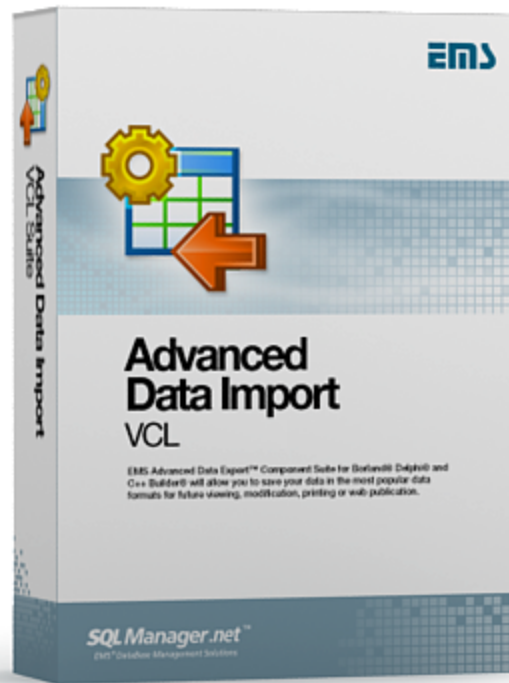




Advanced Data Import for RAD Studio VCL User's Manual

Advanced Data Import for RAD Studio VCL User's Manual

© 1999-2025 EMS Software Development

All rights reserved.

This manual documents EMS Advanced Data Import for RAD Studio VCL

No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Use of this documentation is subject to the following terms: you may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way.

Document generated on: 09.10.2025

Table of Contents

Part I Welcome to Advanced Data Import for RAD Studio VCL!	10
Overview	10
What's new	11
Installation	12
Registration	14
How to register Advanced Data Import	15
Other EMS Products	16
Part II Advanced Data Import Component	23
TQImport3	24
TQImport3 Reference	24
Properties	25
AddType	26
AutoTrimValue	27
Canceled	28
CommitAfterDone	29
CommitRecCount	30
DataSet	31
DBGGrid	32
ErrorLog	33
ErrorLogFileName	34
ErrorRecs	35
Errors	36
FieldFormats	37
FileName	38
Formats	39
GridCaptionRow	40
GridStartRow	41
ImportDestination	42
ImportedRecs	43
ImportMode	44
ImportRecCount	45
KeyColumns	46
LastAction	47
ListView	48
Map	49
RewriteErrorLogFile	50
ShowErrorLog	51
StringGrid	52
Methods	53
Cancel	54
Execute	55
ImportToCSV	56

LoadConfiguration.....	57
SaveConfiguration.....	58
Events	59
OnAfterImport.....	60
OnAfterPost.....	61
OnBeforeImport.....	62
OnBeforePost.....	63
OnDestinationLocate.....	64
OnImportCancel.....	65
OnImportError.....	66
OnImportRecord.....	67
OnImportRow Complete.....	68
OnNeedCommit.....	69
OnUserDefinedImport.....	70
TADO_QImport3Access	71
TADO_QImport3Access Reference	71
Properties	72
SkipFirstRow s.....	73
SourceType.....	74
SQL.....	75
TableName.....	76
TQImport3ASCII	77
TQImport3ASCII Reference	77
Properties	78
Comma.....	79
Encoding.....	80
SkipFirstRow s.....	81
Map.....	82
Quote.....	83
Events	84
TQImport3DataSet	85
TQImport3DataSet Reference	85
Properties	86
GoToFirstRecord.....	87
SkipFirstRow s.....	88
Source.....	89
TQImport3DBF	90
TQImport3DBF Reference	90
Properties	91
SkipDeleted.....	92
SkipFirstRow s.....	93
Map.....	94
Events	95
TQImport3Docx	96
TQImport3Docx Reference	96
Properties	97
Map.....	98
SkipFirstRow s.....	99
NeedFillMerge.....	100
TableNumber.....	101
Events	102

TQImport3HTML	103
TQImport3HTML Reference	103
Properties	104
Map.....	105
SkipFirstRow s.....	106
TableNumber.....	107
Events	108
TQImport3ODS	109
TQImport3ODS Reference	109
Properties	110
Map.....	111
SkipFirstRow s.....	112
SheetName.....	113
NotExpandMergedValue.....	114
Events	115
TQImport3ODT	116
TQImport3ODT Reference	116
Properties	117
Map.....	118
SkipFirstRow s.....	119
SheetName.....	120
UseComplexTables.....	121
Events	122
TQImport3Wizard	123
TQImport3Wizard Reference	123
Properties	124
AddType.....	125
AllowedImports.....	126
AutoChangeExtension.....	127
AutoLoadTemplate.....	128
AutoSaveTemplate.....	129
CloseAfterImport.....	130
CommitAfterDone.....	131
CommitRecCount.....	132
ConfirmOnCancel.....	133
CSVView erRow s.....	134
DataSet.....	135
DBGrid.....	136
ErrorLog.....	137
ErrorLogFileName.....	138
ExcelMaxColWidth.....	139
ExcelView erRow s.....	140
FieldFormats.....	141
FileName.....	142
Formats.....	143
GoToLastPageAfterLoadTemplate.....	144
GridCaptionRow	145
GridStartRow	146
ImportDestination.....	147
ImportMode.....	148
ImportRecCount.....	149
KeyColumns.....	150

ListView	151
Picture	152
RewriteErrorLogFile	153
ShowErrorLog	154
ShowHelpButton	155
ShowProgress	156
ShowSaveLoadButtons	157
StringGrid	158
TemplateFileName	159
TextViewerRows	160
Methods	161
Execute	162
Events	163
OnAfterImport	164
OnBeforeImport	165
OnBeforePost	166
OnDestinationLocate	167
OnImportCancel	168
OnImportError	169
OnImportRecord	170
OnLoadTemplate	171
OnNeedCommit	172
TQImport3XLS	173
TQImport3XLS Reference	173
Properties	174
ImportRange	175
Map	176
SkipFirstCols	179
SkipFirstRows	180
Events	181
TQImport3XLSx	182
TQImport3XLSx Reference	182
Properties	183
Map	184
SkipFirstRows	187
LoadHiddenSheets	188
NeedFillMerge	189
SheetName	190
Events	191
TQImport3XML	192
TQImport3XML Reference	192
Properties	193
SkipFirstRows	194
WriteOnFly	195
TQImport3XMLDoc	196
TQImport3XMLDoc Reference	196
Properties	197
XPath	198
SkipFirstRows	199
Map	200
DataLocation	201
Events	202

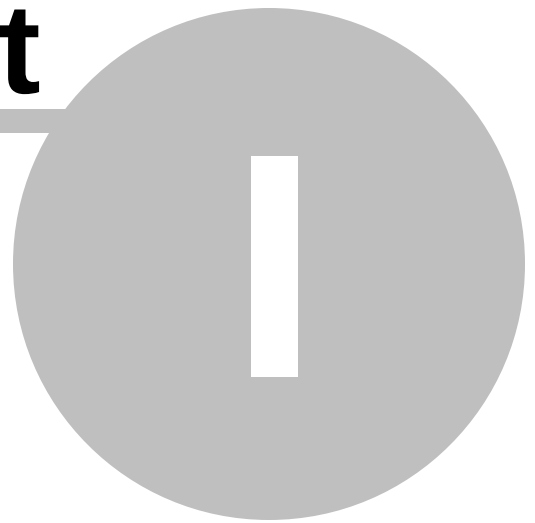
Part III Units

204

QImport3 unit	204
TQImportFormats object	205
Properties.....	206
BooleanFalse	207
BooleanTrue	208
DateSeparator	209
DecimalSeparator.....	210
LongDateFormat.....	211
LongTimeFormat.....	212
NullValues	213
ShortDateFormat.....	214
ShortTimeFormat.....	215
ThousandSeparator.....	216
TimeSeparator	217
TQImportRow object	218
Properties.....	219
Items	220
TQImportCol object	221
Properties.....	222
Name	223
Value	224
TQImportFieldFormat object	225
Properties.....	226
CharCase	227
CharSet	228
ConstantValue	229
DefaultValue	230
FieldName	231
GeneratorStep	232
GeneratorValue.....	233
NullValue	234
QuoteAction	235
LeftQuote	236
RightQuote	237
Replacements	238
TQImportFieldFormats object	239
Properties.....	240
Items	241
TQImportReplacement object	242
Properties.....	243
IgnoreCase	244
ReplaceWith	245
TextToFind	246
TQImportReplacements object	247
Properties.....	248
Items	249
QImport3Access unit	250
QImport3ASCII unit	251
QImport3DataSet unit	252

QImport3DBF unit	253
QImport3Docx unit	254
QImport3HTML unit	255
QImport3ODS unit	256
QImport3ODT unit	257
QImport3Wizard unit	258
QImport3XLS unit	259
QImport3XLSx unit	260
QImport3XML unit	261
QImport3XMLDoc unit	262
Part IV Advanced Data Import Wizard Guide	264
Selecting Data Source	266
Setting Correspondence	267
MS Excel	268
MS Access	270
DBF	272
TXT	274
CSV	275
Specifying Base Formats	277
Setting Import Options	280
Part V Appendix	283
Supported file formats	283

Part



1 Welcome to Advanced Data Import for RAD Studio VCL!

1.1 Overview

EMS Advanced Data Import for RAD Studio VCL is a component that allows you to import data from files of the most popular data formats to the database. You can import data from MS Excel, MS Access, DBF, XML, TXT, CSV, ODF, and HTML. There will be no need to waste your time on tiresome data conversion - Advanced Data Import for RAD Studio VCL will do the task quickly, irrespective of the source data format.

