2.5 Charts

```
property Charts: TxlsCharts;
```

Description
The *Charts* property allows you to insert charts based on data from the ExportSource to the result Excel document. TxlsCharts is a collection of TxlsChart objects which define the parameters of each single chart.

See also:
TxlsCharts object
Cells property
HyperLinks property
Images property
MergedCells property
Notes property
Pictures property

2.16.2.6 DefColWidth

```
property DefColWidth: integer;
```

Description
Use the *DefColWidth* property to set the column width height for the current sheet.

2.16.2.7 DefRowHeight

```
property DefRowHeight: double;
```

Description
Use the *DefRowHeight* property to set the default row height for the current sheet.

2.16.2.8 ExportStage

```
property ExportStage: TxlsExportStage;
```

Description
The *ExportStage* property indicates the stage of the export process. The following values are available:
esNone - no export
esHeader - header exported
esCaption - captions exported
esData - data exported
esAggregate - aggregate functions exported
esFooter - footer exported

2.16.2.9 FieldFormats

```
property FieldFormats: TxlsFieldFormats;
```

Description
The *FieldFormats* property allows you to set the formats for each source field separately.

Note: The **StripStyles** property has a higher priority than the **FieldFormats** property. If the **StripStyles** property is defined and the **StripType**property is set to *ssCol* or *ssRow*, all the field styles defined in the **FieldFormats** property takes no effect.

See also:
TxlsFieldFormats component

2.16.2.10 FooterRows

```
property FooterRows: word;
```

Description
The *FooterRows* property defines the number of rows in the Footer property to include in the result file. If the FooterRows value is 0 then all the rows from the Footer property will be included in the result file.

See also:
HeaderRows property

2.16.2.11 HeaderRows

```
property HeaderRows: word;
```

Description
The *HeaderRows* property defines the number of rows in the Header property to include in the result file. If the *HeaderRows* value is 0 then all the rows from the Header property will be included in the result file.

See also:
FooterRows property

2.16.2.12 HyperLinks

```
property HyperLinks: TxlsHyperLinks;
```

Description
The *HyperLinks* property allows you to insert hyperlinks to the result Excel document. TxlsHyperLinks is a collection of TxlsHyperLink objects which define the parameters of each single hyperlink.

See also:
TxlsHyperLinks object
Cells property
Charts property
Images property
MergedCells property
Notes property
Pictures property

2.16.2.13 Images

```
property Images: TxlsImages;
```

Description
The *Images* property allows you to insert images in the result Excel document. TxlsImages is a collection of TxlsImage objects which define the parameters of each single image.

See also:
TxlsImages object
Cells property
Charts property
HyperLinks property
MergedCells property
Notes property
Pictures property

2.16.2.14 MergedCells

```
property MergedCells: TxlsMergedCellList;
```

Description
The *MergedCells* property allows you to merge a few cells in the result Excel document in one cell. TxlsMergedCellList is a collection of TxlsMergedCells objects which define the parameters of merging.

See also:
TxlsMergedCellList object
Cells property
Charts property
HyperLinks property
Images property
Notes property
Pictures property

2.16.2.15 Pictures

```
property Pictures: TxlsPictures;
```

Description
The *Pictures* property allows you to add pictures to the result Excel document. TxlsPictures is a collection of TxlsPicture objects which define the parameters of each single picture.

See also:
TxlsPictures object
Cells property
Charts property
HyperLinks property
Images property
MergedCells property
Notes property

2.16.2.16 Notes

```
property Notes: TxlsNotes;
```

Description
The *Notes* property allows you to insert notes to the result Excel document. TxlsNotes is a collection of TxlsNote objects which define the parameters of each single note.

See also:
TxlsNotes object
Cells property
Charts property
HyperLinks property
Images property
MergedCells property
Pictures property

2.16.2.17 Options

```
property Options: TXLSOptions;
```

Description
The *Options* property is "complex" and contains some subproperties - properties of the TXLSOptions class.

See also:
TxlsOptions object

2.16.2.18 Sheets

```
property Sheets: TxlsSheets;
```

Description
Use property *Sheets* to export data to several Excel worksheets with possibility to set a specific format and a separate data source for each of them. Sheets is a collection of *TxlsSheet* objects, properties of which totally correspond to the properties of *TQExport4XLS* component, including properties DataSet, DBGrid, ListView, StringGrid and ExportSource which allow you to set data source for each sheet; the Options property to set all the sheet options, FieldFormats to set format for each field separately; and more. Also *TxlsSheet* class has a boolean property *Exported* which allows you to define, if the current sheet is exported or not.

Note that if you keep the *Sheets* collection empty, the result file consists of only one sheet which uses options, set in the component properties, but if you add at least one sheet to the collection, the sheet options are used, and component properties are not taken into consideration.

2.16.2.19 SplitByRows

```
property SplitByRows: integer;
```

Description

The *SplitByRows* property is used to define the number of rows that the set of exported records will be splitted in order to place every set of rows on a single Excel sheet.

2.16.2.20 StartDataCol

```
property StartDataCol: byte;
```

Description
The *StartDataCol* property defines the number of column to start insert data from.

2.16.2.21 StripStyles

```
property StripStyle: TxlsFormats;
```

Description
Use *StripStyles* do define repeating styles for columns or rows of the result sheet. *StripStyle* determines font size, color and other attributes, border and fill styles and more. To apply this styles to columns or rows use property StripType.

The pictures below show the examples of the result Excel sheet with two styles applied.

The styles were defined in the following way:

```
StripStyles.Items[0].Fill.Background = clrPaleBlue;
StripStyles.Items[0].Fill.Pattern = ptSolid;
StripStyles.Items[1].Fill.Background = clrLightTurquoise;
StripStyles.Items[1].Fill.Pattern = ptSolid;

StripType = ssCol.

StripType = ssRow.
```

Note: The **StripStyles** property has a higher priority than the **FieldFormats** property. If the **StripStyles** property is defined and the **StripType**property is set to *ssCol* or *ssRow*, all the field styles defined in the **FieldFormats** property takes no effect.

See also:
StripType property

2.16.2.22 StripType

```
type
  TxlsStripType = (ssNone, ssCol, ssRow);

property StripType: TxlsStripType;
```

Description
The *StripType* property defines if property StripStyles should be applied to columns or rows of the result Excel sheet. If StripType is *ssNone*, then StripStyles are not used, if StripType is *ssCol*, then StripStyles are applied to the sheet columns, and if StripType is *ssRow*, then StripStyles are applied to the sheet rows. Note that if StripType is not *ssNone*, then all the FieldFormats are ignored, except the Aggregate property.

See also:
StripStyles property

2.16.2.23 PageSetup

```
property PageSetup: TxlsPageSetup;
```

Description
The PageSetup property allows you to define printing parameters for the XLS format. The *PageSetup* property is "complex" and contains some subproperties - properties of the TxlsPageSetup class.

2.16.3 Methods

Key methods

- AddBooleanCell
- AddDateTimeCell
- AddMergedCells
- AddNumericCell
- AddStringCell

Execute

2.16.3.1 AddBooleanCell

```
function AddBooleanCell(Col, Row: word; Value: boolean): TxlsCell;
```

Description
The *AddBooleanCell* adds the boolean value defined in the Value parameter to the specified position. The position is defined by the *Col* parameter as a column number and the *Row* parameter as a row number. The *Row* and the *Col* parameters are 1-based. The returned value is the created TxlsCell object. You can define the properties of this object to customize the created cell.

See also:
AddDateTimeCell method
AddNumericCell method
AddStringCell method
Cells property

2.16.3.2 AddDateTimeCell

```
function AddDateTimeCell(Col, Row: word; DateTimeFormat: string; Value: TDateTime): Tx
```

Description
The *AddDateTimeCell* adds the date/time value defined in the *Value* parameter to the specified position. The position is defined by the *Col* parameter as a column number and the *Row* parameter as a row number. The *Row* and the *Col* parameters are 1-based. The *DateTimeFormat* parameter is a formatting string for the value. The returned value is the created TxlsCell object. You can define the properties of this object to customize the created cell.

See also:
AddBooleanCell method
AddNumericCell method
AddStringCell method
Cells property

2.16.3.3 AddMergedCells

```
function AddMergedCells(FirstRow, LastRow, FirstCol, LastCol: word): TxlsMergedCells;
```

Description
The *AddMergedCells* method allows you to merge cells in the defined ranges to one cell. The row ranges are set by the *FirsRow* and the *LastRow* parameters. The column ranges are set by the *FirstCol* and the *LastCol* parameters. These parameters are 1-based. The returned value is the TxlsMergedCells object.

See also:
TxlsMergedCells object
MergedCells property

2.16.3.4 AddNumericCell

```
function AddNumericCell(Col, Row: word; NumericFormat: string; Value: double): TxlsCe
```

Description
The *AddNumericCell* method adds the numeric value defined in the Value parameter to the specified position. The position is defined by the *Col* parameter as a column number and the *Row* parameter as a row number. The *Row* and the *Col* parameters are 1-based. The *NumericFormat* parameter is a formatting string for the value. The returned value is the created TxlsCell object. You can define the properties of this object to customize the created cell.

See also:
AddBooleanCell method
AddDateTimeCell method
AddStringCell method
Cells property

2.16.3.5 AddStringCell

```
function AddStringCell(Col, Row: word; const Value: string): TxlsCell;
```

Description
The *AddStringCell* methods adds the string value defined in the *Value* parameter to the specified position. The position is defined by the *Col* parameter as a column number and the *Row* parameter as a row number. The *Row* and the *Col* parameters are 1-based. The returned value is the created TxlsCell object. You can define the properties of this object to customize the created cell.

See also:
AddBooleanCell method
AddDateTimeCell method
AddStringCell method
Cells property

2.16.4 Events

Key events

- OnAfterExportSheet
- OnBeforeExportSheet
- OnGetDataParams
- OnGetAggregateParams
- OnGetBeforeDataParams
- OnGetCaptionParams
- OnGetFooterParams
- OnGetHeaderParams
- OnAdvancedExportedRecord
- OnAdvancedGetExportText

2.16.4.1 OnAfterExportSheet event

```
type
  TxlsExportSheetEvent = procedure(Sender: TObject; SheetIndex: integer) of object;

property OnAfterExportSheet: TxlsExportSheetEvent;
```

Description
The *OnAfterExportSheet* event takes place after the sheet export. Depending on the sheet number you can change the sheet parameters. See demo application to learn more about this event.

2.16.4.2 OnBeforeExportSheet event

```
type
  TxlsExportSheetEvent = procedure(Sender: TObject; SheetIndex: integer) of object;

property OnBeforeExportSheet: TxlsExportSheetEvent;
```

Description

The *OnBeforeExportSheet* event takes place before the sheet export. Depending on the sheet number you can change the sheet parameters. See demo application to learn more about this event.

2.16.4.3 OnGetDataParams

```
property OnGeDataParams: TGetDataParamsEvent;
```

Description
The *OnGetDataParams* event takes place when the parameters of the data cell are received. Depending on the column and row number, you can change the format of the cell value. See demo application to learn more about this event.

See also:
TGetDataParamsEvent type
OnGetAggregateParams event
OnGetBeforeDataParams event
OnGetCaptionParams event
OnGetFooterParams event
OnGetHeaderParams event

2.16.4.4 OnGetAggregateParams

```
property OnGetAggregateParams: TGetAggregateParamsEvent;
```

Description
The *OnGetAggregateParams* event takes place when the parameters of the cell containing aggregate function are received. Depending on the column number, you can change the format of the cell value, or, if the value is empty, set a string value to the cell, e.g. 'Total:'. See demo application to learn more about this event.

See also:
TGetAggregateParamsEvent type
OnGeDataParams event
OnGetBeforeDataParams event
OnGetCaptionParams event
OnGetFooterParams event
OnGetHeaderParams event

2.16.4.5 OnGetBeforeDataParams

```
property OnGetBeforeDataParams: TGetHeaderFooterParamsEvent;
```

Description
The *OnGetBeforeDataParams* event takes place when an empty cell should be inserted to the result table (e.g. if StartDataCol>0). Depending on the cell position and format you can set a string value to the cell. See demo application to learn more about this event.

See also:
TGetHeaderFooterParamsEvent type
OnGeDataParams event
OnGetAggregateParams event
OnGetCaptionParams event
OnGetFooterParams event
OnGetHeaderParams event

2.16.4.6 OnGetCaptionParams

```
property OnGetCaptionParams: TGetCaptionParamsEvent;
```

Description
The *OnGetCaptionsParams* event takes place when the parameters of the cells with column captions are received. Depending on the column number and cell format you can edit the cell value. See demo application to learn more about this event.

See also:
TGetCaptionParamsEvent type
OnGeDataParams event
OnGetAggregateParams event
OnGetBeforeDataParams event
OnGetFooterParams event
OnGetHeaderParams event

2.16.4.7 OnGetFooterParams

```
property OnGetFooterParams: TGetHeaderFooterParamsEvent;
```

Description
The *OnGetFooterParams* event takes place when the parameters of the footer cell are received. Depending on the cell position and format you can edit the cell value. See demo application to learn more about this event.

See also:
TGetHeaderFooterParamsEvent type
OnGeDataParams event
OnGetAggregateParams event
OnGetBeforeDataParams event
OnGetCaptionParams event
OnGetHeaderParams event

2.16.4.8 OnGetHeaderParams

```
property OnGetHeaderParams: TGetHeaderFooterParamsEvent;
```

Description
The *OnGetHeaderParams* event takes place when the parameters of the header cell are received. Depending on the cell position and format you can edit the cell value. See demo application to learn more about this event.

See also:
TGetHeaderFooterParamsEvent type
OnGeDataParams event
OnGetAggregateParams event
OnGetBeforeDataParams event
OnGetCaptionParams event
OnGetFooterParams event

2.16.4.9 OnAdvancedExportedRecord

```
type
  TxlsExportedRecordEvent = procedure(Sender: TObject; Sheet, RecNo: integer) of objec

property OnAdvancedExportedRecord: TxlsExportedRecordEvent;
```

Description
The *OnAdvancedExportedRecord* event is the expanded analog of the OnExportedRecord event of the TQExport class. The only difference in these events is the *Sheet* parameter which defines the result Excel sheet to apply the formatting to.

See also:
OnExportedRecord event
OnAdvancedGetExportText event

2.16.4.10 OnAdvancedGetExportText

```
type
  TxlsGetExportTextEvent = procedure(Sender: TObject; Sheet, ColNo: integer; var Text

property OnAdvancedGetExportText: TxlsGetExportTextEvent;
```

Description
The *OnAdvancedGetExportText* event is the expanded analogue of the OnGetExportText event of the TQExport4 class. The only difference in these events is the *Sheet* parameter which defines the result Excel sheet to apply the formatting to.

Note: The *OnAdvancedGetExportText* event takes place AFTER the OnGetExportText event.