Visit our web-site for details: <https://www.sqlmanager.net/>

Key features

- Data import from the most popular data formats: MS Excel, MS Access, DBF, XML, TXT, CSV, OpenDocument format (ODS, ODT), and HTML.
- Import of Unicode data (UTF-8, UTF-16/UCS-2, UTF-32/UCS-4). Automatic detection and manual preset of text encoding for imported data.
- Powerful component and property editors which admit to setting many import parameters at the design-time easily.
- High productivity even on slow computers.
- Adjustable parameters for each type of import.
- 100% native Delphi code for MS Excel, DBF, TXT, CSV - no additional libraries or programs needed for the components to work; OLE, DDE, BDE, etc. are not required either.
- Detailed help system and a demo application for quicker mastering the product.
- RAD Studio XE2-XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11 Alexandria, 12 Athens, 13 Florence
- 64-bit Windows platform support

Product information

Homepage	https://sqlmanager.net/products/tools/advancedimport
Support Ticket System	https://www.sqlmanager.net/support
Register on-line	https://sqlmanager.net/products/tools/advancedimport/buy

1.2 What's new

Version**Advanced Data Import for RAD Studio VCL 3.16****Release date**

October 9, 2025

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

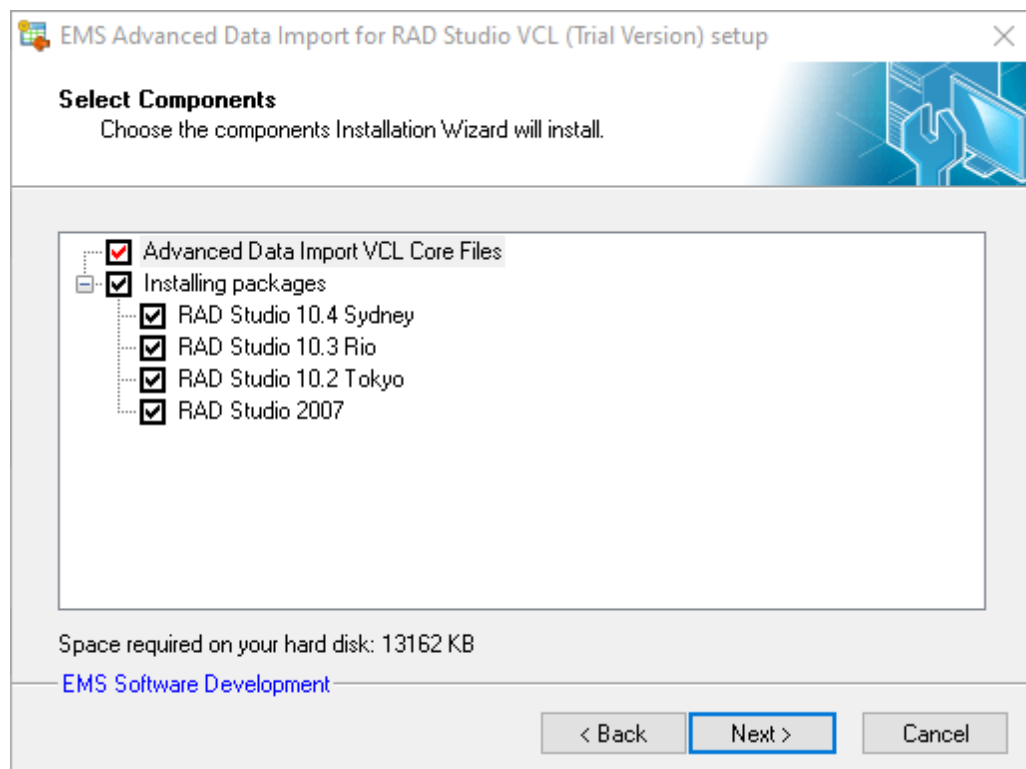
- Implemented support for RAD Studio 13 Florence
- Implemented support for Windows 11 ARM
- Fixed error of opening Access 97-2003 file in Windows 64-bit platform apps
- Fixed compiling of applications with C++Builder
- End of support for RAD Studio 2010 & XE

1.3 Installation

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

- download the distribution package of **Advanced Data Import 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**:



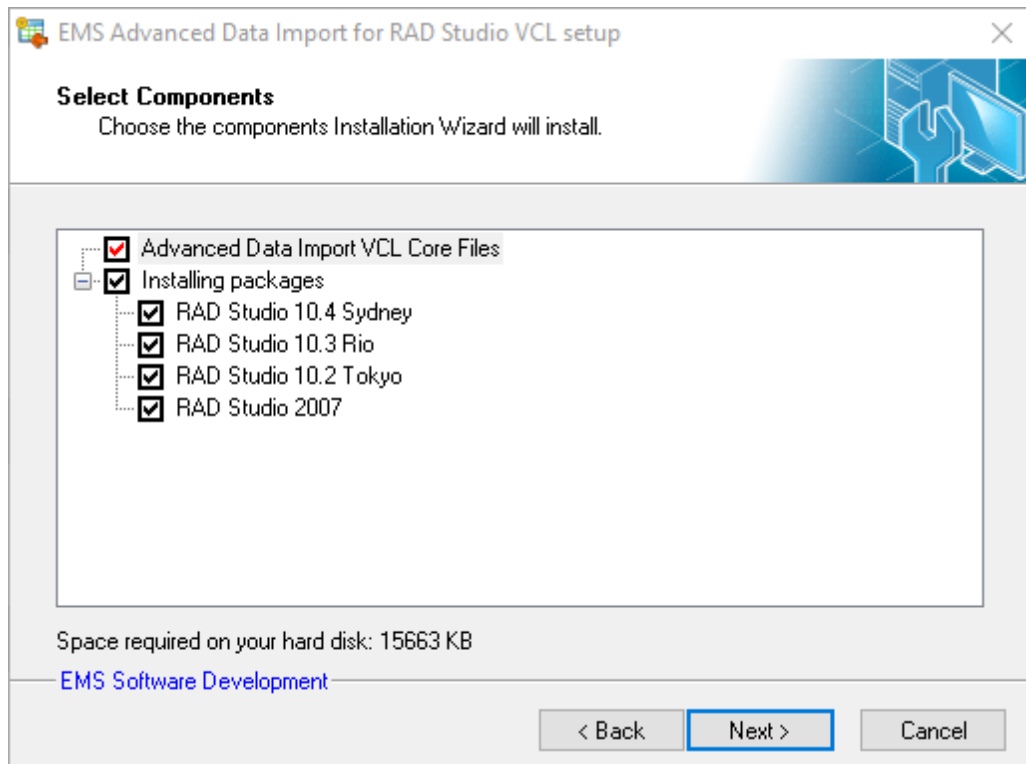
When you are done, you can finish installation of the **trial version** of **Advanced Data Import for RAD Studio VCL**.

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

- download the distribution package of **Advanced Data Import 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 Import 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 **PayPro Global** registration service. The **PayPro Global** order process is protected via a secure connection and makes on-line ordering by credit/debit card quick and safe.

PayPro Global 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.

If you want to review your order information, or you have questions about ordering or payments please visit our [PayPro Global Shopper Support](#), provided by **PayPro Global**.

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. Also you will receive a copy of registration keys 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	PayPro Global
Advanced Data Import for RAD Studio VCL Full version (with sources) + 1-Year Maintenance*	Register Now!
Advanced Data Import for RAD Studio VCL Full version (with sources) + 2-Year Maintenance*	
Advanced Data Import for RAD Studio VCL Full version (with sources) + 3-Year Maintenance*	
Advanced Data Import 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 Import

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

- receive the notification letter from **PayPro Global** 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 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.