See also:
OnGetExportText event
OnAdvancedExportedRecord event

2.17 TQExport4Xlsx

2.17.1 TQExport4Xlsx Reference

Unit
QExport4Xlsx

Description
The *TQExport4Xlsx* component allows you to export your data to the MS Excel sheets format.

2.17.2 Properties

Run-time only Key properties

- SheetName
- XlsxOptions

2.17.2.1 SheetName

```
property SheetName: WideString;
```

Description
This property allows you to set sheet name of MS Excel 2007.

2.17.2.2 XlsxOptions

```
property XlsxOptions: TQExport4XlsxOptions;
```

Description
The *XlsxOptions* property is "complex" and contains some subproperties - properties of the *TQExport4XlsxOptions* class.

2.17.3 Events

Key events

OnGetDataParams

2.17.3.1 OnGetDataParams

```
property OnGetDataParams: TOnGetDataParamsEvent;
```

Description
The *OnGetDataParams* event takes place when the parameters of the data cell are received. Depending on the column and row number, you can change the style of the cell. To apply the style set *UseStyle* = *True*.

2.18 TQExport4XML

2.18.1 TQExport4XML Reference

Unit
QExport4XML

Description
The *TQExport4XML* component allows you to export your data to the XML format.

2.18.2 Properties

Run-time only Key properties

- Options
- DocumentType
- ExportXSDSchema

2.18.2.1 Options

```
property Options: TXMLOptions;
```

Description
The *Options* property is "complex" and contains some subproperties - properties of the *TXMLOptions* class.

See also:
TXMLOptions object

2.18.2.2 DocumentType

```
type
TQExportXMLType = (xtDatapacket2, xtAccess);

property DocumentType: TQExportXMLType;
```

Description

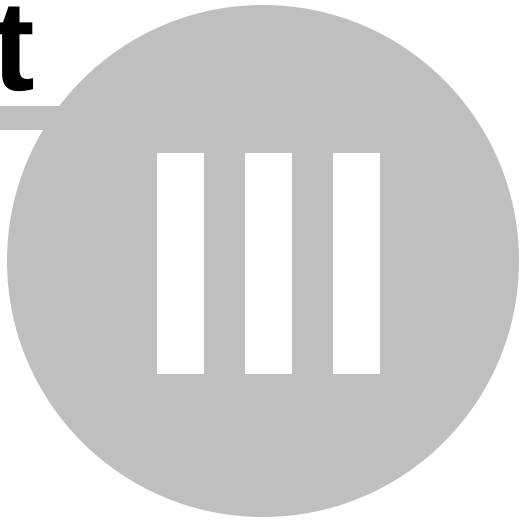
Use this property to define the xml document type. If the *xtDatapacket2* is specified, then the xml document of Data Packet standard will be generated (used by default). If the *xtAccess* is specified, then the target XML document will be of the Access type.

2.18.2.3 ExportXSDSchema

```
property ExportXSDSchema Boolean;
```

Description
You can only use this property when DocumentType is set to *xtAccess*. This property allows you to create a XSD schema with description of the table from which data are exported.

Part



3 Units

3.1 QExport4 unit

Components
TQExport4

Objects
TQExport4Text
TQExportCol
TQExportFormats
TQExportRow

Types
TExportedRecordEvent
TGetCellParamsEvent
TGetExportTextEvent
TQExportStopEvent

3.1.1 TQExport4Text component

Unit
QExport4

Description
The *TQExport4Text* class contains properties which define the parameters of the result export file: filename, and if file is opened/printed after export.

3.1.1.1 Properties

Run-time only Key properties

- AppendDateTimeToFileName
- ExportedFileName
- FileName
- PrintFile
- ShowFile

3.1.1.1.1 AppendDateTimeToFileName

```
property AppendDateTimeToFileName : boolean;
```

Description
Setting the *AppendDateTimeToFileName* property to *True* adds the export date and time to the name of the result exported file.

See also:
FileName property

3.1.1.1.2 ExportedFileName

```
property ExportedFileName : string;
```

Description
The *ExportedFileName* property defines the result exported file name considering the value of the AppendDateTimeToFileName property. If this property is set to *True* then the result file name consists of the value defined in the FileName property and exported date and time.

See also:
AppendDateTimeToFileName
FileName property

3.1.1.1.3 FileName

```
property FileName: string;
```

Description
The *FileName* property determines the name of the result export file. The property must not be empty; if it is empty at the moment of the Execute method invocation, the component will stop being executed having generated an exception with the following message:
Invalid File Name!.

See also:
AppendDateTimeToFileName property
PrintFile property
ShowFile property

3.1.1.1.4 PrintFile

```
property PrintFile: boolean;
```

Description
If the *PrintFile* property is *true*, then the result file will be sent to printing immediately after export.

See also:
FileName property
ShowFile property

3.1.1.1.5 Show File

```
property ShowFile: boolean;
```

Description
If the *ShowFile* property is *true*, then the result file will be opened in the associated application immediately after export.

See also:
FileName property
PrintFile property

3.1.2 TQExportCol object

Unit
QExport4

Description
The *TQExportCol* object is an element of the data row represented by the TQExportRow object. The *TQExportCol* object contains two properties - Name and Value.

3.1.2.1 Properties

Run-time only Key properties

- Name
- Value

3.1.2.1.1 Name

```
property Name: string;
```

Description
The *Name* property returns column name. This property is read-only.

3.1.2.1.2 Value

```
property Value: string;
```

Description
Use the *Value* property to change the column value in the current data row.

3.1.3 TQExportFormats object

Unit
QExport4

Description
The complex property *Formats* is used in all the descendant components and determines the type of the exported data. For representation of each data type there are string properties, such as *IntegerFormat*, *FloatFormat*, *CurrencyFormat* and *DateTimeFormat*, etc. The default settings of all the properties are those, specified in the operation system.
The settings of these properties will be applied to all the source fields of the corresponding types. To set a special format for a separate field (fields), use the UserFormats property.

3.1.3.1 Properties

Run-time only Key properties

- BooleanFalse
- BooleanTrue
- CurrencyFormat
- DateFormat
- DateTimeFormat
- FloatFormat
- IntegerFormat
- NullString
- TimeFormat

3.1.3.1.1 BooleanFalse

```
property BooleanFalse: string;
```

Description
Use property *BooleanFalse* to set the string value representing the *False* values of the source boolean fields in the result table, e.g. 'False' or '-'.

See also:
BooleanTrue property
ResetFormats method

3.1.3.1.2 BooleanTrue

```
property BooleanTrue: string;
```

Description
Use property *BooleanTrue* to set the string value representing the *True* values of the source boolean fields in the result table, e.g. 'True' or '+'.

See also:
BooleanFalse property
ResetFormats method

3.1.3.1.3 CurrencyFormat

```
property CurrencyFormat: string;
```

Description
The *CurrencyFormat* property determines the representation of currency fields in the result file. By default the value is set to default system format for currency.

See also:
FloatFormat property
IntegerFormat property
ResetFormats method

3.1.3.1.4 DateFormat

```
property DateFormat: string;
```

Description
The *DateFormat* property determines the representation of date fields in the result file. By default the value is set to default system format for dates.

See also:
TimeFormat property
DateTimeFormat property
ResetFormats method

3.1.3.1.5 DateTimeFormat

```
property DateTimeFormat: string;
```

Description
The *DateTimeFormat* property determines the representation of date/time fields in the result file. By default the value is set to default system format for date/time fileds. If you want to format time in twelve-hour form, you should add AM/PM to your format string.

Example:
mm/dd/yyyy hh:nn:ss AM/PM

See also:
DateFormat property
TimeFormat property
ResetFormats method

3.1.3.1.6 FloatFormat

```
property FloatFormat: string;
```

Description
The *FloatFormat* property determines the representation of float fields in the result file. By default the value is set to '#,###,##0.00'

See also:
CurrencyFormat property
IntegerFormat property
ResetFormats method

3.1.3.1.7 IntegerFormat

```
property IntegerFormat: string;
```

Description

The *IntegerFormat* property determines the representation of integer fields in the result file. By default the value is set to '#,###,##0'.

See also:

CurrencyFormat property
FloatFormat property
ResetFormats method

3.1.3.1.8 NullString

```
property NullString: string;
```

Description
Use property *NullString* to set the string value representing the Null values in the result table, e.g. 'Null' or '0'.

See also:
ResetFormats method

3.1.3.1.9 TimeFormat

```
property TimeFormat: string;
```

Description
The *TimeFormat* property determines the representation of the time fields in the result file. By default the value is set to default system format for time fields. If you want to format time in twelve-hour form, you should add AM/PM to your format string.

Example:
mm/dd/yyyy hh:nn:ss AM/PM

See also:
DateFormat property
DateTimeFormat property
ResetFormats method

3.1.3.1.10 KeepOriginalFormat

```
property KeepOriginalFormat: boolean;
```

Description
If the *KeepOriginalFormat* property is *true*, then system formatting masks will be used for internal conversions.

3.1.3.2 Methods

- Key methods
  - ResetFormats

3.1.3.2.1 ResetFormats

```
procedure ResetFormats;
```

Description
Use method *ResetFormats* to set all the *TQExportFormats* properties to their default values.

3.1.4 TQExportRow object

Unit
QExport4

Description
TQExportRow represents the current data row as a collection of TQExportCol objects. Use the Items property to access these objects by their indexes. This class is used by the OnBeforeExportRow event of the TQExport4 class to manage data during the export process.

3.1.4.1 Properties

- Run-time only
- Key properties

- Items

3.1.4.1.1 Items

```
property Items[Index: Integer] : TQExportCol;
```

Description
Use *Items* to access the TQExportCol objects by *Index*.

See also:
TQExportCol object

3.1.5 TExportedRecordEvent type

Unit
QExport4

```
type
  TExportedRecordEvent = procedure(Sender: TObject; RecNo: integer) of object;
```

Description
The *TExportedRecordEvent* type is used in OnExportedRecord and OnSkippedRecord events. Variable *RecNo* indicates the number of the exported or skipped record in accordance.

See also:
OnExportedRecord event
OnSkippedRecord event

3.1.6 TGetCellParamsEvent type

Unit
QExport4

```
type
  TGetCellParamsEvent = procedure(Sender: TObject; RecNo, ColNo: integer; const Value
var Align: TQExportColAlign; AFont: TFont; var Background: TColor) of object;
```

Description
The *TGetCellParamsEvent* type is the OnGetCellParams event type. Parameters *RecNo* and *ColNo* allow you to identify the exported record, parameter *Value* contains its value. Output parameter *Align* allows you to set the cell alignment (*ecaLeft*, *ecaCenter* or *ecaRight*), *AFont* sets the cell font, and *Background* sets the cell background color.

See also:
OnGetCellParams event

3.1.7 TGetExportTextEvent type

Unit
QExport4

```
type
  TGetExportTextEvent = procedure(Sender: TObject; ColNo: integer; var Text: string) o
```

Description
The *TGetExportTextEvent* type is the OnGetExportText event type. Variable *ColNo* indicates the source column from which the string is exported, and variable Text contains the string value which can be edited before inserting the string to file.

See also:
OnGetExportText event

3.1.8 TQExportStopEvent type

Unit
QExport4

```
type
  TQExportStopEvent = procedure(Sender: TObject; var CanContinue: boolean) of object;
```

Description
The *TQExportStopEvent* type is the OnStopExport event type which takes place after the Abort method is invoked. Use *CanContinue* to allow (*CanContinue=True*) or forbid (*CanContinue=False*) further export.

See also:
OnStopExport event

3.2 QExport4Access unit

Components
TADO_QExport4Access

3.3 QExport4ASCII unit

Components
TQExport4ASCII

3.4 QExport4DBF unit

Components
TQExport4DBF

3.5 QExport4Clipboard unit

Components
TQExport4Clipboard

3.6 QExport4CustomSource unit

Objects

3.6.1 TqeCustomColumn object

Unit
QExport4CustomSource

Description
The *TqeCustomColumn* class describes a column in the custom export source. This class contains such column parameters, as column name, caption, type, size and width.

3.6.1.1 Properties

Run-time only Key properties

- Caption
- ColumnName
- ColumnType
- Size
- Width

3.6.1.1.1 Caption

```
property Caption: string;
```

Description
Use the *Caption* property to define the column caption in the result document.

3.6.1.1.2 ColumnName

```
property ColumnName: string;
```

Description

The *ColumnName* property defines the column name. This name is used to access the column.

3.6.1.1.3 ColumnType

```
type
  TQExportColType = (ectInteger, ectBigint, ectFloat, ectCurrency, ectDate, ectTime, e
ectBoolean, ectUnknown);

property ColumnType: TQExportColType;
```

Description

The *ColumnType* property defines the type of data in the column which is used to format data. You can define format settings through the Formats property of the TQexport3 class or its descendant which uses this column.

3.6.1.1.4 Size

```
property Size: integer;
```

Description
The *Size* property is used to define the field size in the TQExport4SQL class.

3.6.1.1.5 Width

```
property Width: integer;
```

Description

The *Width* property defines the column length in symbols. This property is used in the TQExport4ASCII class.

3.6.2 TqeCustomColumns object

Unit
QExport4CustomSource

Description
The *TqeCustomColumns* is a collection of TqeCustomColumn objects. Use the Items property to access these objects by their indexes.

See also:
TqeCustomColumn object

3.6.2.1 Properties

- Run-time only
- Key properties

- Items

3.6.2.1.1 Items

```
property Items[Index: integer]: TqeCustomColumn;
```

Description
Use *Items* to access the TqeCustomColumn objects by *Index*.

3.7 QExport4Dialog unit

Components
TQExport4Dialog

Types
TQExportGetColonEvent
TQGetExportTextEvent
TQRecordExportedEvent

3.7.1 TQExportGetColonEvent type

Unit
QExport4Dialog

```
type
  TQExportGetColonEvent = procedure(Sender: TObject; Colon: TStrings) of object;
```

Description
The OnGetFooter and OnGetHeader event type. Using Colon you can add strings to the document footer or header respectively.

See also:
OnGetFooter event
OnGetHeader event

3.7.2 TQGetExportTextEvent type

Unit
QExport4Dialog

```
type
  TQGetExportTextEvent = procedure(Sender: TQExport4; ColNo: integer; var Text: string
```

Description
The *TGetExportTextEvent* type is the OnGetExportText event type. Variable *ColNo* indicates the source column from which the string is exported, and variable *Text* contains the string value which can be edited before inserting the string to file.

See also:
OnGetExportText event

3.7.3 TQRecordExportedEvent type

Unit
QExport4Dialog

```
type
  TQRecordExportedEvent = procedure(Sender: TQExport4; RecNo: integer) of object;
```

Description
The *TExportedRecordEvent* type is used in OnExportedRecord and OnSkippedRecord events. Variable *RecNo* indicates the number of the exported or skipped record in accordance.

See also:
OnExportedRecord event
OnSkippedRecord event

3.8 QExport4Docx unit

Components
TQExport4Docx

Objects
TDocxOptions
TDocxStripStyle
TDocxCellStyle

3.8.1 TDocxOptions object

Unit
QExport4Docx

Description
The *TQExport4DocxOptions* class contains the properties that set the parameters of the result document.

3.8.1.1 Properties

Run-time only Key properties

- HeaderStyle
- CaptionRowStyle
- DataStyle
- FooterStyle
- StripStyleType
- StripStylesList

3.8.1.1.1 HeaderStyle

```
property HeaderStyle: TDocxCellStyle;
```

Description
Use this property to define the document's header style.

3.8.1.1.2 CaptionRow Style

```
property CaptionRowStyle: TDocxCellStyle;
```

Description

The *CaptionRowStyle* property contains parameters which define the captions appearance in the result document, such as colors and font.

3.8.1.1.3 DataStyle

```
property DataStyle: TDocxCellStyle;
```

Description
The *DataStyle* property contains parameters which define the data appearance in the result document, such as colors and font.

3.8.1.1.4 FooterStyle

```
property FooterStyle: TDocxCellStyle;
```

Description
Use this property to define the document's footer style.

3.8.1.1.5 StripStyleType

```
type
  TMSStripStyleType = (ssNone, ssColumn, ssRow);

property StripStyleType: TMSStripStyleType;
```

Description

This property defines strips direction - horizontal or vertical. If *StripType* has the *stCol* value, the strips are vertical, if the *stRow* value, the strips are horizontal. If the *stNone* value is set, the strips are disabled.

3.8.1.1.6 StripStylesList

```
property StripStylesList: TDocxStripStyleList;
```

Description
Use this property to define the strips styles. The *StripStyles* property contains a collection of styles which would be used one by one.

3.8.2 TDocxStripStyle object

Unit
QExport4Docx

Description
StripStyle determines font size, color and other attributes, border and fill styles and more.

3.8.2.1 Properties

Run-time only Key properties

Options

3.8.2.1.1 Options

```
property Options: TDocxCellStyle;
```

Description

The *Options* property is "complex" and contains some subproperties - properties of the *TXlsxCellStyle* class.

3.8.3 TDocxCellStyle object

Unit
QExport4Docx

Description
The *TDocxCellStyle* class contains parameters for defining the text appearance which can be applied to some of the result document parts.

3.8.3.1 Properties

Run-time only Key properties

- Font
- BackgroundColor
- UseBackground
- Alignment

3.8.3.1.1 Font

```
property Font: TFont;
```

Description
Use the *Font* property to define text font parameters for the current style.

3.8.3.1.2 BackgroundColor

```
property BackgroundColor: TColor;
```

Description

Use the *BackgroudColor* property to define the background color in the current style. This color is applied only if the *UseBackground* property is *True*.

3.8.3.1.3 UseBackground

```
property UseBackground: boolean;
```

Description

The *UseBackground* option enables using the background color in the current style. The default value is *True*.

3.8.3.1.4 Alignment

```
type
  TMSCellAlignment = (caLeft, caRight, caCenter, caJustify);

property Alignment: TMSCellAlignment;
```

Description

The *Alignment* property defines the text alignment for the current style. The following values are available:
caLeft - text is left-justified
caRight - text is right-justified
caCenter - text is centered
caJustify - text is distributed over the paragraph width

3.9 QExport4HTML unit

Components
TQExport4HTML

Objects
TTableOptions
THTMLOptions
TQExportHTMLNavigation

Types
THTMLTemplate
TUsingCSS

3.9.1 TTableOptions object

Unit
QExport4HTML

Description
The *TTableOptions* class contains the properties which determine the table parameters in the result HTML document.

3.9.1.1 Properties

Run-time only Key properties

- AdvancedAttributes
- Border
- BorderColor
- CellPadding
- CellSpacing
- HeadersRowBgColor
- HeadersRowFontColor
- OddRowBgColor
- TableBgColor
- TableFontColor

3.9.1.1.1 AdvancedAttributes

```
property AdvancedAttributes: TStrings;
```

Description
Some additional attributes of the *Table* element. Their list can be found in the HTML format specification at http://www.w3.org.

3.9.1.1.2 Border

```
property Border: integer;
```

Description
The *Border* property corresponds to the *Border* attribute of the *Table* element and sets the width of the table border in pixels. The default setting of this property is 1.

See also:
BorderColor property

3.9.1.1.3 BorderColor

```
property BorderColor: TColor;
```

Description
The *BorderColor* property determines the color of the output table border. The default color is white. See Color Using Notes to set the colors correctly.

See also:
Border property
Color Using Notes

3.9.1.1.4 CellPadding

```
property CellPadding: integer;
```

Description
The *CellPadding* property corresponds to the *CellPadding* attribute of the *Table* element and sets the space between the cell border and its contents. Together with the *CellSpacing* attribute it manages the process of displaying table cells.

See also:
CellSpacing property

3.9.1.1.5 CellSpacing

```
property CellSpacing: integer;
```

Description
The *CellSpacing* property corresponds to the *Cellspacing* attribute of the *Table* element. This attribute sets the space (in pixels) that the user's agent (browser) must leave between the left side of the table and the left edge of the first column on the left, the top border of the table and the top edge of the very top line, and the same for the right and the bottom borders of the table. This attribute also sets the space between cells.

See also:
CellPadding property

3.9.1.1.6 HeadersRowBgColor

```
property HeadersRowBgColor: TColor;
```

Description
The *HeadersRowBgColor* property sets the background color in the table column headers (the default headers are the source field names; they can be changed by setting the Captions property). See Color Using Notes to set the colors correctly.

See also:
HeadersRowFontColor property
OddRowBgColor property
TableBgColor property
Color Using Notes

3.9.1.1.7 HeadersRow FontColor

```
property HeadersRowFontColor: TColor;
```

Description
The *HeadersRowFontColor* property sets the font color in the table column headers (the default headers are the source field names; they can be changed by setting the Captions property). See Color Using Notes to set the colors correctly.

See also:
HeadersRowBgColor property
TableFontColor property
Color Using Notes

3.9.1.1.8 OddRow BgColor

```
property OddRowBgColor: TColor;
```

Description
The *OddRowBgColor* property sets the background color for the odd table rows. See Color Using Notes to set the colors correctly.

See also:
HeadersRowBgColor property
TableBgColor property
Color Using Notes

3.9.1.1.9 TableBgColor

```
property TableBgColor: TColor;
```

Description
The *TableBgColor* property sets the background color for the even table rows. See Color Using Notes to set the colors correctly.

See also:
HeadersRowBgColor property
OddRowBgColor property
TableFontColor property
Color Using Notes

3.9.1.1.10 TableFontColor

```
property TableFontColor: TColor;
```

Description
The *TableFontColor* property sets the font color for *all* table rows. See Color Using Notes to set the colors correctly.

See also:
HeadersRowFontColor property
TableBgColor property
Color Using Notes

3.9.1.1.11 BackgroundFileName

```
property BackgroundFileName: string;
```

Description

The *BackgroundFileName* property specifies the file with the background picture for the table.

3.9.2 THTMLOptions object

Unit
QExport4HTML

Description
The class contains properties that set the parameters of the result HTML document.

3.9.2.1 Properties

Run-time only Key properties

- AdvancedAttributes
- ALinkColor
- BackgroundColor
- BackgroundFileName
- DefaultOptions
- LinkColor
- TextFont
- VLinkColor

3.9.2.1.1 LinkColor

```
property LinkColor: TColor;
```

Description
The *LinkColor* property corresponds to the *Link* attribute of the *BODY* element and sets the color of the "regular" (not having been visited and not being clicked on at the given moment) hyperlinks of the result document. See Color Using Notes to set the colors correctly.

See also:
ALinkColor property
VLinkColor property
Color Using Notes

3.9.2.1.2 AdvancedAttributes

```
property AdvancedAttributes: TStrings;
```

Description
Some additional attributes of the *Body* element. Their list can be found in the HTML format specification at http://www.w3.org.

3.9.2.1.3 ALinkColor

```
property ALinkColor: TColor;
```

Description
The *ALinkColor* property corresponds to the *ALink* attribute of the *BODY* element and sets the color of the "active" hyperlinks (when the mouse pointer is over the link) of the result document. See Color Using Notes to set the colors correctly.

See also:
LinkColor property
VLinkColor property
Color Using Notes

3.9.2.1.4 BackgroundColor

```
property BackgroundColor: TColor;
```

Description
The *BackgroundColor* property sets the background color of the result HTML document (but not of the table). See Color Using Notes to set the colors correctly.

See also:
Background property
Color Using Notes

3.9.2.1.5 BackgroundFileName

```
property BackgroundFileName: string;
```

Description

The *BackgroundFileName* property sets the background image of the result HTML document. If an empty string or not existing file name is specified as a property value, the document won't have a background image. When setting this property, it is recommended to specify a relative file path (for example, *./Img/html.jpg*), that will let you change the location of your documents without making any changes to the HTML code. It is also recommended to select the background image of the page close to color scheme to the background of the document (see BackgroundColor), otherwise the users that have the image display mode turned off can get not a very correct image of your design.

See also:

BackgroundColor property

3.9.2.1.6 DefaultOptions

```
property DefaultOptions: TDefaultOptions;
```

Description
This property determines the usage of the default font name and size in the result HTML document. If this property includes *doFontName* (*DefaultOptions.doFontName=True*) then the default font name will be used; if it includes *doFontSize* (*DefaultOptions.doFontSize=True*), then the default font size will be used. It may also include both of none of them.

See also:
TextFont property

3.9.2.1.7 TextFont

```
property TextFont: TFont;
```

Description
The *TextFont* property sets the common font of the HTML document. When selecting the font name, don't forget that in case the selected font is missing in the user system, the browser will automatically select the default font which can affect the design of your document. That is why it is better to use only those fonts that are present at all the platforms popular nowadays.

See also:
DefaultOptions property

3.9.2.1.8 VLinkColor

```
property VLinkColor: TColor;
```

Description
The *VLinkColor* property corresponds to the *VLink* attribute of the *BODY* element and sets the color of the "visited" hyperlinks of the output document, i.e. of the links that had already been clicked on (Visited Link). See Color Using Notes to set the colors correctly.

See also:
ALinkColor property
LinkColor property

3.9.3 TQExportHTMLNavigation object

Unit
QExport4HTML

Description
The *TQExportNavigation* object allows you to customize the multi-file export to HTML. It contains properties for defining the number and the appearance of the navigation links in the result HTML documents.

3.9.3.1 Properties

Run-time only Key properties

- FirstLinkTitle
- IndexLinkTemplate
- IndexLinkTitle
- LastLinkTitle
- NextLinkTitle
- OnBottom
- OnTop
- PriorLinkTitle

3.9.3.1.1 FirstLinkTitle

```
property FirstLinkTitle: string;
```

Description
Use *FirstLinkTitle* to define the caption of the link navigating to the first document of the collection.

See also:
IndexLinkTitle property
LastLinkTitle property
NextLinkTitle property
PriorLinkTitle property

3.9.3.1.2 IndexLinkTemplate

```
property IndexLinkTemplate: string;
```

Description

The *IndexLinkTemplate* property defines the template string for generating links on the index page to other pages in the collection.
E.g. if IndexLinkTemplate='part' then the index page will contain the following links: "part1", "part2", "part3", etc.

3.9.3.1.3 IndexLinkTitle

```
property IndexLinkTitle: string;
```

Description
Use *IndexLinkTitle* to define the caption of the link navigating to the "home" ("Index") page of the collection.

See also:
FirstLinkTitle property
LastLinkTitle property
NextLinkTitle property
PriorLinkTitle property

3.9.3.1.4 LastLinkTitle

```
property LastLinkTitle: string;
```

Description
Use *LastLinkTitle* to define the caption of the link navigating to the last document of the collection.

See also:
FirstLinkTitle property
IndexLinkTitle property
NextLinkTitle property
PriorLinkTitle property

3.9.3.1.5 NextLinkTitle

```
property NextLinkTitle: string;
```

Description
Use *NextLinkTitle* to define the caption of the link navigating to the next document in the collection.

See also:
FirstLinkTitle property
IndexLinkTitle property
LastLinkTitle property
PriorLinkTitle property

3.9.3.1.6 OnBottom

```
property OnBottom: boolean;
```

Description
The *OnBottom* property defines if there are navigation links (First, Prior, Next, Last) on the bottom of each page in the collection.

See also:
OnTop property

3.9.3.1.7 OnTop

```
property OnTop: boolean;
```

Description
The *OnTop* property defines if there are navigation links (First, Prior, Next, Last) on the top of each page in the collection.

See also:
OnBottom property

3.9.3.1.8 PriorLinkTitle

```
property PriorLinkTitle: string;
```

Description
Use *PriorLinkTitle* to define the caption of the link navigating to the previous document in the collection.

See also:
FirstLinkTitle property
IndexLinkTitle property
LastLinkTitle property
NextLinkTitle property

3.9.4 THTMLTemplate type

Unit
QExport4HTML

```
type
  THTMLTemplate = (htCustom, htBW, htClassic, htColorFul, htGray, htMS_Money, htMurky,
```

Description
Defines the pre-defined templates of the result files (the settings of the HTMLOptions and TableOptions properties). Along with the standard templates, you can create your own using the SaveTemplateToFile and LoadTemplateFromFile methods.

See also:
HTMLTemplate property

3.9.5 TUsingCSS type

Unit
QExport4HTML

```
type
  TUsingCSS = (usInternal, usExternal);
```

Description
The variants of positioning the HTML file style table (internally [default] and externally accordingly).

See also:
UsingCSS property

3.10 QExport4LaTeX unit

Components
TLaTeXOptions
TQExport4LaTeX

3.10.1 TLaTeXOptions component

Unit
QExport4LaTeX

Description
The *TLaTeXOptions* class contains the properties which determine the parameters of the result document.

3.10.1.1 Properties

Run-time only Key properties

- CodePage
- DocumentParams
- DocumentStyle
- Languages
- LaTeXVersion

3.10.1.1.1 CodePage

```
property CodePage: integer;
```

Description
The *CodePage* property determines in which encoding LaTeX will perceive the exported file. The default setting of this property is equal to the active code page.

See also:
DocumentParams property
DocumentStyle property
Languages property
LaTeXVersion property

3.10.1.1.2 DocumentParams

```
property DocumentParams: string;
```

Description

The *DocumentParams* property sets not required parameters of the output document which are placed in the (2e) or (2.09) commands. The default setting of this property is *a4paper* which shows to LaTeX that the document will be printed on paper of A4 format. For more info see TUG.

See also:

CodePage property
DocumentStyle property
Languages property
LaTeXVersion property

3.10.1.1.3 DocumentStyle

```
property DocumentStyle: TLaTeXDocStyle;
```

Description
The *DocumentStyle* property sets the style of the LaTeX document. The following values are available: dsArticle - article style (default) and dsBook - book style. The corresponding string with the same name is placed in the (2e) or (2.09) command.