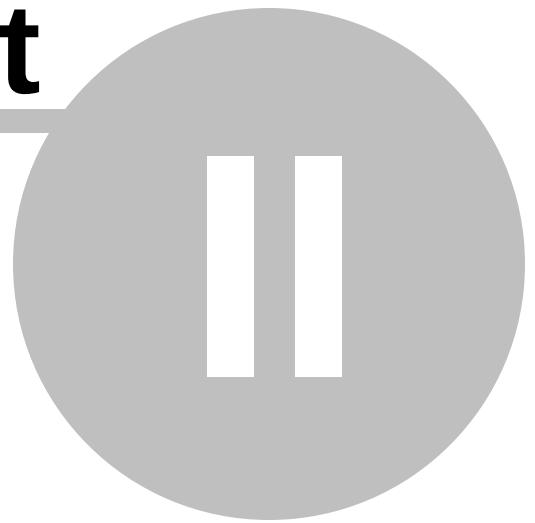


[Advanced Localizer for RAD Studio VCL](#)

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 Import Component

EMS Advanced Data Import for RAD Studio VCL represents a set of tools for importing data to any TDataSet component from files in different [formats](#), such as Microsoft Excel, Microsoft Access, DBF, XML, CSV, TXT, HTML, DOCX, ODT, ODS, XLSX, and XML Document.

From a programmer's point of view the component represents a homomorphic hierarchy of classes with the common ancestor [TQImport3](#). Beside the basic properties, methods and events, some specific characteristics are included in descendant classes.

The [TQImport3Wizard](#) component allows you to define all the import settings and import your data to the dataset from Excel, DBF, XML, CSV and TXT files within a dialog window. Using [TQImport3Wizard](#) you can add all the functionality of **Advanced Data Import for RAD Studio VCL** to your application within a single line of code!

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

Component	Brief description
TQImport3	Common data import component
TADO_QImport3Access	Provides data import from MS Access databases
TQImport3ASCII	Provides data import from CSV and TXT files
TQImport3DataSet	Provides data import from a TDataSet component.
TQImport3DBF	Provides data import from DBF files
TQImport3Docx	Provides data import from MS Word tables
TQImport3HTML	Intended for importing tables from HTML pages
TQImport3ODS	Provides data import from OpenDocument Spreadsheet files (Open Document format)
TQImport3ODT	Provides data import from the tables of OpenDocument Text files (Open Document format)
TQImport3Wizard	Provides a dialog window to define all the import settings
TADO_QImport3Wizard	Provides the same functionality as TQImport3Wizard component, including import from Access
TQImport3XLS	Provides data import from XLS files
TQImport3XLSx	Intended for importing the MS Excel sheets
TQImport3XML	Provides data import from XML files
TQImport3XMLDoc	Provides data import from generic XML files using XPath and DataLocation (tlAttributes, tlSubNodes)

2.1 TQImport3

2.1.1 TQImport3 Reference

Unit

[**QImport3**](#)

Description

The *TQImport3* class is a base class of the collection. Here the basic properties and events for all the descendant classes are determined, and so are the two basic methods - [Execute](#) - which is later overridden in each of the components, and [Cancel](#).

2.1.2 Properties

▶ Run-time only

 Key properties

-  [AddType](#)
- ▶  [Canceled](#)
-  [CommitAfterDone](#)
-  [CommitRecCount](#)
-  [DataSet](#)
-  [DBGrid](#)
-  [ErrorLog](#)
-  [ErrorLogFileName](#)
- ▶  [ErrorRecs](#)
- ▶  [Errors](#)
-  [FieldFormats](#)
- ▶  [FileName](#)
-  [Formats](#)
-  [GridCaptionRow](#)
-  [GridStartRow](#)
-  [ImportDestination](#)
- ▶  [ImportedRecs](#)
-  [ImportMode](#)
-  [ImportRecCount](#)
-  [KeyColumns](#)
-  [LastAction](#)
-  [ListView](#)
-  [Map](#)
-  [RewriteErrorLogFile](#)
-  [ShowErrorLog](#)
-  [StringGrid](#)

2.1.2.1 AddType

```
type TQImportAddType = (qatAppend, qatInsert);
```

```
property AddType: TQImportAddType;
```

Description

The *AddType* property defines what statement - *Append* or *Insert* - is used when writing data to the [ImportDestination](#). The default value is *qatInsert*.

2.1.2.2 AutoTrimValue

property AutoTrimValue: boolean;

Description

If property *AutoTrimValue* property is true Trim function is applied to all text data. The default value is false.

2.1.2.3 Canceled

property Canceled: boolean;

Description

The *Canceled* property is read-only and it indicates if the import process was canceled during its execution. When import is started this property is false, but if import is canceled before the completion, this property becomes true.

See also:

[OnImportCancel event](#)
[Cancel method](#)

2.1.2.4 CommitAfterDone

property CommitAfterDone: **boolean**;

Description

If the *CommitAfterDone* property is *true* then the [OnNeedCommit](#) event takes place after import is finished.

See also:

[CommitRecCount property](#)

[OnNeedCommit event](#)

2.1.2.5 CommitRecCount

property CommitRecCount: **integer**;

Description

The *CommitRecCount* property defines the number of records, after importing which the [OnNeedCommit](#) event takes place.

See also:

[CommitAfterDone property](#)

[OnNeedCommit event](#)

2.1.2.6 DataSet

property DataSet: TDataSet;

Description

The *DataSet* property points to the target TDataSet component to import data to. To use a DataSet instance as import destination, you should set the [ImportDestination](#) property to qidDataSet (default) and the dataset which must be opened (DataSet.Active = true) before the import begins (before the [Execute](#) method is invoked).

See also:

[ImportDestination property](#)

[DBGrid property](#)

[ListView property](#)

[StringGrid property](#)

[Execute method](#)

2.1.2.7 DBGrid

property DBGrid: TDBGrid;

Description

The *DBGrid* property points to the target TDBGrid component to import data to. To use a DBGrid instance as import destination, you should set the [ImportDestination](#) property to qidDBGrid.

See also:

[ImportDestination property](#)

[DataSet property](#)

[ListView property](#)

[StringGrid property](#)

2.1.2.8 ErrorLog

property ErrorLog: **boolean**;

Description

If the *ErrorLog* property is true then all the error messages arisen during the import process will be added as string values to the [Errors](#) property.

See also:

[ErrorRecs property](#)

[Errors property](#)

[OnImportError event](#)

[RewriteErrorLogFile](#)

2.1.2.9 ErrorLogFileName

property ErrorLogFileName: **string**;

Description

The *ErrorLogFileName* property defines the name of the file to store messages of all the errors occurred during import process. It is used only if the [ErrorLog](#) property is *True*.

See also:

[ErrorLog property](#)

[Errors property](#)

[ShowErrorLog property](#)

[RewriteErrorLogFile](#)

2.1.2.10 ErrorRecs

property ErrorRecs: **integer**;

Description

The *ErrorRecs* property is read-only and indicates the number of errors that occurred during the import process.

See also:

[ErrorLog property](#)

[Errors property](#)

[OnImportError event](#)

2.1.2.11 Errors

property Errors: TStrings;

Description

The *Errors* property is read-only and it stores all the error messages arisen during the import process. It is used only if the property [ErrorLog](#) is true, otherwise it will always be empty.

See also:

[ErrorLog property](#)

[ErrorRecs property](#)

[OnImportError](#)

2.1.2.12 FieldFormats

property Formats: [TQImportFieldFormats](#);

Description

The complex property *FieldFormats* is used in all the descendant components and determines the data formats of each imported field.

See also:

[TQImportFieldFormats object](#)

2.1.2.13 FileName

property FileName: **string**;

Description

The *FileName* property is used in all the *TQImport3* descendant components and determines the name of the file to import data from. This property must not be empty!

See also:

[DataSet property](#)

[Execute method](#)

2.1.2.14 Formats

property *Formats*: [TQImportFormats](#);

Description

The complex property *Formats* is used in all the descendant components and determines the type of the imported data.

See also:

[TQImportFormats object](#)

2.1.2.15 GridCaptionRow

property GridCaptionRow: **integer**;

Description

The *GridCaptionRow* property is used only if [ImportDestination](#) is *qidStringGrid*. It allows you to set the column captions instead of numbers in the [Map](#) property and defines the row in the target *StringGrid* to take captions from.

See also:

[GridStartRow property](#)

[DBGrid property](#)

2.1.2.16 GridStartRow

property GridStartRow: **integer**;

Description

The *GridStartRow* property is used only if [ImportDestination](#) is *qidStringGrid*. It defines the row number in the target StringGrid to start import from. The default value is -1, it means that new rows are added to the end of StringGrid when import begins.