See also:
CodePage property
DocumentParams property
Languages property
LaTeXVersion property

3.10.1.1.4 Languages

```
property Languages: string;
```

Description
The *Languages* property determines the languages which will be used in the result file. The default is English. To add other languages, just type their English names through a comma. For example, for the Russian language support the property should have the setting "english,russian" (without quotes).

See also:
CodePage property
DocumentParams property
DocumentStyle property
LaTeXVersion property

3.10.1.1.5 LaTeXVersion

```
property LaTeXVersion: TLaTeXVersion;
```

Description
The *LaTeXVersion* property determines the version of the output file. The following values are available: *LaTeX209* and *LaTeX2e* (default).

See also:
CodePage property
DocumentParams property
DocumentStyle property
Languages property

3.11 BaseODSClass4 unit

Objects

TODSCellParagraphStyle
TODSParagraphStyle
TODFBorder

3.11.1 TODSCellParagraphStyle

```
TODSCellParagraphStyle = class(TODSParagraphStyle)
```

Unit
BaseODFClass4

Description
The *TODSCellParagraphStyle* class contains parameters for defining the text appearance which can be applied to some of the result document parts.

3.11.1.1 Properties

Run-time only Key properties

- VerticalAligment
- Border

3.11.1.1.1 VerticalAligment

```
type
  TODFTextVerticalAligment = (taODFTop, taODFMiddle, taODFBottom);

property VerticalAligment: TODFTextVerticalAligment;
```

Description

The *Alignment* property defines the text alignment for the current style. The following values are available:
taODFTop - text is top-justified
taODFMiddle - text is middle-justified
taODFBottom - text is bottom-justified

3.11.1.1.2 Border

```
property Border: TODFBorder;
```

Description

The *Borders* property defines the appearance of the cell borders.

3.11.2 TODSParagraphStyle

Unit
BaseODFClass4

Description
The *TODSParagraphStyle* class contains parameters for defining the text appearance which can be applied to some of the result document parts.

3.11.2.1 Properties

Run-time only Key properties

- Font
- BackgroundColor
- AllowBackground
- Alignment

3.11.2.1.1 Font

```
property Font: TFont;
```

Description
Use the *Font* property to define text font parameters for the current style.

3.11.2.1.2 BackgroundColor

```
property BackgroundColor: TColor;
```

Description

Use the *BackgroudColor* property to define the background color in the current style. This color is applied only if the *AllowBackground* property is *True*.

3.11.2.1.3 Allow Background

```
property AllowBackground: boolean;
```

Description
The *AllowBackground* option enables using the background color in the current style. The default value is *True*.

3.11.2.1.4 Alignment

```
type
  TODFTextAlignment = (taODFLeft, taODFRight, taODFCenter, taODFJustify);

property Alignment: TODFTextAlignment;
```

Description

The *Alignment* property defines the text alignment for the current style. The following values are available:
taODFLeft - text is left-justified
taODFRight - text is right-justified
taODFCenter - text is centered
taODFJustify - text is distributed over the paragraph width

3.11.3 TODFBorder

Unit
BaseODFClass4

Description
The *TODFBorder* object defines the appearance style of the cell borders.

3.11.3.1 Properties

Run-time only Key properties

- BorderStyle
- BorderWidth
- BorderColor

3.11.3.1.1 BorderStyle

```
type
  TODFBorderStyle = (bsODFNone, bsODFSolid);

property BorderStyle: TODFBorderStyle;
```

Description
The *BorderStyle* property defines the style of the cell borders.

3.11.3.1.2 BorderWidth

```
property BorderWidth: Integer;
```

Description
The *BorderWidth* property defines the width of the cell borders.

3.11.3.1.3 BorderColor

```
property BorderColor: TColor;
```

Description
The *BorderColor* property defines the color of the cell borders.

3.12 BaseODTClass4 unit

Objects

TODTCellParagraphStyle
TODTParagraphStyle
TODFBorder

3.12.1 TODTCellParagraphStyle

```
TODTCellParagraphStyle = class(TODTParagraphStyle)
```

Unit
BaseODFClass4

Description
The *TODTCellParagraphStyle* class contains parameters for defining the text appearance which can be applied to some of the result document parts.

3.12.1.1 Properties

Run-time only Key properties

- VerticalAligment
- Border

3.12.1.1.1 VerticalAligment

```
type
  TODFTextVerticalAligment = (taODFTop, taODFMiddle, taODFBottom);

property VerticalAligment: TODFTextVerticalAligment;
```

Description

The *Alignment* property defines the text alignment for the current style. The following values are available:
taODFTop - text is top-justified
taODFMiddle - text is middle-justified
taODFBottom - text is bottom-justified

3.12.1.1.2 Border

```
property Border: TODFBorder;
```

Description
The *Borders* property defines the appearance of the cell borders.

3.12.2 TODTParagraphStyle

```
TODTParagraphStyle = class(TODSParagraphStyle):
```

Unit
BaseODFClass4

Description
The *TODTParagraphStyle* class contains parameters for defining the text appearance which can be applied to some of the result document parts.

3.12.2.1 Properties

Run-time only Key properties

- HighlightColor
- AllowHighlight

3.12.2.1.1 HighlightColor

```
property HighlightColor: TColor;
```

Description

Use the *HighlightColor* property to define the highlight color in the current style. This color is applied only if the *AllowHighlight* property is *True*.

3.12.2.1.2 Allow Highlight

```
property AllowHighlight: Boolean;
```

Description
The *AllowHighlight* option enables using the highlight color in the current style. The default value is *False*.

3.12.3 TODFBorder

Unit
BaseODFClass4

Description
The *TODFBorder* object defines the apeearance style of the cell borders.

3.12.3.1 Properties

Run-time only Key properties

- BorderStyle
- BorderWidth
- BorderColor

3.12.3.1.1 BorderStyle

```
type
  TODFBorderStyle = (bsODFNone, bsODFSolid);

property BorderStyle: TODFBorderStyle;
```

Description
The *BorderStyle* property defines the style of the cell borders.

3.12.3.1.2 BorderWidth

```
property BorderWidth: Integer;
```

Description
The *BorderWidth* property defines the width of the cell borders.

3.12.3.1.3 BorderColor

```
property BorderColor: TColor;
```

Description
The *BorderColor* property defines the color of the cell borders.

3.13 QExport4ODS unit

Components
TQExport4ODS

Objects
TODSOptions
TODSCellParagraphStyle
TODSParagraphStyle
TODFBorder

3.13.1 TQExport4ODSOptions object

Unit
QExport4ODS

Description
The *TQExportODSOptions* class contains the properties that set the parameters of the result document.

3.13.1.1 Properties

Run-time only Key properties

- HeaderStyle
- FooterStyle
- CaptionRowStyle
- DataStyle
- StripStyle
- StripStylesList

3.13.1.1.1 HeaderStyle

```
property HeaderStyle: TODSCellParagraphStyle;
```

Description
Use this property to define the document's header style.

3.13.1.1.2 FooterStyle

```
property FooterStyle: TODSCellParagraphStyle;
```

Description
Use this property to define the document's footer style.

3.13.1.1.3 CaptionRow Style

```
property CaptionRowStyle: TODSCellParagraphStyle;
```

Description

The *CaptionRowStyle* property contains parameters which define the captions appearance in the result document, such as colors and font.

3.13.1.1.4 DataStyle

```
property DataStyle: TODSCellParagraphStyle;
```

Description
The *DataStyle* property contains parameters which define the data appearance in the result document, such as colors and font.

3.13.1.1.5 StripStyle

```
type
  TODFStripStyleType = (sstNone, sstColumn, sstRow);

property StripStyle: TODFStripStyleType;
```

Description

This property defines strips direction - horizontal or vertical. If *StripType* has the *stCol* value, the strips are vertical, if the *stRow* value, the strips are horizontal. If the *stNone* value is set, the strips are disabled.

3.13.1.1.6 StripStylesList

```
property StripStylesList: TODSStylesList;
```

Description
Use this property to define the strips styles. The *StripStyles* property contains a collection of styles which would be used one by one.

3.14 QExport4ODT unit

Components
TQExport4ODT

Objects
ODTOptions
TODTCellParagraphStyle
TODTParagraphStyle
TODFBorder

3.14.1 TQExportODTOptions object

Unit
QExport4ODT

Description
The *TQExportODTOptions* class contains the properties that set the parameters of the result document.

3.14.1.1 Properties

Run-time only Key properties

- HeaderStyle
- FooterStyle
- CaptionRowStyle
- DataStyle
- StripStyle
- StripStylesList

3.14.1.1.1 HeaderStyle

```
property HeaderStyle: TODTParagraphStyle;
```

Description
Use this property to define the document's header style.

3.14.1.1.2 FooterStyle

```
property FooterStyle: TODTParagraphStyle;
```

Description
Use this property to define the document's footer style.

3.14.1.1.3 CaptionRow Style

```
property CaptionRowStyle: TODTCellParagraphStyle;
```

Description

The *CaptionRowStyle* property contains parameters which define the captions appearance in the result document, such as colors and font.

3.14.1.1.4 DataStyle

```
property DataStyle: TODTCellParagraphStyle;
```

Description
The *DataStyle* property contains parameters which define the data appearance in the result document, such as colors and font.

3.14.1.1.5 StripStyle

```
type
  TODFStripStyleType = (sstNone, sstColumn, sstRow);

property StripStyle: TODFStripStyleType;
```

Description

This property defines strips direction - horizontal or vertical. If *StripType* has the *stCol* value, the strips are vertical, if the *stRow* value, the strips are horizontal. If the *stNone* value is set, the strips are disabled.

3.14.1.1.6 StripStylesList

```
property StripStylesList: TODTStylesList;
```

Description
Use this property to define the strips styles. The *StripStyles* property contains a collection of styles which would be used one by one.

3.15 QExport4PDF unit

Components
TQExport4PDF

Objects
TPDFFont
TPDFOptions

3.15.1 TPDFFont object

Unit
QExport4PDF

Description
The *TPDFFont* object contains properties for defining various fonts for use in the result PDF file.

3.15.1.1 Properties

BaseFont
FontColor
FontEncoding
FontSize
Charset

3.15.1.1.1 BaseFont

```
type
  TPDFFontName = (poHelvetica, poHelveticaBold, poHelveticaOblique, poHelveticaBoldOb
poCourierBold, poCourierOblique, poCourierBoldOblique, poTimesRoman, poTimesBold, poT
poTimesBoldItalic, poSymbol, poZapfDingbats);

property BaseFont: TPDFFontName;
```

Description
The *BaseFont* property defines the name of the font in the result PDF document.

See also:
FontColor property
FontEncoding property
FontSize property

3.15.1.1.2 FontColor

```
property FontColor: TColor;
```

Description
The *FontColor* property defines the color of the customized PDF font.

See also:
BaseFont property
FontEncoding property
FontSize property

3.15.1.1.3 FontEncoding

```
type
  TPDFFontEncoding = (poStandardEncoding, poWinAnsiEncoding, poMacRomanEncoding, poPDF

property FontEncoding: TPDFFontEncoding;
```

Description
The *FontEncoding* property is used for setting the character encoding in the result PDF document.

See also:
BaseFont property
FontColor property
FontSize property

3.15.1.1.4 FontSize

```
property FontSize: integer;
```

Description
The *FontSize* property defines the size of the customized PDF font.

See also:
BaseFont property
FontColor property
FontEncoding property

3.15.1.1.5 Charset

```
property Charset: TFontCharset;
```

Description
The *Charset* property defines charset in which the text is exported.

3.15.2 TPDFOptions object

Unit
QExport4PDF

Description
The *TPDFOptions* class contains properties for defining the parameters of the result PDF document.

3.15.2.1 Properties

Run-time only Key properties

- CaptionFont
- ColSpacing
- DataFont
- FooterFont
- GridLineColor
- GridLineWidth
- HeaderFont
- RowSpacing

3.15.2.1.1 CaptionFont

```
property CaptionFont: TPDFFont;
```

Description
Use *CaptionFont* to define the properties of the font for the result PDF captions.

See also:
DataFont property
FooterFont property
HeaderFont property

3.15.2.1.2 ColSpacing

```
property ColSpacing: double;
```

Description
The *ColSpacing* property defines the column spacing in the result PDF table.

See also:
RowSpacing property

3.15.2.1.3 DataFont

```
property DataFont: TPDFFont;
```

Description
Use *DataFont* to define the properties of the font for displaying data in the result PDF table.

See also:
CaptionFont property
FooterFont property
HeaderFont property

3.15.2.1.4 FooterFont

```
property FooterFont: TPDFFont;
```

Description
Use *FooterFont* to define the properties of the font for the result PDF footers.

See also:
CaptionFont property
DataFont property
HeaderFont property

3.15.2.1.5 GridLineColor

```
property GridLineColor: TColor;
```

Description
The *GridLineColor* property defines the color of the grid lines in the result PDF table.

See also:
GridLineWidth property

3.15.2.1.6 GridLineWidth

```
property GridLineWidth: integer;
```

Description
The *GridLineWidth* property defines the width of the grid lines in the result PDF table.

See also:
GridLineColor property

3.15.2.1.7 HeaderFont

```
property HeaderFont: TPDFFont;
```

Description
Use *HeaderFont* to define the properties of the font for the result PDF headers.

See also:
CaptionFont property
DataFont property
FooterFont property

3.15.2.1.8 Row Spacing

```
property RowSpacing: double;
```

Description
The *RowSpacing* property defines the row spacing in the result PDF table.

See also:
ColSpacing property

3.16 QExport4RTF unit

Components
TQExport4RTF

Objects
TRTFOptions
TrtfStyle
TrtfStyles

3.16.1 TRTFOptions object

Unit
QExport4RTF

Description
The *TRTFOptions* class contains the properties which determine the parameters of the result RTF document.

3.16.1.1 Properties

Run-time only Key properties

- PageOrientation
- CaptionAligns
- CaptionStyle
- DataStyle
- FooterStyle
- HeaderStyle
- StripStyles
- StripType

3.16.1.1.1 PageOrientation

```
type
  TQExportPageOrientation = (poPortrait, poLandscape);

property PageOrientation: TQExportPageOrientation;
```

Description
The *PageOrientation* property defines the orientation of the result document. The following values are available:
poPortrait - vertical page orientation
poLandscape - horizontal page orientation

3.16.1.1.2 CaptionAligns

```
property CaptionAligns: TStrings;
```

Description
The *CaptionAligns* property defines the captions' alignments for the columns. The alignments are stored as strings in the "<Name> = <Alignment>" format.

Example:

```
Column1 = Left
Column2 = Center
Column3 = Right
```

See also:
CaptionStyle property

3.16.1.1.3 CaptionStyle

```
property CaptionStyle: TrtfStyle;
```

Description
The *CaptionStyle* property contains parameters which define the captions appearance in the result document, such as colors and font.

See also:
TrtfStyle object

3.16.1.1.4 DataStyle

```
property DataStyle: TrtfStyle;
```

Description
The *DataStyle* property contains parameters which define the data appearance in the result document, such as colors and font.

See also:
TrtfStyle object

3.16.1.1.5 FooterStyle

```
property FooterStyle: TrtfStyle;
```

Description
Use this property to define the document's footer style.

See also:
TrtfStyle object

3.16.1.1.6 HeaderStyle

```
property HeaderStyle: TrtfStyle;
```

Description
Use this property to define the document's header style.

See also:
TrtfStyle object

3.16.1.1.7 StripStyles

```
property StripStyles: TrtfStyles;
```

Description
Use this property to define the strips styles. The *StripStyles* property contains a collection of styles which would be used one by one.

See also:
TrtfStyles object
StripType property

3.16.1.1.8 StripType

```
type
  TrtfStripType = (stNone, stCol, stRow);

property StripType: TrtfStripType;
```

Description
This property defines strips direction - horizontal or vertical. If *StripType* has the *stCol* value, the strips are vertical, if the *stRow* value, the strips are horizontal. If the *stNone* value is set, the strips are disabled.

See also:
StripStyles property

3.16.2 TrtfStyle object

Unit
QExport4RTF

Description
The *TrtfStyle* class contains parameters for defining the text appearance which can be applied to some of the result document parts.

3.16.2.1 Properties

Run-time only Key properties

- Alignment
- AllowBackground
- AllowHighlight
- BackgroundColor
- Font
- HighlightColor

3.16.2.1.1 Alignment

```
type
  TrtfTextAlignment = (palLeft, palRight, palCenter, palFill);

property Alignment: TrtfTextAlignment;
```

Description

The *Alignment* property defines the text alignment for the current style. The following values are available:
palLeft - text is left-justified
palRight - text is right-justified
palCenter - text is centered
palFill - text is distributed over the paragraph width

3.16.2.1.2 Allow Background

```
property AllowBackground: boolean;
```

Description
The *AllowBackground* option enables using the background color in the current style. The default value is *True*.

See also:
BackgroundColor property

3.16.2.1.3 Allow Highlight

```
property AllowHighlight: boolean;
```

Description
The *AllowHighlight* option enables highlighting text in the current style. The default value is *False*.

See also:
HighlightColor property

3.16.2.1.4 BackgroundColor

```
property BackgroundColor: TColor;
```

Description
Use the *BackgroudColor* property to define the background color in the current style. This color is applied only if the AllowBackground property is *True*.

See also:
AllowBackground property

3.16.2.1.5 Font

```
property Font: TFont;
```

Description
Use the *Font* property to define text font parameters for the current style.

3.16.2.1.6 HighlightColor

```
property HighlightColor: TColor;
```

Description
Use the *HighlightColor* property to define the color of the text highlighting in the current style. This color is applied only if the AllowHighlight property is *True*.

See also:
AllowHighlight property

3.16.3 TrtfStyles object

Unit
QExport4RTF

Description
The *TrtfStyles* is a collection of the TrtfStyle objects. Use the Items property to access these objects by their indexes.

3.16.3.1 Properties

- Run-time only
- Key properties

- Items

3.16.3.1.1 Items

```
property Items[Index: integer] : TrtfStyle;
```

Description

Use *Items* to access the TrtfStyle objects by *Index*.

3.17 QExport4SQL unit

Components
TQExport4SQL

3.18 QExport4XLS unit

Components
TQExport4XLS

Objects
TxlsFormat
TxlsFormats
TxlsFieldFormat
TxlsFieldFormats
TxlsFont
TxlsBorder
TxlsBorders
TxlsFill
TxlsAlignment
TxlsHyperLink
TxlsHyperLinks
TxlsNote
TxlsNotes
TxlsChartSeries
TxlsChartSeriesList
TxlsChart
TxlsCharts
TxlsPicture
TxlsPictures
TxlsImage
TxlsImages
TxlsCell
TxlsCells
TxlsMergedCells
TxlsMergedCellList
TxlsNoteFormat
TxlsDataRange
TxlsChartPosition
TXLSOptions

Types
TxlsColor
TxlsBorderStyle
TxlsPattern

TGetAggregateParamsEvent
TGetCaptionParamsEvent
TGetHeaderFooterParamsEvent
TGetDataParamsEvent

3.18.1 TxlsFormat object

Unit
QExport4XLS

Description
TxlsFormat describes the style characteristics used when displaying text in cells. *TxlsFormat* defines font (name, style, size etc.), borders (left, right, top, bottom etc), fill and alignment (horizontal and vertical).

See also:
TxlsFormats object

3.18.1.1 Properties

Run-time only Key properties

- Alignment
- Borders
- Fill
- Font
- Wrap

3.18.1.1.1 Alignment

```
property Alignment: TxlsAlignment;
```

Description
The *Alignment* property defines the text alignment in the cells.

See also:
TxlsAlignment object
Borders property
Fill property
Font property
Wrap property

3.18.1.1.2 Borders

```
property Borders: TxlsBorders;
```

Description
The *Borders* property defines the appearance of the cell borders.

See also:
TxlsBorders object
Alignment property
Fill property
Font property
Wrap property

3.18.1.1.3 Fill

```
property Fill: TxlsFill;
```

Description
The *Fill* property defines the cell filling.

See also:
TxlsFill object
Alignment property
Borders property
Font property
Wrap property

3.18.1.1.4 Font

```
property Font: TxlsFont;
```

Description
The *Font* property defines the cell font.

See also:
TxlsFont object
Alignment property
Borders property
Fill property
Wrap property

3.18.1.1.5 Wrap

```
property Wrap: boolean;
```

Description
If the *Wrap* property is *true*, text is automatically wrapped in the cells.

See also:
Alignment property
Borders property
Fill property
Font property

3.18.2 TxlsFormats object

Unit
QExport4XLS

Description
The *TxlsFormats* is a collection of TxlsFormat objects. Use property Items to access these objects by their indexes.

See also:
TxlsFormat object

3.18.2.1 Properties

- Run-time only
- Key properties

- Items

3.18.2.1.1 Items

```
property Items[Index: integer]: TxlsFormat;
```

Description

Use *Items* to access the TxlsFormat objects by *Index*.

3.18.3 TxlsFieldFormat object

Unit
QExport4XLS

Description
TxlsFieldFormat inherites from TxlsFormat. It allows you to set the format of the certain table column by setting the source field name in the FieldName property. Along with all the settings, available in *TxlsFormat*, you can set the column width (Width property) and/or add one of four aggregate functions below the column: sum, average, minimum or maximum value of the column.

See also:
TxlsFieldFormats object
TxlsFormat object

3.18.3.1 Properties

Run-time only Key properties

- Aggregate
- FieldName
- Width

3.18.3.1.1 Aggregate

```
property Aggregate: TxlsAggregate;
```

Description
Use *Aggregate* to add an aggregate function to the cell below the column. The following values are available:
aggNone - no function
aggSum - sum of the column values
aggAvg - average column value
aggMin - minimum value of the column
aggMax - maximum value of the column

The default value is *aggNone*.

3.18.3.1.2 FieldName

```
property FieldName: string;
```

Description
The *FieldName* property defines the formatted source field.

3.18.3.1.3 Width

```
property Width: integer;
```

Description
Use *Width* to set the column width in symbols.

3.18.4 TxlsFieldFormats object

Unit
QExport4XLS

Description
The *TxlsFieldFormats* is a collection of TxlsFieldFormat objects. Use property Items to access these objects by their indexes.

See also:
TxlsFieldFormat object

3.18.4.1 Properties

- Run-time only
- Key properties
- Items

3.18.4.1.1 Items

```
property Items[Index: integer]: TxlsFieldFormat;
```

Description
Use *Items* to access the TxlsFieldFormat objects by *Index*.

3.18.5 TxlsFont object

Unit
QExport4XLS

Description
The *TxlsFont* class contains properties which define font in the result Excel cells.

3.18.5.1 Properties

Run-time only Key properties

- Charset
- Color
- Name
- Script
- Size
- Style
- Underline

3.18.5.1.1 Charset

```
property Charset: TFontCharset;
```

Description
The *CharSet* property defines the font charset. Default value is 1 (DEFAULT_CHARSET).

See also:
Color property
Name property
Script property
Size property
Style property
Underline property

3.18.5.1.2 Color

```
property Color: TxlsColor;
```

Description
The *Color* property defines the font color. Default color is *clrBlack*. See TxlsColor type for details.

See also:
TxlsColor type
Charset property
Name property
Script property
Size property
Style property
Underline property

3.18.5.1.3 Name

```
property Name: TFontName;
```

Description
The *Name* property defines the font name. Default value is 'Arial'.

See also:
Charset property
Color property
Script property
Size property
Style property
Underline property

3.18.5.1.4 Script

```
property Script: TxlsFontScript;
```

Description
The *Script* property defines the font script. The following values are available:
fscNone - no script
fscSuperScript - superscript
fscSubScript - subscript

The default value is *fscNone*.

See also:
Charset property
Color property
Name property
Size property
Style property
Underline property

3.18.5.1.5 Size

```
property Size: integer;
```

Description
The *Size* property sets the font size in points. The default value is 10.

See also:
Charset property
Color property
Name property
Script property
Style property
Underline property

3.18.5.1.6 Style

```
property Style: TxlsFontStyles;
```

Description
The *Style* property defines the font style. The value of the property is a set of *TxlsFontStyle*, where the following values are available: *xfsBold*, *xfsItalic*, and *xfsStrikeOut*. The default value is empty set. In CaptionsFormat the default value of this property is *[xfsBold]*.

See also:
Charset property
Color property
Name property
Script property
Size property
Underline property

3.18.5.1.7 Underline

```
property Underline: TxlsFontUnderline;
```

Description
The *Underline* property defines the font underlining. The following values are available: *fulNone*, *fulSingle*, *fulDouble*, *fulSingleAccounting*, *fulDoubleAccounting*. The default value is *fulNone*.

See also:
Charset property
Color property
Name property
Script property
Size property
Style property

3.18.6 TxlsBorder object

Unit
QExport4XLS

Description
The *TxlsBorder* class contains properties which define the color and the style of the cell borders in the result Excel table.

3.18.6.1 Properties

Run-time only Key properties

- Color
- Style

3.18.6.1.1 Color

```
property Color: TxlsColor;
```

Description
The *Color* property defines the border color. Default value is *clrBlack*. See TxlsColor type for details.

See also:
TxlsColor type
Style property

3.18.6.1.2 Style

```
property Style: TxlsBorderStyle;
```

Description
The *Style* property defines the border style. Default value is *bstNone*. See TxlsBorderStyle type for details.

See also:
TxlsBorderStyle type
Color property

3.18.7 TxlsBorders object

Unit
QExport4XLS

Description
Each property of the *TxlsBorders* object correspond to the separate border of the Excel cell (bottom, left, etc.). The values of these properties are of the same type - TxlsBorder - which allows you to set the appearance of the certain border.

3.18.7.1 Properties

Run-time only Key properties

- Bottom
- DiagDown
- DiagUp
- Left
- Right
- Top

3.18.7.1.1 Bottom

```
property Bottom: TxlsBorder;
```

Description
The *Bottom* property defines the bottom border of the cell.

See also:
TxlsBorder object
DiagDown property
DiagUp property
Left property
Right property
Top property

3.18.7.1.2 DiagDown

```
property DiagDown: TxlsBorder;
```

Description
The *DiagDown* property defines the diagonal down border of the cell.

See also:
TxlsBorder object
Bottom property
DiagUp property
Left property
Right property
Top property

3.18.7.1.3 DiagUp

```
property DiagUp: TxlsBorder;
```

Description
The *DiagUp* property defines the diagonal up border of the cell.

See also:
TxlsBorder object
Bottom property
DiagDown property
Left property
Right property
Top property

3.18.7.1.4 Left

```
property Left: TxlsBorder;
```

Description
The *Left* property defines the left border of the cell.

See also:
TxlsBorder object
Bottom property
DiagDown property
DiagUp property
Right property
Top property

3.18.7.1.5 Right

```
property Right: TxlsBorder;
```

Description
The *Right* property defines the right border of the cell.

See also:
TxlsBorder object
Bottom property
DiagDown property
DiagUp property
Left property
Top property

3.18.7.1.6 Top

```
property Top: TxlsBorder;
```

Description
The *Top* property defines the top border of the cell.

See also:
TxlsBorder object
Bottom property
DiagDown property
DiagUp property
Left property
Right property

3.18.8 TxlsFill object

Unit
QExport4XLS

Description
The *TxlsFill* object contains properties which determine the cell fill in the result Excel table.

3.18.8.1 Properties

Run-time only Key properties

- Background
- Foreground
- Pattern

3.18.8.1.1 Background

```
property Background: TxlsColor;
```

Description

The *Background* property defines the background color. The default value is *clrWhite*.

See also:

Charset property
Color property
Name property
Size property
Style property
Underline property

3.18.8.1.2 Foreground

```
property Foreground: TxlsColor;
```

Description
The *Foreground* property defines the foreground color. The default value is *clrBlack*.

See also:
TxlsColor type
Background property
Pattern property

3.18.8.1.3 Pattern

```
property Pattern: TxlsPattern;
```

Description
The *Pattern* property defines the filling pattern. The default value is *ptNone*. See TxlsPattern type for details.

See also:
TxlsPattern type
Background property
Foreground property

3.18.9 TxlsAlignment object

Unit
QExport4XLS

Description
The *TxlsAlignment* object allows you to define horizontal and vertical alignment of the result Excel cells.

3.18.9.1 Properties

Run-time only Key properties

- Horizontal
- Vertical

3.18.9.1.1 Horizontal

```
type
  TxlsHorizontalAlignment = (halGeneral, halLeft, halCenter, halRight, halFill);

property Horizontal: TxlsHorizontalAlignment;
```

Description

The *Horizontal* property defines the horizontal alignment of the cell. The following values are available: *halGeneral*, *halLeft*, *halCenter*, *halRight*, and *halFill*. The default value is *halGeneral*.

See also:

Vertical property

3.18.9.1.2 Vertical

```
type
  TxlsVerticalAlignment = (valTop, valCenter, valBottom, valJustify);

property Vertical: TxlsVerticalAlignment;
```

Description
The *Vertical* property defines the vertical alignment of the cell. The following values are available: *valTop*, *valCenter*, *valBottom*, and *valJustify*. The default value is *valBottom*.

See also:
Horizontal property

3.18.10 TxlsHyperLink object

Unit
QExport4XLS

Description
The *TxlsHyperLink* class contains properties which define the hyperlink parameters in the result Excel document.

See also:
TxlsHyperLinks object

3.18.10.1 Properties

Run-time only Key properties

- Col
- Format
- Row
- ScreenTip
- Style
- Target
- Title

3.18.10.1.1 Col

```
property Col: word;
```

Description
The *Col* property defines the horizontal position of the link.

See also:
Format property
Row property
ScreenTip property
Style property
Target property
Title property

3.18.10.1.2 Format

```
property Format: TxlsFormat;
```

Description
The *Format* property defines parameters of displaying the hyperlink in the result document.