See also:

[GridCaptionRow property](#)

[DBGrid property](#)

2.1.2.17 ImportDestination

type TQImportDestination = (qidDataSet, qidDBGrid, qidListView, qidStringGrid, qidUserDefined);

property ImportDestination: TQImportDestination;

Description

The *ImportDestination* property defines the type of the target object to import data to: TDataSet, TDBGrid, TListView or TStringGrid. Depending on the value of this property, one of the following properties are used for defining the target object: [DataSet](#), [DBGrid](#), [ListView](#) or [StringGrid](#). If you set the property value to qidUserDefined then you should process the imported value yourself using the OnUserDefinedImport event.

See also:

[DataSet property](#)

[DBGrid property](#)

[ListView property](#)

[StringGrid property](#)

2.1.2.18 ImportedRecs

property ImportedRecs: **integer**;

Description

The *ImportedRecs* property is read-only and it indicates the number of records successfully imported to the dataset.

See also:

[ImportRecCount property](#)

[OnImportRecord event](#)

2.1.2.19 ImportMode

type TQImportMode = (qimInsertAll, qimInsertNew, qimUpdate, qimUpdateOrInsert, qimDelete)

property ImportMode: TQImportMode;

Description

The *ImportMode* property allows you to define actions executed while importing data. The following values are available:

<i>qimInsertAll</i>	Inserts all the records from the source file to the target object
<i>qimInsertNew</i>	Inserts records which are not in the target object yet, others are skipped
<i>qimUpdate</i>	Updates those records which already exist in the target object, others are skipped
<i>qimUpdateOrInsert</i>	Updates existing records and inserts new records
<i>qimDelete</i>	Deletes those records which already exist in the target object, others are skipped
<i>qimDeleteOrInsert</i>	Deletes existing records and inserts new records

Note that for all of these modes, except InsertAll, you should define the [KeyColumns](#) property for defining the columns to search by.

See also:

[KeyColumns property](#)

[ImportDestination property](#)

2.1.2.20 ImportRecCount

property ImportRecCount: **integer**;

Description

The *ImportRecCount* property defines the number of records to import to the dataset. If its value is 0 then all the records will be imported, otherwise only the defined number of records will be imported.

See also:

[ImportedRecs property](#)

[OnImportRecord event](#)

2.1.2.21 KeyColumns

property KeyColumns: TStrings;

Description

The *KeyColumns* property is used if the [ImportMode](#) property is different from *qimInsertAll*. It defines columns which must be unique for searching the existing records in the target object.

See also:

[ImportMode property](#)

2.1.2.22 LastAction

```
type TQImportAction = (qiaNone, qiaInsert, qiaUpdate, qiaDelete);
```

```
property LastAction: TQImportAction;
```

Description

Read *LastAction* to get the last import action - *Insert*, *Update* or *Delete*, which depends on the [ImportMode](#) property value and on whether the duplicate value was met or not in the import destination ([ImportDestination](#) property).

You can use this property in [OnAfterPost](#) event, for example, for creating import logs.

See also:

[ImportMode](#) property

[OnAfterPost](#) event

2.1.2.23 ListView

property ListView: TListView;

Description

The *ListView* property points to the target *TListView* component to import data to. To use a *ListView* instance as import destination, you should set the [ImportDestination](#) property to *qidListView*.

See also:

[ImportDestination property](#)

[DataSet property](#)

[DBGrid property](#)

[StringGrid property](#)

2.1.2.24 Map

property Map: TStrings;

Description

The *Map* property is used in all the *TQImport3* descendant components and it sets the correspondence between the dataset fields and the imported table fields. It has its own implementation for each component.

See [TQImport3DBF: Map](#), [TQImport3XLS: Map](#), and [TQImport3ASCII: Map](#).

See also:

[TQImport3DBF: Map](#)

[TQImport3XLS: Map](#)

[TQImport3ASCII: Map](#)

2.1.2.25 RewriteErrorLogFile

property RewriteErrorLogFile: **boolean**;

Description

The *RewriteErrorLogFile* property defines whether the existing error log file defined by the [ErrorLogFileName](#) property, should be overwritten during import or not.

See also:

[ErrorLogFileName](#)

[ErrorLog](#)

[ShowErrorLog](#)

2.1.2.26 ShowErrorLog

property ShowErrorLog: **boolean**;

Description

If property *ShowErrorLog* is true then all the import error messages are shown on screen after the import is finished.

See also:

[ErrorLog property](#)

[ErrorLogFileName property](#)

[Errors property](#)

[RewriteErrorLogFile](#)

2.1.2.27 StringGrid

property StringGrid: TStringGrid;

Description

The *StringGrid* property points to the target *TStringGrid* component to import data to. To use a *StringGrid* instance as import destination, you should set the [ImportDestination](#) property to *qIdStringGrid*.

See also:

[ImportDestination property](#)

[DataSet property](#)

[DBGrid property](#)

[ListView property](#)

2.1.3 Methods

 Key methods

	Cancel
	Execute
	ImportToCSV
	LoadConfiguration
	SaveConfiguration

2.1.3.1 Cancel

procedure Cancel;

Description

The *Cancel* method stops the current import process initiated by the [Execute](#) method invocation. At the same time, the [OnImportCancel](#) event takes place.

See also:

[Canceled property](#)

[Execute method](#)

[OnImportCancel event](#)

2.1.3.2 Execute

function Execute: boolean;

Description

The *Execute* method is the central method of the collection. It executes the data import from the source file to the [DataSet](#). 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 [DataSet](#) properties settings are correct; in case of an error, the exceptions are raised and the method stops being executed, returning false as a result. If all the required properties are set correctly, the method starts its work with generating the [OnBeforeImport](#) event. After it, the data import to the dataset is started, and after the importing each record the event [OnImportRecord](#) takes place. On completion of the process, the event [OnAfterImport](#) is invoked. The import process can be interrupted by [Cancel](#) method invocation, in this case the [OnImportCancel](#) event is invoked.

See also:

[Cancel method](#)

[DataSet property](#)

[FileName property](#)

[OnBeforeImport event](#)

2.1.3.3 ImportToCSV

procedure ImportToCSV(Stream: TStream; Comma, Quote: **Char**);

Description

The *ImportToCSV* method executes import to the given stream. The imported data are formatted as CSV, using the parameters ListSeparator and Quote, and then are written to the Stream. If Quote is the #0 character, then no quotes are used.

2.1.3.4 LoadConfiguration

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

Description

The *LoadConfiguration* method allows you to load all the component property values (FileName, Map, Formats, etc.) from the text file, specified by FileName.

See also:

[SaveConfiguration method](#)

2.1.3.5 SaveConfiguration

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

Description

The *SaveConfiguration* method allows you to save all the component property values (FileName, Map, Formats, etc.) to the text file specified by *FileName*.

See also:

[LoadConfiguration method](#)

2.1.4 Events

Key events

-  [OnAfterImport](#)
-  [OnAfterPost](#)
-  [OnBeforeImport](#)
-  [OnBeforePost](#)
-  [OnDestinationLocate](#)
-  [OnImportCancel](#)
-  [OnImportError](#)
-  [OnImportRecord](#)
-  [OnNeedCommit](#)
-  [OnUserDefinedImport](#)

2.1.4.1 OnAfterImport

property OnAfterImport: TNotifyEvent;

Description

The *OnAfterImport* event takes place when the import is complete or it was interrupted by the [Cancel](#) method.

See also:

[OnBeforeImport event](#)

[Cancel method](#)

2.1.4.2 OnAfterPost

```
type TImportAfterPostEvent = procedure(Sender: TObject; Row: TQImportRow); of object;  
property OnAfterPost: TImportAfterPostEvent;
```

Description

The *OnAfterPost* event takes place after the imported value is posted to the [ImportDestination](#) object. Use this event to take any actions without changing the importing value, according to the Row parameter. You can also use the LastAction property value here.

See also:

[OnBeforePost event](#)
[OnAfterImport event](#)
[TQImportRow object](#)

2.1.4.3 OnBeforeImport

property OnBeforeImport: TNotifyEvent;

Description

The *OnBeforeImport* event takes place right before the import is started, i.e. when the [Execute](#) method is invoked.

See also:

[OnAfterImport event](#)

[Execute method](#)

2.1.4.4 OnBeforePost

```
type TImportBeforePostEvent = procedure(Sender: TObject; Row: TQImportRow; var Accept: boolean);  
property OnBeforePost: TImportBeforePostEvent;
```

Description

The *OnBeforePost* event takes place when the current table row is already delivered from the source table but not posted to the [ImportDestination](#) yet (if *ImportDestination* is not *qidUserDefined*). Row is the list containing of field names and values and allowing you to change the field value in the current row before it is posted. The *Accept* parameter defines if the current row is imported or not.

In the example below all the country names are imported in upper case and all the rows with population more than 10 000 000 are not imported at all.

```
procedure QImport1.BeforePost(Sender: TObject; Row: TQImportRow; var Accept: boolean);  
var  
    i: integer;  
begin  
    for i := 0 to Row.Count - 1 do begin  
        if Row[i].Name = 'Country' then begin  
            Row[i].Value := AnsiUpperCase(Row[i].Value);  
            Continue;  
        end;  
        if (Row[i].Name = 'Population') and (StrToInt(Row[i].Value) > 10000000) then begin  
            Accept := False;  
            Break;  
        end;  
    end;  
end;
```

See also:

[OnAfterPost event](#)
[OnUserDefinedImport](#)
[OnBeforeImport event](#)
[TQImportRow object](#)

2.1.4.5 OnDestinationLocate

type

TDestinationLocateEvent = **procedure**(Sender: TObject; KeyColumns: TStrings; Row: TQIDRow; var KeyFields: **string**; var KeyValues: **Variant**) of **object**;

property OnDestinationLocate: TDestinationLocateEvent;

Description

If the [ImportMode](#) property has any value except *qimInsertAll*, and the [ImportDestination](#) property is set to *qidDataSet* or *qidDBGrid*, and it is necessary to find the record by *KeyColumns* values, then the *TDataSet.Locate* method is called. You can use the *OnDestinationLocate* event to customize the *TDataSet.Locate* method parameters, such as *KeyFields* and *KeyValues*.

See also:

[ImportDestination property](#)

[ImportMode property](#)

2.1.4.6 OnImportCancel

```
type TImportCancelEvent = procedure(Sender: TObject; var Continue: boolean); of object  
property OnImportCancel: TImportCancelEvent;
```

Description

The *OnImportCancel* event takes place when the export process begun by the [Execute](#) method is interrupted by using the [Cancel](#) method invocation. Variable *Continue* defines whether the import process shall be stopped. If *Continue=False* then import will be aborted, otherwise it will be continued.

See also:

[OnAfterImport event](#)
[Cancel method](#)

2.1.4.7 OnImportError

property OnImportError: TNotifyEvent;

Description

The *OnImportError* event takes place when any error occurs during the import process.

See also:

[ErrorLog property](#)

[ErrorRecs property](#)

[Errors property](#)

2.1.4.8 OnImportRecord

property OnImportRecord: TNotifyEvent;

Description

The *OnImportRecord* event takes place after the importing data to each [Dataset](#) record. It is most frequently used to inform the user about the import execution process, e.g. to display the number of records exported, to increase the ProgressBar position, etc.

See also:

[ImportRecs property](#)

[ImportRecCount](#)

2.1.4.9 OnImportRowComplete

type TUserDefinedImportEvent = **procedure**(Sender: TObject; Row: [TQImportRow](#)); of **object**
property OnImportRowComplete: TUserDefinedImportEvent;

Description

The *OnImportRowComplete* event takes place when the import of row is complete.

See also:

[ImportRecs property](#)

[ImportRecCount](#)

2.1.4.10 OnNeedCommit

property OnNeedCommit: TNotifyEvent;

Description

The *OnNeedCommit* event takes place after importing a number of records, defined in the [CommitRecCount](#) property, or when the import is finished, if the [CommitAfterDone](#) is true. If neither of these properties are defined, the *OnNeedCommit* event will never take place.

Note that you should process the event manually to commit writing data to the dataset.

See also:

[CommitAfterDone property](#)

[CommitRecCount property](#)

2.1.4.11 OnUserDefinedImport

```
type TUserDefinedImportEvent = procedure(Sender: TObject; Row: TQImportRow); of object  
property OnUserDefinedImport: TUserDefinedImportEvent;
```

Description

The *OnUserDefinedImport* event takes place only if the [ImportDestination](#) property is *qidUserDefined*. It occurs each time the value is read from the source file and allows you to put the Row value to any destination you like.

See also:

[TQImportRow object](#)
[OnBeforePost event](#)

2.2 TADO_QImport3Access

2.2.1 TADO_QImport3Access Reference

Unit

[**QImport3Access**](#)

Description

The *TADO_QImport3Access* component is intended for importing data from the MS Access databases. Using the [SourceType](#) property you can receive data either from a table specified by the [TableName](#) property, or from the result of a query defined in the [SQL](#) property.

Note that *TADO_QImport3Access* installs only on Delphi 5 or higher, as it uses ADO.

2.2.2 Properties

▶ Run-time only

 Key properties

 [SkipFirstRows](#)
 [SourceType](#)
 [SQL](#)
 [TableName](#)

2.2.2.1 SkipFirstRows

property SkipFirstRows;

Description

The *SkipFirstRows* property defines the number of rows, counted from the first in the source table, which are not imported to the dataset. E.g. if you set SkipFirstRows=4 then the first four rows of the source Access table or query result are not imported.

2.2.2.2 SourceType

```
type TQImportAccessSourceType = (isTable, isSQL);
```

```
property SourceType: TQImportAccessSourceType;
```

Description

The *SourceType* property defines if data are imported from the Access table specified by the [TableName](#) property, or from the result of the query defined by the [SQL](#) property.

See also:

[SQL property](#)

[TableName property](#)

2.2.2.3 SQL

property SQL: TStrings;

Description

The *SQL* property sets the SQL text for the query to be executed which result is returned to the dataset. This property works only if [SourceType](#) is *isSQL*.

See also:

[SourceType property](#)

[TableName property](#)

2.2.2.4 TableName

property TableName: **string**;

Description

The *TableName* property defines the name of the Access table to import data from. This property works only if [SourceType](#) is *isTable*.

See also:

[SourceType property](#)

[SQL property](#)

2.3 TQImport3ASCII

2.3.1 TQImport3ASCII Reference

Unit

[TQImport3ASCII](#)

Description

The *TQImport3ASCII* component is used to import data from files in formats usually used as working or interchange [formats](#), i.e. Comma Separated Value (CSV) and Plain Text Format.

2.3.2 Properties

▶ Run-time only

 Key properties

	Comma
	CommitAfterDone
	CommitRecCount
	DataSet
	Encoding
	FieldFormats
	FileName
	Formats
	ImportRecCount
	Map
	Quote
	SkipFirstRows

2.3.2.1 Comma

property Comma: **AnsiChar**;

Description

The *Comma* property defines the character to be used as comma in the source table.

2.3.2.2 Encoding

property Encoding: TQICharsetType;

Description

The *Encoding* property defines the encoding to be used while importing to the dataset. By default, Windows encoding is used.

2.3.2.3 SkipFirstRows

property SkipFirstRows: **integer**;

Description

The *SkipFirstRows* property defines the number of records counted from the first in the source table which are not imported to the dataset. E.g. if you set SkipFirstRows=4 then the first four records of the source DBF table will not be imported.

2.3.2.4 Map

property Map: TStrings;

Description

Defining the *Map* property for the ASCII import depends on the type of the source table. If each column has its own fixed size then you should provide two integers for each dataset field: a position number (where column begins) and a column size.

E.g.

DataSet.FieldName1=4

or

DataSet.FieldName1=20;15.

See also:

[Map property](#) (TQImport3)

2.3.2.5 Quote

property Quote: **AnsiChar**;

Description

If the source data contains quoted characters, you should define the character used for quotation in the *Quote* property, e.g. `''`.

2.3.3 Events

 Key events

[OnAfterImport](#)
[OnBeforeImport](#)
[OnBeforePost](#)
[OnImportError](#)
[OnImportRecord](#)
[OnNeedCommit](#)

2.4 TQImport3DataSet

2.4.1 TQImport3DataSet Reference

Unit

[**TQImport3DataSet**](#)

Description

The *TQImport3DataSet* component is intended for moving data from one dataset defined by the [Source property](#) to the other one set as the [DataSet property](#) value.

2.4.2 Properties

▶ Run-time only

 Key properties

 [GoToFirstRecord](#)
 [SkipFirstRows](#)
 [Source](#)

2.4.2.1 GoToFirstRecord

property GoToFirstRecord: **boolean**;

Description

If GoToFirstRecord property is True, then import starts from the first record of the source dataset, otherwise from the current record.

2.4.2.2 SkipFirstRows

property SkipFirstRows;

Description

The SkipFirstRows property defines the number of rows, counted from the first in the source dataset, which are not imported to the dataset. E.g. if you set SkipFirstRows=4 then the first four rows of the source dataset are not imported.

2.4.2.3 Source

property Source: TDataSet;

Description

The Source property defines the source TDataSet component, import from which takes place.