See also:
TxlsFormat object
Col property
Row property
ScreenTip property
Style property
Target property
Title property

3.18.10.1.3 Row

```
property Row: word;
```

Description
The *Row* property defines the vertical position of the link.

See also:
Col property
Format property
ScreenTip property
Style property
Target property
Title property

3.18.10.1.4 ScreenTip

```
property ScreenTip: WideString;
```

Description
The *ScreenTip* property defines the text of the hint to display in Excel for the link.

See also:
Col property
Format property
Row property
Style property
Target property
Title property

3.18.10.1.5 Style

```
type
  TxlsHyperLinkStyle = (hlsURL, hlsLocalFile);

property Style: TxlsHyperLinkStyle;
```

Description
The *Style* property defines the type of the hyperlink target. The following values are available:
hlsURL - the global URL
hlsLocalFile - link to local file

See also:
Col property
Format property
Row property
ScreenTip property
Target property
Title property

3.18.10.1.6 Target

```
property Target: WideString;
```

Description
Use this property to define the hyperlink target.

See also:
Col property
Format property
Row property
ScreenTip property
Style property
Title property

3.18.10.1.7 Title

```
property Title: WideString;
```

Description
The *Title* property defines the hyperlink text.

See also:
Col property
Format property
Row property
ScreenTip property
Style property
Target property

3.18.11 TxlsHyperLinks object

Unit
QExport4XLS

Description
The *TxlsHyperLinks* is a collection of TxlsHyperLink objects. Use property Items to access these objects by their indexes.

See also:
TxlsHyperLink object

3.18.11.1 Properties

- Run-time only
- Key properties
- Items

3.18.11.1.1 Items

```
property Items[Index: integer] : TxlsHyperLink;
```

Description

Use *Items* to access the TxlsHyperLink objects by *Index*.

3.18.12 TxlsNote object

Unit
QExport4XLS

Description
The *TxlsNote* class contains properties which define the note parameters in the result Excel document.

See also:
TxlsNotes object

3.18.12.1 Properties

Run-time only Key properties

- Col
- Format
- Lines
- Row

3.18.12.1.1 Col

```
property Col: word;
```

Description
The *Col* property defines the horizontal position of the note.

See also:
Format property
Lines property
Row property

3.18.12.1.2 Format

```
property Format: TxlsNoteFormat;
```

Description
The *Format* property defines parameters of displaying the note in the result document.

See also:
TxlsNoteFormat object
Col property
Lines property
Row property

3.18.12.1.3 Lines

```
property Lines: TStrings;
```

Description
The *Lines* property contains the note text.

See also:
Col property
Format property
Row property

3.18.12.1.4 Row

```
property Row: word;
```

Description
The *Row* property defines the vertical position of the note.

See also:
Col property
Format property
Lines property

3.18.13 TxlsNotes object

Unit
QExport4XLS

Description
The *TxlsNotes* is a collection of TxlsNote objects. Use property Items to access these objects by their indexes.

See also:
TxlsNote object

3.18.13.1 Properties

- Run-time only
- Key properties
- Items

3.18.13.1.1 Items

```
property Items[Index: integer] : TxlsNote;
```

Description

Use *Items* to access the TxlsNote objects by *Index*.

3.18.14 TxlsChartSeries object

Unit
QExport4XLS

Description
The *TxlsChartSeries* class contains properties which define the chart series in the chart which is defined in the TxlsChart object.

See also:
TxlsChart object
TxlsChartSeriesList object

3.18.14.1 Properties

Run-time only Key properties

- Color
- DataColumn
- DataRange
- DataRangeType
- Title

3.18.14.1.1 Color

```
property Color: TxlsColor;
```

Description
The *Color* property defines the chart series color.

See also:
DataRange property
Title property

3.18.14.1.2 DataColumn

```
property DataColumn: string;
```

Description
The *DataColumn* property defines the data column name for the result chart series. This property is used only if DataRangeType is *rtColumn*.

See also:
DataRangeType property

3.18.14.1.3 DataRange

```
property DataRange: TxlsDataRange;
```

Description
The *DataRange* property allows you to define custom data range for the series. This property is used only if DataRangeType is *rtCustom*.

See also:
TxlsDataRange object
Color property
Title property

3.18.14.1.4 DataRangeType

```
type
  TxlsRangeType = (rtColumn, rtCustom);

property DataRangeType: TxlsRangeType;
```

Description

The *DataRangeType* property defines the data range type for the series. If *DataRangeType* value is *rtColumn*, data range is defined by the data column. The column name for the range is defined by the DataColumn property. If *DataRangeType* value is *rtCustom*, the data range is defined by the DataRange property.

See also:

DataColumn property
DataRange property

3.18.14.1.5 Title

```
property Title: WideString;
```

Description
The *Title* property defines the title of the result chart series.

See also:
Color property
DataRange property

3.18.15 TxlsChartSeriesList object

Unit
QExport4XLS

Description
The *TxlsChartSeriesList* is a collection of TxlsChartSeries objects. Use property Items to access these objects by their indexes.

See also:
TxlsChartSeries object

3.18.15.1 Properties

- Run-time only
- Key properties
- Items

3.18.15.1.1 Items

```
property Items[Index: integer]: TxlsChartSeries;
```

Description
Use *Items* to access the TxlsChartSeries objects by *Index*.

3.18.16 TxlsChart object

Unit
QExport4XLS

Description
The *TxlsChart* class contains properties which define the chart parameters in the result Excel document.

See also:
TxlsCharts object

3.18.16.1 Properties

Run-time only Key properties

- AutoColor
- CategoryLabels
- CategoryLabelsColumn
- CategoryLabelsType
- LegendPlacement
- Position
- Series
- ShowLegend
- Style
- Title

3.18.16.1.1 AutoColor

```
property AutoColor: boolean;
```

Description
The *AutoColor* property enables or disables the automatic defining colors of the chart series in the result Excel document.

See also:
CategoryLabels property
LegendPlacement property
Position property
Series property
ShowLegend property
Style property
Title property

3.18.16.1.2 CategoryLabels

```
property CategoryLabels: TxlsDataRange;
```

Description
The *CategoryLabels* property allows you to define the data range for the horizontal axis lables of the chart. This property is used only if CategoryLabelsType property is *rtCustom* .

See also:
CategoryLabelsColumn property
CategoryLabelsType property

3.18.16.1.3 CategoryLabelsType

```
type
  TxlsRangeType = (rtColumn, rtCustom);

property CategoryLabelsType: TxlsRangeType;
```

Description

Use the *CategoryLabelsType* property to define the type of data range for marking the horizontal axis of the chart. If the *CategoryLabelsType* value is *rtColumn*, data range is defined by the data column. The column name for the range is defined by the CategoryLabelsColumn property. If the *CategoryLabelsType* value is *rtCustom*, the data range is defined by the CategoryLabels property.

See also:

CategoryLabels property
CategoryLabelsColumn property

3.18.16.1.4 CategoryLabelsColumn

```
property CategoryLabelsColumn: string;
```

Description

The *CategoryLabelsColumn* property defines the data column name for the horizontal axis lables of the chart. This property is used only if the CategoryLabelsType property is *rtColumn*.

See also:

CategoryLabels property
CategoryLabelsColumn property

3.18.16.1.5 LegendPlacement

```
type
  TxlsChartLegendPlacement = (clpBottom, clpCorner, clpTop, clpRight, clpLeft);

property LegendPlacement: TxlsChartLegendPlacement;
```

Description
The *LegendPlacement* property defines the position of the chart legend.

See also:
AutoColor property
CategoryLabels property
Position property
Series property
ShowLegend property
Style property
Title property

3.18.16.1.6 Position

```
property Position: TxlsChartPosition;
```

Description
Use this property to define the chart position in the result Excel document.

See also:
TxlsChartPosition object
AutoColor property
CategoryLabels property
LegendPlacement property
Series property
ShowLegend property
Style property
Title property

3.18.16.1.7 Series

```
property Series: TxlsChartSeriesList;
```

Description
This property contains the collection of the chart series which belongs to this chart.

See also:
TxlsChartSeriesList object
AutoColor property
CategoryLabels property
LegendPlacement property
Position property
ShowLegend property
Style property
Title property

3.18.16.1.8 Show Legend

```
property ShowLegend: boolean;
```

Description
Use the *ShowLegend* option to enable or disable displaying of the chart legend.

See also:
AutoColor property
CategoryLabels property
LegendPlacement property
Position property
Series property
Style property
Title property

3.18.16.1.9 Style

```
type
  TxlsChartStyle = (xcsColumn, xcsColumn3d, xcsBar, xcsBar3d, xcsLine, xcsLineMark, xc
xcsPie, xcsPie3d, xcsArea, xcsArea3d, xcsSurface, xcsSurface3d, xcsRadar, xcsRadarArea

property Style: TxlsChartStyle;
```

Description
Use the *Style* property to define the chart style.

See also:
AutoColor property
CategoryLabels property
LegendPlacement property
Position property
Series property
ShowLegend property
Title property

3.18.16.1.10 Title

```
property Title: WideString;
```

Description
The *Title* property defines the chart title in the result Excel document.

See also:
AutoColor property
CategoryLabels property
LegendPlacement property
Position property
Series property
ShowLegend property
Style property

3.18.17 TxlsCharts object

Unit
QExport4XLS

Description
TxlsCharts is a collection of TxlsChart objects. Use the Items property to access these objects by their indexes.

See also:
TxlsChart object

3.18.17.1 Properties

- Run-time only
- Key properties
- Items

3.18.17.1.1 Items

```
property Items[Index: integer]: TxlsChart;
```

Description

Use *Items* to access the TxlsChart objects by *Index*.

3.18.18 TxlsPicture object

Unit
QExport4XLS

Description
The *TxlsPicture* class defines the picture in the result Excel document. Picture is not a displayable object in itself. It is used to create TxlsImage objects and add them to the document.

See also:
TxlsCharts object

3.18.18.1 Properties

Run-time only Key properties

- Name
- FileName

3.18.18.1.1 Name

```
property Name: string;
```

Description
This property defines the name of the picture in the result Excel file.

3.18.18.1.2 FileName

```
property FileName: string;
```

Description
This property defines the name of the file that contains the picture.

3.18.19 TxlsPictures object

Unit
QExport4XLS

Description
The *TxlsPictures* is a collection of TxlsPicture objects. Use the Items property to access these objects by their indexes.

See also:
TxlsPicture object

3.18.19.1 Properties

- Run-time only
- Key properties
- Items

3.18.19.1.1 Items

```
property Items[Index: integer]: TxlsPicture;
```

Description

Use *Items* to access the TxlsPicture objects by *Index*.

3.18.20 TxlsImage object

Unit
QExport4XLS

Description
The *TxlsImage* class contains properties which define the visible image in the result Excel document.

See also:
TxlsImages object

3.18.20.1 Properties

Run-time only Key properties

- Col
- PictureName
- Row
- Title
- Zoom

3.18.20.1.1 Col

```
property Col: word;
```

Description
This property defines the horizontal position of the image in the result Excel file.

See also:
PictureName property
Row property
Title property
Zoom property

3.18.20.1.2 PictureName

```
property PictureName: string;
```

Description
This property defines the name of the picture that the image uses. The picture is defined by the respective TxlsPicture object.

See also:
Col property
Row property
Title property
Zoom property

3.18.20.1.3 Row

```
property Row: word;
```

Description
This property defines the vertical position of the image in the result Excel file.

See also:
Col property
PictureName property
Title property
Zoom property

3.18.20.1.4 Title

```
property Title: WideString;
```

Description
This property defines the title of the image that would be displayed in the result Excel file.

See also:
Col property
PictureName property
Row property
Zoom property

3.18.20.1.5 Zoom

```
type
    TxlsZoom = 0..1000;

property Zoom: TxlsZoom;
```

Description
This property defines zooming ratio for the image in the result Excel file in percentage wise.

See also:
Col property
PictureName property
Row property
Title property

3.18.21 TxlsImages object

Unit
QExport4XLS

Description
The *TxlsImages* is a collection of TxlsImage objects. Use the Items property to access these objects by their indexes.

See also:
TxlsImage object

3.18.21.1 Properties

- Run-time only
- Key properties
- Items

3.18.21.1.1 Items

```
property Items[Index: integer]: TxlsImage;
```

Description

Use *Items* to access the TxlsImage objects by *Index*.

3.18.22 TxlsCell object

Unit
QExport4XLS

Description
The *TxlsCell* class contains properties which allow you to place a value to the definite place in the result Excel document.

See also:
TxlsCells object

3.18.22.1 Properties

Run-time only Key properties

- CellType
- Col
- DateTimeFormat
- Format
- IsBoolean
- IsDateTime
- IsNumeric
- IsString
- NumericFormat
- Row
- Value

3.18.22.1.1 CellType

```
type
    TxlsCellType = (ctBoolean, ctDateTime, ctNumeric, ctString);

property CellType: TxlsCellType;
```

Description
This property defines the type of the cell value.

See also:
Col property
DateTimeFormat property
Format property
IsBoolean property
IsDateTime property
IsNumeric property
IsString property
NumericFormat property
Row property
Value property

3.18.22.1.2 Col

```
property Col: word;
```

Description
This property defines the horizontal position of the cell in the result Excel file.

See also:
CellType property
DateTimeFormat property
Format property
IsBoolean property
IsDateTime property
IsNumeric property
IsString property
NumericFormat property
Row property
Value property

3.18.22.1.3 DateTimeFormat

```
property DateTimeFormat: string;
```

Description
This property defines the formatting string for the date/time values of the cell.

See also:
CellType property
Col property
Format property
IsBoolean property
IsDateTime property
IsNumeric property
IsString property
NumericFormat property
Row property
Value property

3.18.22.1.4 Format

```
property Format: TxlsFormat;
```

Description
This property defines the formatting options for the cell in the result Excel document.

See also:
CellType property
Col property
DateTimeFormat property
IsBoolean property
IsDateTime property
IsNumeric property
IsString property
NumericFormat property
Row property
Value property

3.18.22.1.5 IsBoolean

```
property IsBoolean: boolean;
```

Description
This property returns *True* if the CellType property is set to *ctBoolean*.

See also:
CellType property
Col property
DateTimeFormat property
Format property
IsDateTime property
IsNumeric property
IsString property
NumericFormat property
Row property
Value property

3.18.22.1.6 IsDateTime

```
property IsDateTime: boolean;
```

Description
This property returns *True* if the CellType property is set to *ctDateTime*.

See also:
CellType property
Col property
DateTimeFormat property
Format property
IsBoolean property
IsNumeric property
IsString property
NumericFormat property
Row property
Value property

3.18.22.1.7 IsNumeric

```
property IsNumeric: boolean;
```

Description
This property returns *True* if the CellType property is set to *ctNumeric*.