2.5 TQImport3DBF

2.5.1 TQImport3DBF Reference

Unit

[TQImport3DBF](#)

Description

The *TQImport3DBF* component allows you to import your data from the DBF files.

2.5.2 Properties

▶ Run-time only

 Key properties

[CommitAfterDone](#)

[CommitRecCount](#)

[DataSet](#)

[FieldFormats](#)

[FileName](#)

[Formats](#)

[ImportRecCount](#)

 [SkipDeleted](#)

 [SkipFirstRows](#)

 [Map](#)

2.5.2.1 SkipDeleted

property SkipDeleted: **boolean**;

Description

If *SkipDeleted* is *true* (default), then all the records in the source DBF file marked as deleted are not imported to the target object.

2.5.2.2 SkipFirstRows

property SkipFirstRows: **integer**;

Description

The *SkipFirstRows* property defines the number of records, counted from the first in the source table, which are not imported to the dataset. E.g. if you set SkipFirstRows=4 then the first four records of the source DBF table will not be imported.

2.5.2.3 Map

property Map: TStrings;

Description

To set the correspondence between the dataset fields and the DBF table field you should define the *Map* property in the following way:

DataSet.FieldName1=TableFieldName1,
DataSetFieldName2=TableFieldName2, etc.

See also:

[Map property](#) (TQImport3)

2.5.3 Events

 Key events

[OnAfterImport](#)
[OnBeforeImport](#)
[OnBeforePost](#)
[OnImportError](#)
[OnImportRecord](#)
[OnNeedCommit](#)

2.6 TQImport3Docx

2.6.1 TQImport3Docx Reference

Unit

[TQImport3Docx](#)

Description

The *TQImport3Docx* component is intended for importing the MS Word 2007 files.

2.6.2 Properties

▶ Run-time only

 Key properties

[CommitAfterDone](#)

[CommitRecCount](#)

[DataSet](#)

[FieldFormats](#)

[FileName](#)

[Formats](#)

 [NeedFillMerge](#)

[ImportRecCount](#)

 [Map](#)

 [SkipFirstRows](#)

 [TableNumber](#)

2.6.2.1 Map

property Map: TStrings;

Description

To set the correspondence between the dataset fields and the Docx table column you should define the *Map* property in the following way:

DataSet.FieldName1=1,
DataSet.FieldName2=2, etc.

2.6.2.2 SkipFirstRows

property SkipFirstRows: **integer**;

Description

The *SkipFirstRows* property defines the number of records, counted from the first in the source table, which are not imported to the dataset. E.g. if you set SkipFirstRows=4 then the first four records of the source Docx table will not be imported.

2.6.2.3 NeedFillMerge

property NeedFillMerge: **Boolean**;

Description

The *NeedFillMerge* property defines whether the merged cells should be filled.

2.6.2.4 `TableNumber`

property `TableNumber`: **integer**;

Description

The *TableNumber* property sets the number of table to be imported.

2.6.3 Events

 Key events

[OnAfterImport](#)
[OnBeforeImport](#)
[OnBeforePost](#)
[OnImportError](#)
[OnImportRecord](#)
[OnNeedCommit](#)

2.7 TQImport3HTML

2.7.1 TQImport3HTML Reference

Unit

[TQImport3HTML](#)

Description

The *TQImport3HTML* component is used to import data from the tables contained in HTML files.

2.7.2 Properties

▶ Run-time only

 Key properties

[CommitAfterDone](#)

[CommitRecCount](#)

[DataSet](#)



[TableNumber](#)

[FieldFormats](#)

[FileName](#)

[Formats](#)

[ImportRecCount](#)



[Map](#)



[SkipFirstRows](#)

2.7.2.1 Map

property Map: TStrings;

Description

To set the correspondence between the dataset fields and the HTML table column you should define the Map property in the following way:

DataSet.FieldName1=1,
DataSet.FieldName2=2, etc.

2.7.2.2 SkipFirstRows

property SkipFirstRows: **integer**;

Description

The *SkipFirstRows* property defines the number of records, counted from the first in the source table, which are not imported to the dataset. E.g. if you set SkipFirstRows=4 then the first four records of the source HTML table will not be imported.

2.7.2.3 **TableNumber**

property TableNumber: **integer**;

Description

The *TableNumber* property defines the number of table in HTML document. E.g. if you set TableNumber=2 then the second table from the HTML document will be selected for import.

2.7.3 Events

 Key events

[OnAfterImport](#)
[OnBeforeImport](#)
[OnBeforePost](#)
[OnImportError](#)
[OnImportRecord](#)
[OnNeedCommit](#)

2.8 TQImport3ODS

2.8.1 TQImport3ODS Reference

Unit

[TQImport3ODS](#)

Description

The *TQImport3ODS* component is intended to import data from the OpenDocument Spreadsheet files.

2.8.2 Properties

▶ Run-time only

 Key properties

[CommitAfterDone](#)

[CommitRecCount](#)

[DataSet](#)

[FieldFormats](#)

[FileName](#)

[Formats](#)

 [NotExpandMergedValue](#)

[ImportRecCount](#)

 [Map](#)

 [SkipFirstRows](#)

 [SheetName](#)

2.8.2.1 Map

property Map: TStrings;

Description

To set the correspondence between the dataset fields and the OpenDocument Spreadsheet file column you should define the *Map* property in the following way:
DataSet.FieldName1=A,
DataSet.FieldName2=B, etc.

2.8.2.2 SkipFirstRows

property SkipFirstRows: **integer**;

Description

The *SkipFirstRows* property defines the number of records, counted from the first in the source table which are not imported to the dataset. E.g. if you set SkipFirstRows=4 then the first four records of the source ODS will not be imported.

2.8.2.3 SheetName

property SheetName: **string**;

Description

The *SheetName* property sets the name of the sheet.

2.8.2.4 NotExpandMergedValue

property NotExpandMergedValue: **Boolean**;

Description

The *NotExpandMergedValue* property defines whether the merged cell values should be spread (true by default, the merged cell values are not expanded).

2.8.3 Events

 Key events

[OnAfterImport](#)
[OnBeforeImport](#)
[OnBeforePost](#)
[OnImportError](#)
[OnImportRecord](#)
[OnNeedCommit](#)

2.9 TQImport3ODT

2.9.1 TQImport3ODT Reference

Unit

[TQImport3ODT](#)

Description

The *TQImport3ODT* component is intended for importing the OpenDocument Text files.

2.9.2 Properties

▶ Run-time only

 Key properties

[CommitAfterDone](#)

[CommitRecCount](#)

[DataSet](#)

[FieldFormats](#)

[FileName](#)

[Formats](#)

[ImportRecCount](#)

 [SheetName](#)

 [UseComplexTables](#)

 [SkipFirstRows](#)

 [Map](#)

2.9.2.1 Map

property Map: TStrings;

Description

To set the correspondence between the dataset fields and the OpenDocument Text table column you should define the *Map* property in the following way:

DataSet.FieldName1=A,

DataSet.FieldName2=B, etc.

2.9.2.2 SkipFirstRows

property SkipFirstRows: **integer**;

Description

The *SkipFirstRows* property defines the number of records, counted from the first in the source table which are not imported to the dataset. E.g. if you set SkipFirstRows=4 then the first four records of the source ODT table will not be imported.

2.9.2.3 SheetName

property SheetName: **string**;

Description

The *SheetName* property sets the name of the table.

2.9.2.4 UseComplexTables

property UseComplexTables: **Boolean**;

Description

The *UseComplexTables* property allows to import data from the ODT file table if it has a complex merged structure (when it can be parsed improperly).

2.9.3 Events

 Key events

[OnAfterImport](#)
[OnBeforeImport](#)
[OnBeforePost](#)
[OnImportError](#)
[OnImportRecord](#)
[OnNeedCommit](#)

2.10 TQImport3Wizard

2.10.1 TQImport3Wizard Reference

Unit

[TQImport3Wizard](#)

Description

The *TQImport3Wizard* component allows you to define all the import settings and import your data to the dataset from Excel, DBF, CSV, XML and TXT files within a dialog window. Using *TQImport3Wizard* you can add all the functionality of the *Advanced Data Import* components to your application within a single line of code!

2.10.2 Properties

▶ Run-time only

🔑 Key properties

🔑	AddType
🔑	AllowedImports
🔑	AutoChangeExtension
🔑	AutoLoadTemplate
🔑	AutoSaveTemplate
🔑	CloseAfterImport
🔑	CommitAfterDone
🔑	CommitRecCount
🔑	ConfirmOnCancel
🔑	CSVViewerRows
🔑	DataSet
🔑	DBGrid
🔑	ErrorLog
🔑	ErrorLogFileName
🔑	ExcelMaxColWidth
🔑	ExcelViewerRows
🔑	FieldFormats
🔑	FileName
🔑	Formats
🔑	GoToLastPageAfterLoadTemplate
🔑	GridCaptionRow
🔑	GridStartRow
🔑	ImportDestination
🔑	ImportMode
🔑	ImportRecCount
🔑	KeyColumns
🔑	ListView
🔑	Picture
🔑	RewriteErrorLogFile
🔑	ShowErrorLog
🔑	ShowHelpButton
🔑	ShowProgress
🔑	ShowSaveLoadButtons
🔑	StringGrid
🔑	TemplateFileName
🔑	TextViewerRows

2.10.2.1 AddType

```
type TQImportAddType = (qatAppend, qatInsert);
```

```
property AddType: TQImportAddType;
```

Description

The *AddType* property defines what statement - *Append* or *Insert* - is used when writing data to the [ImportDestination](#). The default value is *qatInsert*.

2.10.2.2 AllowedImports

```
type TAllowedImport = (aiXLS, aiDBF, aiXML, aiTXT, aiCSV);
```

```
type TAllowedImports = set of TAllowedImport;
```

```
property AllowedImports: TAllowedImports;
```

Description

The *AllowedImports* property defines the import types available in the dialog window. Add or remove values from the property to enable or disable corresponding import types.

See also:

[AutoChangeExtension property](#)

[FileName property](#)

2.10.2.3 AutoChangeExtension

property AutoChangeExtension: **boolean**;

Description

If the *AutoChangeExtension* property is *true* then on choosing the import type in the dialog window the file extension will be changed automatically in the 'Source Filename' edit field.

See also:

[AllowedImports property](#)

[FileName property](#)

2.10.2.4 AutoLoadTemplate

property AutoLoadTemplate: **boolean**;

Description

If the *AutoLoadTemplate* property is *true*, then all the import settings are automatically load on starting wizard from the template file specified by the [TemplateFileName](#) property.

See also:

[AutoSaveTemplate property](#)

[TemplateFileName property](#)

2.10.2.5 AutoSaveTemplate

property AutoSaveTemplate: **boolean**;

Description

If the *AutoSaveTemplate* property is *true*, then all the import settings are automatically saved on wizard finish to the template file specified by the [TemplateFileName](#) property.

See also:

[AutoLoadTemplate property](#)

[TemplateFileName property](#)

2.10.2.6 CloseAfterImport

property CloseAfterImport: boolean;

Description

If the *CloseAfterImport* property is *True*, then the Wizard closes automatically after the import process is finished.

2.10.2.7 CommitAfterDone

property CommitAfterDone: **boolean**;

Description

If the *CommitAfterDone* property is *True* then the [OnNeedCommit event](#) takes place after the import finished.

See also:

[CommitRecCount property](#)

[OnNeedCommit event](#)

2.10.2.8 CommitRecCount

property CommitRecCount: **integer**;

Description

The *CommitRecCount* property defines the number of records after importing which the [OnNeedCommit event](#) takes place.

See also:

[CommitAfterDone property](#)

[OnNeedCommit event](#)

2.10.2.9 ConfirmOnCancel

property ConfirmOnCancel: **boolean**;

Description

The *ConfirmOnCancel* property defines whether user is asked for confirmation on trying to cancel import or not.

2.10.2.10 CSVViewerRows

property CSVViewerRows: **integer**;

Description

Use the *CSVViewerRows* property to define the number of the CSV file rows shown in the grid on the Step 1 of the wizard if import from CSV is selected.

See also:

[ExcelViewerRows property](#)

[TextViewerRows property](#)

2.10.2.11 DataSet

property DataSet: TDataSet;

Description

The *DataSet* property points to the target TDataSet component to import data to. To use a DataSet instance as import destination, you should set the [ImportDestination](#) property to qidDataSet (default) and the dataset which must be opened (DataSet.Active = true) before the import begins (before the [Execute](#) method is invoked).

See also:

[ImportDestination property](#)

[DBGrid property](#)

[ListView property](#)

[StringGrid property](#)

[Execute method](#)

2.10.2.12 DBGrid

property DBGrid: TDBGrid;

Description

The *DBGrid* property points to the target TDBGrid component to import data to. To use a DBGrid instance as import destination, you should set the [ImportDestination](#) property to qidDBGrid.

See also:

[ImportDestination property](#)

[DataSet property](#)

[ListView property](#)

[StringGrid property](#)

[Execute method](#)

2.10.2.13 ErrorLog

property ErrorLog: **boolean**;

Description

If *ErrorLog* property is true then all the error messages arisen during the import process are stored in the file specified by the [ErrorLogFileName property](#).

See also:

[ErrorLogFileName property](#)

[ShowErrorLog property](#)

[OnImportError event](#)

2.10.2.14 ErrorLogFileName

property ErrorLogFileName: **string**;

Description

The the *ErrorLogFileName* property defines the name of the file to store messages of all the errors occurred during import process. It is used only if the [ErrorLog property](#) is True.

See also:

[ErrorLog property](#)

[RewriteErrorLogFile property](#)

[ShowErrorLog property](#)

[OnImportError event](#)

2.10.2.15 ExcelMaxColWidth

property ExcelMaxColWidth: **integer**;

Description

The the *ExcelMaxColWidth* property defines the maximum width of the Excel grid columns shown on the Step 1 of the wizard if import from Excel is selected.

See also:

[ExcelViewerRows property](#)

2.10.2.16 ExcelViewerRows

property ExcelViewerRows: **integer**;

Description

Use the *ExcelViewerRows* property to define the number of the Excel grid rows shown on the Step 1 of the wizard if import from Excel is selected.

See also:

[ExcelMaxColWidth property](#)

[CSVViewerRows property](#)

[TextViewerRows property](#)

2.10.2.17 FieldFormats

property Formats: [TQImportFieldFormats](#);

Description

The complex property *FieldFormats* determines the data formats of each certain imported field.

See also:

[TQImportFieldFormats object](#)
[Formats property](#)

2.10.2.18 FileName

property FileName: **string**;

Description

The *FileName* property determines the name of the file to import data from. This property must not be empty!

See also:

[ImportDestination property](#)

[Execute method](#)

2.10.2.19 Formats

property *Formats*: [TQImportFormats](#);

Description

The complex property *Formats* determines the type of the imported data.

See also:

[TQImportFormats object](#)

[FieldFormats property](#)

2.10.2.20 GoToLastPageAfterLoadTemplate

property GoToLastPage: **boolean**;

Description

If the *GoToLastPage* property is *true*, then, on executing import, the last wizard page is displayed right after the template is loaded into the wizard skipping other pages.

See also:

[AutoLoadTemplate property](#)

[AutoSaveTemplate property](#)

2.10.2.21 GridCaptionRow

property GridCaptionRow: **integer**;

Description

The *GridCaptionRow* property is used only if [ImportDestination](#) is *qidStringGrid*. It allows you to set the column captions instead of numbers while setting the column correspondence in the wizard and defines the row in the target *StringGrid* to take captions from.

See also:

[ImportDestination property](#)

[GridStartRow property](#)

2.10.2.22 GridStartRow

property GridStartRow: **integer**;

Description

The *GridStartRow* property is used only if [ImportDestination](#) is *qidStringGrid*. It defines the row number in the target StringGrid to start import from. The default value is -1, it means that new rows are added to the end of *StringGrid* when import begins.

See also:

[ImportDestination property](#)

[GridCaptionRow property](#)

2.10.2.23 ImportDestination

```
type TQImportDestination = (qidDataSet, qidDBGrid, qidListView, qidStringGrid);
```

```
property ImportDestination: TQImportDestination;
```

Description

The *ImportDestination* property defines the type of the target object to import data to: TDataSet, TDBGrid, TListView or TStringGrid. Depending on the value of this property, one of the following properties are used for defining the target object: [DataSet](#), [DBGrid](#), [ListView](#) or [StringGrid](#).

See also:

[DataSet property](#)

[DBGrid property](#)

[ListView property](#)

[StringGrid property](#)

2.10.2.24 ImportMode

type TQImportMode = (qimInsertAll, qimInsertNew, qimUpdate, qimUpdateOrInsert, qimDelete)

property ImportMode: TQImportMode;

Description

The *ImportMode* property allows you to define actions executed while importing data. The following values are available:

<i>qimInsertAll</i>	Inserts all the records from the source file to the target object
<i>qimInsertNew</i>	Inserts records which are not in the target object yet, others are skipped
<i>qimUpdate</i>	Updates those records which already exist in the target object, others are skipped
<i>qimUpdateOrInsert</i>	Updates existing records and inserts new records
<i>qimDelete</i>	Deletes those records which already exist in the target object, others are skipped
<i>qimDeleteOrInsert</i>	Deletes existing records and inserts new records

Note that for all of these modes, except *InsertAll*, you should define the [KeyColumns](#) property for defining the columns to search by.