See also:
CellType property
Col property
DateTimeFormat property
Format property
IsBoolean property
IsDateTime property
IsString property
NumericFormat property
Row property
Value property

3.18.22.1.8 IsString

```
property IsString: boolean;
```

Description
This property returns *True* if the CellType property is set to *ctString*.

See also:
CellType property
Col property
DateTimeFormat property
Format property
IsBoolean property
IsDateTime property
IsNumeric property
NumericFormat property
Row property
Value property

3.18.22.1.9 NumericFormat

```
property NumericFormat: string;
```

Description
This property defines the formatting string for the numeric values of the cell.

See also:
CellType property
Col property
DateTimeFormat property
Format property
IsBoolean property
IsDateTime property
IsNumeric property
IsString property
Row property
Value property

3.18.22.1.10 Row

```
property Row: word;
```

Description
This property defines the vertical position of the cell in the result Excel file.

See also:
CellType property
Col property
DateTimeFormat property
Format property
IsBoolean property
IsDateTime property
IsNumeric property
IsString property
NumericFormat property
Value property

3.18.22.1.11 Value

```
property Value: Variant;
```

Description
This property defines the value of the cell.

See also:
CellType property
Col property
DateTimeFormat property
Format property
IsBoolean property
IsDateTime property
IsNumeric property
IsString property
NumericFormat property
Row property

3.18.23 TxlsCells object

Unit
QExport4XLS

Description
The *TxlsCells* is a collection of TxlsCell objects. Use the Items property to access these objects by their indexes.

See also:
TxlsCell object

3.18.23.1 Properties

- Run-time only
- Key properties
- Items

3.18.23.1.1 Items

```
property Items[Index: integer]: TxlsCell;
```

Description

Use *Items* to access the TxlsCell objects by *Index*.

3.18.24 TxlsMergedCells object

Unit
QExport4XLS

Description
The *TxlsMergedCells* class contains properties which allow you to merge cells in the result Excel document.

See also:
TxlsMergedCellList object

3.18.24.1 Properties

Run-time only Key properties

- FirstCol
- FirstRow
- LastCol
- LastRow

3.18.24.1.1 FirstCol

```
property FirstCol: word;
```

Description
This property defines the first column of the cell range to merge.

See also:
FirstRow property
LastCol property
LastRow property

3.18.24.1.2 FirstRow

```
property FirstCol: word;
```

Description
This property defines the first row of the cell range to merge.

See also:
FirstCol property
LastCol property
LastRow property

3.18.24.1.3 LastCol

```
property LastCol: word;
```

Description
This property defines the last column of the cell range to merge.

See also:
FirstCol property
FirstRow property
LastRow property

3.18.24.1.4 LastRow

```
property LastRow: word;
```

Description
This property defines the last row of the cell range to merge.

See also:
FirstCol property
FirstRow property
LastCol property

3.18.25 TxlsMergedCellList component

Unit
QExport4XLS

Description
The *TxlsMergedCellList* is a collection of TxlsMergedCells objects. Use the Items property to access these objects by their indexes.

See also:
TxlsMergedCells object

3.18.25.1 Properties

- Run-time only
- Key properties
- Items

3.18.25.1.1 Items

```
property Items[Index: integer]: TxlsMergedCells;
```

Description
Use *Items* to access the TxlsMergedCells objects by *Index*.

3.18.26 TxlsNoteFormat object

Unit
QExport4XLS

Description
The *TxlsNoteFormat* object contains properties which describe style characteristics used for displaying Excel notes. *TxlsNoteFormat* defines font (name, style, size etc.), fill type, transparency and alignment (horizontal and vertical).

3.18.26.1 Properties

Run-time only Key properties

- Alignment
- BackgroundColor
- FillType
- Font
- ForegroundColor
- Gradient
- Orientation
- Transparency

3.18.26.1.1 Alignment

```
property Alignment: TxlsAlignment;
```

Description
The *Alignment* property defines the text alignment in the notes.

See also:
TxlsAlignment object
BackgroundColor property
FillType property
Font property
ForegroundColor property
Gradient property
Orientation property
Transparency property

3.18.26.1.2 BackgroundColor

```
property BackgroundColor: TColor;
```

Description
The *Background* property defines the target color for the gradient fill.

See also:
Alignment property
FillType property
Font property
ForegroundColor property
Gradient property
Orientation property
Transparency property

3.18.26.1.3 FillType

```
type
  TxlsNoteFillType = (nftSolid, nftGradient);

property FillType: TxlsNoteFillType;
```

Description
The *FillType* property defines the type of filling the note background. The following values are available:
nftSolid - solid fill
nftGradient - gradient fill

The default is *nftSolid*.

See also:
Alignment property
BackgroundColor property
Font property
ForegroundColor property
Gradient property
Orientation property
Transparency property

3.18.26.1.4 Font

```
property Font: TxlsFont;
```

Description
The *Font* property defines the properties of the note text font to use them in the result document.

See also:
TxlsFont object
Alignment property
BackgroundColor property
FillType property
ForegroundColor property
Gradient property
Orientation property
Transparency property

3.18.26.1.5 ForegroundColor

```
property ForegroundColor: TColor;
```

Description
The *ForegroundColor* defines the background color for the notes when the *FillType* property is set to solid fill type, and the source color when the gradient fill is selected.

See also:
Alignment property
BackgroundColor property
FillType property
Font property
Gradient property
Orientation property
Transparency property

3.18.26.1.6 Gradient

```
type
  TxlsNoteGradient = (ngrHorizontal, ngrVertical, ngrDiagonalUp, ngrDiagonalDown, ngrF

property Gradient: TxlsNoteGradient;
```

Description

The *Gradient* property defines the type of the gradient fill. The following gradient types are available:
ngrHorizontal - horizontal gradient fill
ngrVertical - vertical gradient fill
ngrDiagonalUp - diagonal gradient from the top left corner to the bottom right corner
ngrDiagonalDown - diagonal gradient from the top right corner to the bottom left corner
ngrFromCorner - gradient fill from corner
ngrFromCenter - gradient fill from center

See also:
Alignment property
BackgroundColor property
FillType property
Font property
ForegroundColor property
Orientation property
Transparency property

3.18.26.1.7 Orientation

```
type
  TxlsOrientation = (xrtNoRotation, xlrTopToBottom, xlrCounterClockWise, xlrClockWise)

property Orientation: TxlsOrientation;
```

Description
Use the *Orientation* property to define the note orientation. The following values are available: *xrtNoRotation*, *xlrTopToBottom*, *xlrCounterClockWise*, and *xlrClockWise*. The *xlrClockWise* and *xlrCounterClockWise* values rotate the note to 90 degrees, *xlrTopToBottom* orients the note text vertically.

See also:
Alignment property
BackgroundColor property
FillType property
Font property
ForegroundColor property
Gradient property
Transparency property

3.18.26.1.8 Transparency

```
type
  TxlsPercent = 0..100;

property Transparency: TxlsPercent;
```

Description
The *Transparency* property defines the percentage of the note transparency.

See also:
Alignment property
BackgroundColor property
FillType property
Font property
ForegroundColor property
Gradient property
Orientation property

3.18.27 TxlsDataRange object

Unit
QExport4XLS

Description
The *TxlsDataRange* class describes the data range. This class is used to define the chart data range in the TxlsChartSeries class.

See also:
TxlsChartSeries object

3.18.27.1 Properties

Run-time only Key properties

- Col1
- Col2
- Row1
- Row2

3.18.27.1.1 Col1

```
property Col1: byte;
```

Description
The *Col1* property defines the left side of the data range.

See also:
Col2 property
Row1 property
Row2 property

3.18.27.1.2 Col2

```
property Col2: byte;
```

Description
The *Col2* property defines the right side of the data range.

See also:
Col1 property
Row1 property
Row2 property

3.18.27.1.3 Row1

```
property Row1: word;
```

Description
The *Row1* property defines the top of the data range.

See also:
Col1 property
Col2 property
Row2 property

3.18.27.1.4 Row2

```
property Row2: word;
```

Description
The *Row2* property defines the bottom of the data range.

See also:
Col1 property
Col2 property
Row1 property

3.18.28 TxlsChartPosition object

Unit
QExport4XLS

Description
The *TxlsChartPosition* class describes the chart position. This class is used to define the chart position in the TxlsChart class.

See also:
TxlsChart object

3.18.28.1 Properties

Run-time only Key properties

- X1
- X2
- Y1
- Y2

3.18.28.1.1 X1

```
property X1: byte;
```

Description
The *X1* property defines the horizontal position of the top left corner of the chart.

See also:
X2 property
Y1 property
Y2 property

3.18.28.1.2 X2

```
property X2: byte;
```

Description
The *X2* property defines the horisontal position of the bottom right corner of the chart.

See also:
X1 property
Y1 property
Y2 property

3.18.28.1.3 Y1

```
property Y1: word;
```

Description
The *Y1* property defines the vertical position of the top left corner of the chart.

See also:
X1 property
X2 property
Y2 property

3.18.28.1.4 Y2

```
property Y2: word;
```

Description
The *Y2* property defines the vertical position of the bottom right corner of the chart.

See also:
X1 property
X2 property
Y1 property

3.18.29 TXLSOptions object

Unit
QExport4XLS

Description
The *TXLSOptions* class contains properties that set the parameters of the result Excel document.

3.18.29.1 Properties

Run-time only Key properties

- AggregateFormat
- CaptionsFormat
- DataFormat
- FooterFormat
- HeaderFormat
- PageFooter
- PageHeader
- SheetTitle
- HyperLinkFormat
- NoteFormat

3.18.29.1.1 AggregateFormat

```
property AggregateFormat: TxlsFormat;
```

Description
The *AggregateFormat* property defines the style of the cells containing aggregate functions. To add an aggregate function, use property Aggregate of the TxlsFieldFormat. You can also write code in the OnGetAggregateParams event handler to tune the aggregate in runtime.

See also:
TxlsFormat component
CaptionsFormat property
DataFormat property
FooterFormat property
HeaderFormat property
HyperLinkFormat property
NoteFormat property

3.18.29.1.2 CaptionsFormat

```
property CaptionsFormat: TxlsFormat;
```

Description
The *CaptionsFormat* property defines the style of the cells containing captions. You can also write code in the OnGetCaptionParams event handler to tune the Captions in runtime.

See also:
TxlsFormat component
AggregateFormat property
DataFormat property
FooterFormat property
HeaderFormat property

3.18.29.1.3 DataFormat

```
property DataFormat: TxlsFormat;
```

Description

The *DataFormat* property defines the style of the cells containing the exported data. This style will be used in all data columns as default. If you want to change these settings then you can define property FieldFormats or write code in the OnGetDataParams event handler to tune the data cells in runtime.

See also:

TxlsFormat component
AggregateFormat property
CaptionsFormat property
FooterFormat property
HeaderFormat property

3.18.29.1.4 FooterFormat

```
property FooterFormat: TxlsFormat;
```

Description
The *FooterFormat* property defines the footer style in the result file. You can also write code in the OnGetFooterParams event handler to tune the Footer in runtime.

See also:
TxlsFormat component
AggregateFormat property
CaptionsFormat property
DataFormat property
HeaderFormat property

3.18.29.1.5 HeaderFormat

```
property HeaderFormat: TxlsFormat;
```

Description
The *HeaderFormat* property defines the header style in the result file. You can also write code in the OnGetHeaderParams event handler to tune the Header in runtime.

See also:
TxlsFormat component
AggregateFormat property
CaptionsFormat property
DataFormat property
FooterFormat property

3.18.29.1.6 PageFooter

```
property PageFooter: string;
```

Description
The *PageFooter* property sets the footer of the result file page. The default setting of this property is *Page &P of &N* which puts at the top of every page its number and the total quantity of pages in a book (for more details on the use of special symbols see the note on Microsoft Excel).

See also:
PageHeader property
SheetTitle property

3.18.29.1.7 PageHeader

```
property PageHeader: string;
```

Description
The *PageHeader* property sets the header of the output file page. The default setting of this property is an empty string.

See also:
PageFooter property
SheetTitle property

3.18.29.1.8 SheetTitle

```
property SheetTitle: string;
```

Description
The *SheetTitle* property sets the title of the result worksheet.

See also:
PageFooter property
PageHeader property

3.18.29.1.9 HyperLinkFormat

```
property HyperLinkFormat: TxlsFormat;
```

Description
The *HyperLinkFormat* property defines the hyperlink style in the result Excel document.

See also:
TxlsFormat object
AggregateFormat property
CaptionsFormat property
DataFormat property
FooterFormat property
HeaderFormat property
NoteFormat property

3.18.29.1.10 NoteFormat

```
property NoteFormat: TxlsNoteFormat;
```

Description
The *NoteFormat* property defines the note style in the result Excel document.

See also:
TxlsNoteFormat object
AggregateFormat property
CaptionsFormat property
DataFormat property
FooterFormat property
HeaderFormat property
HyperLinkFormat property

3.18.30 TxlsPageSetup object

Unit
QExport4XLS

Description
The *TxlsPageSetup* class contains properties that set the parameters of the page for printing Excel document.

3.18.30.1 Properties

3.18.30.1.1 PaperSize

```
property PaperSize: Integer;
```

Description

The *PaperSize* property defines the size of the printed page. In the Object Inspector you can set the code number of the page size, in the run-time - the constant value. Constant names can be found in QExport4XLSCommon unit. The default value is iA4 (9).

3.18.30.1.2 Scale

```
property Scale: Integer;
```

Description

The *Scale* property defines the scale of the printed page. The value is set in %. The default value is 0, which means that no scaling is applied.

3.18.30.1.3 PageStart

```
property PageStart: Integer;
```

Description

The *PageStart* property defines the number of a page which is printed first.

3.18.30.1.4 FitWidth

```
property FitWidth: Integer;
```

Description
The *FitWidth* property defines the number of pages that are fitted to the width.

3.18.30.1.5 FitHeight

```
property FitHeight: Integer;
```

Description

The *FitHeight* property defines the number of pages that are fitted to the height.

3.18.30.1.6 CopiesCount

```
property CopiesCount: Integer;
```

Description
The CopiesCount property defines the number of printed document copies.

Note: Perhaps, this property is ignored by Excel, although it is documented. Probably, it works in Open Office.

3.18.30.1.7 LeftToRight

```
property LeftToRight: Boolean;
```

Description
If the property value is set to *True*, then the printing is performed from left to right then down (default value).
If the property value is set to *False*, then the printing is performed down then from left to right.

3.18.30.1.8 PortraitOrient

```
property PortraitOrient: Boolean;
```

Description
If the property value is set to *True*, then the page is printed in the portrait orientation (default value).
If the property value is set to *False*, then the page is printed in the landscape orientation.

3.18.30.1.9 NoColor

```
property NoColor: Boolean;
```

Description
If the property value is set to *True*, then the page is printed grayscaled.
If the property value is set to *False*, then the page is printed in color (default value).

3.18.30.1.10 DraftQuality

```
property DraftQuality: Boolean;
```

Description
If the property value is set to *True*, then the page is printed in low quality.
If the property value is set to *False*, then the page is printed in the normal mode (default value).

3.18.30.1.11 PrintNotes

```
property PrintNotes: Boolean;
```

Description
If the property value is set to *True*, then the notes will be printed.
If the property value is set to *False*, then the notes are not printed (default value).

3.18.30.1.12 PrintNotesAtEnd

```
property PrintNotesAtEnd: Boolean;
```

Description
If the property value is set to *True*, then the notes will be printed at the end.
If the property value is set to *False*, then the notes are printed in place (default value).

3.18.31 TGetAggregateParamsEvent type

Unit
QExport4XLS

```
type
  TGetAggregateParamsEvent = procedure(Sender: TObject; Col: integer; Format: TxlsFor
var FormatText, Value: string)
of object;
```

Description
The *OnGetAggregateParams* event type. Parameter *Col* is the column number, *Format* contains the cell parameters. Use *FormatText* variable to set the format of the result numerical value. If parameter *Value* is empty, then you can set Value to any string value, e.g. 'Total:'.

See also:
OnGetAggregateParams event

3.18.32 TGetCaptionParamsEvent type

Unit
QExport4XLS

```
type
  TGetCaptionParamsEvent = procedure(Sender: TObject; Col: integer; Format: TxlsFormat
```

Description
The *OnGetCaptionParams* event type. Parameter *Col* is the column number, *Format* contains the cell parameters. Use variable *Caption* to edit the cell value.

See also:
OnGetCaptionParams event

3.18.33 TGetHeaderFooterParamsEvent type

Unit
QExport4XLS

```
type
  TGetHeaderFooterParamsEvent = procedure(Sender: TObject; Col, Row: integer; Format:
of object;
```

Description
The *TGetHeaderFooterParamsEvent* type is used in OnGetBeforeDataParams, OnGetFooterParams and OnGetHeaderParams events. Parameters *Col* and *Row* indicate the cell position, and *Format* contains the cell parameters. Use variable *S* to set the string value to the cell.

See also:
OnGetBeforeDataParams event
OnGetFooterParams event
OnGetHeaderParams event

3.18.34 TGetDataParamsEvent type

Unit
QExport4XLS

```
type
  TGetDataParamsEvent = procedure(Sender: TObject; Col, Row: integer; Format: TxlsFor
var FormatText: string) of object;
```

Description
The *OnGetDataParams* event type. Parameters *Col* and *Row* indicate the cell position, and Format contains the cell parameters. Use *FormatString* to set the format of the result numeric, currency or date/time value.

See also:
OnGetDataParams event

3.18.35 TxlsColor type

Unit
QExport4XLS

type

```
  TxlsColor = (clrBlack, clrBrown, clrOliveGreen, clrDarkGreen, clrDarkTeal, clrDarkB
clrDarkRed, clrOrange, clrDarkYellow, clrGreen, clrTeal, clrBlue, clrBlueGray, clrGray
clrLime, clrSeaGreen, clrAqua, clrLightBlue, clrViolet, clrGray40Percent, clrPink, cl
clrTurquoise, clrSkyBlue, clrPlum, clrGray25Percent, clrRose, clrTan, clrLightYellow,
clrPaleBlue, clrLavender, clrWhite);
```

Description

clrBlack		clrAqua	
clrBrown		clrLightBlue	
clrOliveGreen		clrViolet	
clrDarkGreen		clrGray40Percent	
clrDarkTeal		clrPink	
clrDarkBlue		clrGold	
clrIndigo		clrYellow	
clrGray80Percent		clrBrightGreen	
clrDarkRed		clrTurquoise	
clrOrange		clrSkyBlue	
clrDarkYellow		clrPlum	
clrGreen		clrGray25Percent	
clrTeal		clrRose	
clrBlue		clrTan	
clrBlueGray		clrLightYellow	
clrGray50Percent		clrLightGreen	
clrRed		clrLightTurquoise	
clrLightOrange		clrPaleBlue	
clrLime		clrLavender	
clrSeaGreen		clrWhite	

See also:
TxlsFont - Color property
TxlsBorder - Color property
TxlsFill - Background property
TxlsFill - Foreground property

3.18.36 TxlsBorderStyle type

Unit
QExport4XLS

type

```
  TxlsBorderStyle = (bstNone, bstThin, bstMedium, bstDashed, bstDotted, bstThick, bstD
bstDashDot, bstMediumDashDot, bstDashDotDot, bstMediumDashDotDot, bstSlantedDashDot);
```

Description

bstNone	None
bstThin	
bstMedium	
bstDashed	
bstDotted	
bstThick	
bstDouble	
bstHair	
bstMediumDashed	
bstDashDot	
bstMediumDashDot	
bstDashDotDot	
bstMediumDashDotDot	
bstSlantedDashDot	

3.18.37 TxlsPattern type

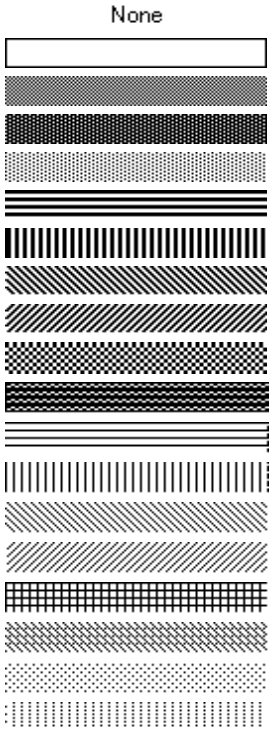
Unit
QExport4XLS

type

```
  TxlsPattern = (ptNone, ptSolid, ptChess, ptWhiteSpots, ptBlackSpots, ptBoldHorizonta
ptBoldDiagRight, ptBoldDiagLeft, ptBoldChess, ptRingMail, ptThinGorizontal, ptThinVert
ptThinDiagRight, ptCells, ptCrissCross, ptThinSpots, ptThinThinSpots);
```

Description

ptNone
ptSolid
ptChess
ptWhiteSpots
ptBlackSpots
ptBoldHorizontal
ptBoldVertical
ptBoldDiagRight
ptBoldDiagLeft
ptBoldChess
ptRingMail
ptThinGorizontal
ptThinVertical
ptThinDiagLeft
ptThinDiagRight
ptCells
ptCrissCross
ptThinSpots
ptThinThinSpots



3.19 QExport4Xlsx unit

Components
TQExport4Xlsx

Objects
TXlsxOptions
TXlsxStripStyle
TXlsxCellStyle

3.19.1 TXlsxOptions object

Unit
QExport4Xlsx

Description
The *TXLSxOptions* class contains properties that set the parameters of the result Excel 2007 document.

3.19.1.1 Properties

Run-time only Key properties

- HeaderStyle
- CaptionRowStyle
- DataStyle
- FooterStyle
- StripStyleType
- StripStylesList

3.19.1.1.1 HeaderStyle

```
property HeaderStyle: TXlsxCellStyle;
```

Description
Use this property to define the document's header style.

See also:
DataFont property
FooterFont property
HeaderFont property

3.19.1.1.2 CaptionRow Style

```
property CaptionRowStyle: TXlsxCellStyle;
```

Description

The *CaptionRowStyle* property contains parameters which define the captions appearance in the result document, such as colors and font.

3.19.1.1.3 DataStyle

```
property DataStyle: TXlsxCellStyle;
```

Description

The *DataStyle* property contains parameters which define the data appearance in the result document, such as colors and font.

3.19.1.1.4 FooterStyle

```
property FooterStyle: TXlsxCellStyle;
```

Description
Use this property to define the document's footer style.

3.19.1.1.5 StripStyleType

```
type
  TMSStripStyleType = (ssNone, ssColumn, ssRow);

property StripStyleType: TMSStripStyleType;
```

Description

This property defines strips direction - horizontal or vertical. If *StripType* has the *stCol* value, the strips are vertical, if the *stRow* value, the strips are horizontal. If the *stNone* value is set, the strips are disabled.

3.19.1.1.6 StripStylesList

```
property StripStylesList: TXlsxStripStyleList;
```

Description
Use this property to define the strips styles. The *StripStyles* property contains a collection of styles which would be used one by one.

3.19.2 TXlsxStripStyle object

Unit
QExport4Xlsx

StripStyle determines font size, color and other attributes, border and fill styles and more.

```
property Options: TXlsxCellStyle;
```

Description
The *Options* property is "complex" and contains some subproperties - properties of the *TXlsxCellStyle* class.

3.19.2.1 Properties

Run-time only Key properties

Options

3.19.2.1.1 Options

```
property Options: TXlsxCellStyle;
```

Description

The *Options* property is "complex" and contains some subproperties - properties of the *TXlsxCellStyle* class.

3.19.3 TXlsxCellStyle object

Unit
QExport4Xlsx

Description
The *TXlsxCellStyle* class contains parameters for defining the text appearance which can be applied to some of the result document parts.

3.19.3.1 Properties

Run-time only Key properties

- Font
- BackgroundColor
- UseBackground
- Alignment
- VerticalAligment
- WrapText
- UseBorder
- Border

3.19.3.1.1 Font

```
property Font: TFont;
```

Description
Use the *Font* property to define text font parameters for the current style.

3.19.3.1.2 BackgroundColor

```
property BackgroundColor: TColor;
```

Description

Use the *BackgroudColor* property to define the background color in the current style. This color is applied only if the *UseBackground* property is True.

3.19.3.1.3 UseBackground

```
property UseBackground: boolean;
```

Description

The *UseBackground* option enables using the background color in the current style. The default value is *True*.

3.19.3.1.4 Alignment

```
type
  TMSCellAlignment = (caLeft, caRight, caCenter, caJustify);

property Alignment: TMSCellAlignment;
```

Description

The *Alignment* property defines the text alignment for the current style. The following values are available:
caLeft - text is left-justified
caRight - text is right-justified
caCenter - text is centered
caJustify - text is distributed over the paragraph width

3.19.3.1.5 VerticalAligment

```
type
  TMSCellVerticalAligment = (cvaTop, cvaMiddle, cvaBottom);

property VerticalAligment: TMSCellVerticalAligment;
```

Description

The *Alignment* property defines the text alignment for the current style. The following values are available:
cvaTop - text is top-justified
cvaMiddle - text is middle-justified
cvaBottom - text is bottom-justified

3.19.3.1.6 WrapText

```
property WrapText: boolean;
```

Description

The *WrapText* property defines the text to appear on multiple lines in a cell.

3.19.3.1.7 UseBorder

```
property UseBorder: boolean;
```

Description

The *UseBorder* option enables using the border in the current style. The default value is *True*.

3.19.3.1.8 Border

```
type
  TXlsxBorderStyle = (xbsThin, xbsDashed, xbsDashDot, xbsDashDotDot, xbsDotted, xbsHa
xbsMedium, xbsMediumDashed,   xbsMediumDashDot, xbsSlantDashDot, xbsMediumDashDotDot),

property Border: TXlsxBorder;
```

Description

The *Border* property defines the appearance of the cell borders.

3.19.4 TOnGetDataParamsEvent type

Unit
QExport4Xlsx

```
type
  TGetDataParamsEvent = procedure(Sender: TObject; const Col, Row: integer;
var Style: TXlsxCellStyle; var UseStyle: Boolean) of object;
```

Description
The *OnGetDataParams* event type. Parameters *Col* and *Row* indicate the cell position. *UseStyle* indicates whether defined *Style* is applied to cell.

See also:
OnGetDataParams event

3.20 QExport4XML unit

Components
TQExport4XML

Objects
TXMLOptions

3.20.1 TXMLOptions object

Unit
QExport4XML

Description
The *TXMLOptions* class contains the properties that set the parameters of the result document.

3.20.1.1 Properties

Run-time only Key properties

- Encoding
- StandAlone
- Version

3.20.1.1.1 Encoding

```
property Encoding: string;
```

Description
This property allows you to set the encoding of the result document.

See also:
StandAlone property
Version property

3.20.1.1.2 StandAlone

```
property StandAlone: boolean;
```

Description
If this option is *true*, the result document will be standalone.

See also:
Encoding property
Version property

3.20.1.1.3 Version

```
property Version: string;
```

Description
This property displays the current XML version.

See also:
Encoding property
StandAlone property

Part

4 Appendix

4.1 Supported file formats

MS Excel 97-2003
The most popular e-table format used by Microsoft® Excel (*.xls).

MS Access
File of Microsoft® Access format (*.mdb, *.accdb) with an ADO connection used.

MS Word 97-2003
One of the most popular text processing formats used by Microsoft® Word (*.doc).

RTF
Rich Text Format (*.rtf) supported by many text processing programs (e.g. WordPad).

HTML
Hyper Text Markup Language file format (*.html, *.htm), complete compatibility with HTML 4.0 specification.

PDF
A standard format in electronic publishing (*.pdf).

Text file
Plain text file format (*.txt).

CSV file
Comma-Separated Value file format (*.csv).

DIF file
Data Interchange File (*.dif) format.

SYLK
Symbolic Links (*.slk) file format.

Note: all the text formats including *Text file*, *CSV*, *DIF*, *SYLK* are usually used as working or interchange formats.

LaTeX
A specific file format (*.tex) which is a popular (especially among mathematicians and physicists) macroextension of *TeX* pack developed by D.Knut.

XML
A markup language for documents containing structured information (*.xml).

DBF
Database file format (*.dbf) used by dBASE and a number of xBASE applications.

MS Excel
The contemporary e-table format used by Microsoft® Excel (*.xlsx).

MS Word

The contemporary text processing format used by Microsoft® Word (*.docx).

ODF Spreadsheets
OASIS Open Document Format for Office Applications - open document file format for spreadsheets (*.ods) used by a number of applications including OpenOffice.org and KOffice.

ODF Text
OASIS Open Document Format for Office Applications - open document file format for word processing (*.odt) documents used by a number of applications including OpenOffice.org and KOffice.

4.2 Color Using Notes

Please note the following:

- Although both Delphi/C++ Builder and HTML store the color in RGB format, the order of color channels differs: for example, the blue color is represented in Delphi as *$FF0000*, and in HTML as *#0000FF*. All the colors in the component properties should be set in Delphi/C++ Builder format, when saving to file they will be reformatted automatically
- When setting "dangerous" (not included in the common color schemes) color settings, be careful: it may happen that the user viewing your file won't have such a bright color scheme, which may negatively influence the document design.

See also:
BorderColor
HeadersRowBgColor
HeadersRowFontColor
OddRowBgColor
TableBgColor
TableFontColor
ALinkColor
BackgroundColor
LinkColor
VLinkColor

Credits

Software Developers:

Alexey Butalov

Alex Paclin

Alexey Saybel

Dmitry Ziborov

Vadim Vinokur

Alexey Gusev

Technical Writers:

Dmitry Doni

Semyon Slobodenyuk

Olga Ryabova

Cover Designer:

Tatyana Makurova

Translators:

Anna Shulkina

Sergey Fominykh

Team Coordinators:

Alexey Butalov

Alexander Chelyadin

Roman Tkachenko