See also:

[KeyColumns property](#)

2.10.2.25 ImportRecCount

property ImportRecCount: **integer**;

Description

The *ImportRecCount* property defines the number of records to import to the dataset. If its value is 0 then all the records will be imported, otherwise only the defined number of records will be imported.

See also:

[OnImportRecord event](#)

2.10.2.26 KeyColumns

property KeyColumns: TStrings;

Description

The *KeyColumns* property is used if the [ImportMode](#) property is different from *qimInsertAll*. It defines columns which must be unique for searching the existing records in the target object.

See also:

[ImportMode property](#)

2.10.2.27 ListView

property ListView: TListView;

Description

The *ListView* property points to the target *TListView* component to import data to. To use a *ListView* instance as import destination, you should set the [ImportDestination](#) property to *qidListView*.

See also:

[DataSet property](#)

[DBGrid property](#)

[StringGrid property](#)

[ImportDestination property](#)

[Execute method](#)

2.10.2.28 Picture

property Picture: TPicture;

Description

Use the *Picture* property to set the picture shown on starting the wizard (on the "Import from" step). Note that the default picture width is 122 and its height is 346.

2.10.2.29 RewriteErrorLogFile

property RewriteErrorLogFile: **boolean**;

Description

If the *RewriteErrorLogFile* property is *True* then the errors are recorded into the file specified by the [ErrorLogFileName property](#), even if such file already exists.

See also:

[ErrorLog property](#)

[ErrorLogFileName property](#)

2.10.2.30 ShowErrorLog

property ShowErrorLog: **boolean**;

Description

If property *ShowErrorLog* is *true* then all the import error messages are shown on screen after the import is finished.

See also:

[ErrorLog property](#)

[ErrorLogFileName property](#)

[OnImportError event](#)

2.10.2.31 ShowHelpButton

property ShowHelpButton: **boolean**;

Description

If the *ShowHelpButton* property is *true*, then the 'Help' button will be visible in the wizard window. Using this button the user will be able to access the component help file.

See also:

[ShowProgress property](#)

[ShowSaveLoadButtons property](#)

2.10.2.32 ShowProgress

property ShowProgress: **boolean**;

Description

If the *ShowProgress* property is true then the progress bar showing the import progress will be visible in the wizard window.

See also:

[ShowHelpButton property](#)

[ShowSaveLoadButtons property](#)

2.10.2.33 ShowSaveLoadButtons

property ShowSaveLoadButtons: **boolean**;

Description

If the *ShowSaveLoadButtons* property is *true*, then buttons 'Save template to file' and 'Load template to file' will be visible in the wizard window. Using this buttons the user will be able to save the template with different import options (source filename, field correspondence, format options, etc) to file and then load this template for fastening the import process.

See also:

[ShowHelpButton property](#)

[ShowProgress property](#)

2.10.2.34 StringGrid

property StringGrid: TStringGrid;

Description

The *StringGrid* property points to the target *TStringGrid* component to import data to. To use a *StringGrid* instance as import destination, you should set the [ImportDestination](#) property to *qidStringGrid*.

See also:

[DataSet property](#)

[DBGrid property](#)

[ListView property](#)

[ImportDestination property](#)

[Execute method](#)

2.10.2.35 TemplateFileName

property TemplateFileName: **string**;

Description

The *TemplateFileName* property defines the template file where all the import settings are stored. If properties [AutoSaveTemplate](#) and [AutoLoadTemplate](#) are *true*, then import settings are automatically saved to and loaded from this file on each wizard session.

See also:

[AutoSaveTemplate property](#)

[AutoLoadTemplate property](#)

2.10.2.36 TextViewerRows

property TextViewerRows: **integer**;

Description

Use the *TextViewerRows* property to define the number of the text file rows shown on the Step 1 of the wizard if import from TXT is selected.

See also:

[CSVViewerRows property](#)

[ExcelViewerRows property](#)

2.10.3 Methods

 Key methods

[Execute](#)

2.10.3.1 Execute

function Execute: **boolean**;

Description

The *Execute* method is the central method of the suite. It executes the data import from the source file to the [ImportDestination](#). 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 [ImportDestination](#) properties settings are correct; in case of an error, the exceptions are raised and the method stops being executed, returning false as a result. If all the required properties are set correctly, the method starts its work with generating the [OnBeforeImport](#) event. After it, the data import to the dataset is started, and after the importing each record the event [OnImportRecord](#) takes place. On completion of the process, the event [OnAfterImport](#) is invoked.

2.10.4 Events

 Key events

[OnAfterImport](#)
[OnBeforeImport](#)
[OnBeforePost](#)
[OnDestinationLocate](#)
[OnImportCancel](#)
[OnImportError](#)
[OnImportRecord](#)
[OnLoadTemplate](#)
[OnNeedCommit](#)

2.10.4.1 OnAfterImport

property OnAfterImport: TNotifyEvent;

Description

The *OnAfterImport* event takes place when the export is complete and it was interrupted by the [Cancel](#) method.

2.10.4.2 OnBeforeImport

property OnBeforeImport: TNotifyEvent;

Description

The *OnBeforeImport* event takes place right before the import is started, i.e. when the [Execute](#) method is invoked.

2.10.4.3 OnBeforePost

```
type TImportBeforePostEvent = procedure(Sender: TObject; Row: TQImportRow; var Accept: boolean)  
property OnBeforePost: TImportBeforePostEvent;
```

Description

The *OnBeforePost* event takes place when the current table row is already delivered from the source table but not posted to the [ImportDestination](#) yet. TQImport3 row is the list, containing of field names and values and allowing you to change the field value in the current row before it is posted. The Accept parameter defines if the current row is imported or not.

In the example below all the country names are imported in upper case and all the rows with population more than 10 000 000 are not imported at all.

```
procedure QImport31.BeforePost(Sender: TObject; Row: TQImportRow; var Accept: boolean)  
var  
    i: integer;  
begin  
    for i := 0 to Row.Count - 1 do begin  
        if Row[i].Name = 'Country' then begin  
            Row[i].Value := AnsiUpperCase(Row[i].Value);  
            Continue;  
        end;  
        if (Row[i].Name = 'Population') and (StrToInt(Row[i].Value) > 10000000) then begin  
            Accept := False;  
            Break;  
        end;  
    end;  
end;
```

2.10.4.4 OnDestinationLocate

type

TDestinationLocateEvent = **procedure**(Sender: TObject; KeyColumns: TStrings; Row: TQIDRow; var KeyFields: **string**; var KeyValues: **Variant**) of **object**;

property OnDestinationLocate: TDestinationLocateEvent;

Description

If the [ImportMode](#) property has any value except *qimInsertAll*, and the [ImportDestination](#) property is set to *qidDataSet* or *qidDBGrid*, and it is necessary to find the record by *KeyColumns* values, then the *TDataSet.Locate* method is called. You can use the *OnDestinationLocate* event to customize the *TDataSet.Locate* method parameters, such as *KeyFields* and *KeyValues*.

See also:

[ImportDestination property](#)

[ImportMode property](#)

2.10.4.5 OnImportCancel

```
type TImportCancelEvent = procedure(Sender: TObject; var Continue: boolean); of object  
property OnImportCancel: TImportCancelEvent;
```

Description

The *OnImportCancel* event takes place when the export process begun by the [Execute](#) method is interrupted. Variable *Continue* defines whether the import process should be stopped. If *Continue=False* then import will be aborted, otherwise it will be continued.

2.10.4.6 OnImportError

property OnImportError: TNotifyEvent;

Description

The *OnImportError* takes place when any error occurs during the import process.

2.10.4.7 OnImportRecord

property OnImportRecord: TNotifyEvent;

Description

The *OnImportRecord* event takes place after the importing data to each [ImportDestination](#) record. It is most frequently used to inform the user about the import execution process, e.g. to display the number of records exported, to increase the ProgressBar position, etc.

2.10.4.8 OnLoadTemplate

```
type TImportLoadTemplateEvent = procedure(Sender: TObject; const FileName: string); of  
property OnLoadTemplate: TImportLoadTemplateEvent;
```

Description

The *OnLoadTemplate* event takes place after the template is loaded to the wizard. According to the *FileName* parameter which stands for the template file name, you can take any actions you need.

2.10.4.9 OnNeedCommit

property OnNeedCommit: TNotifyEvent;

Description

The *OnNeedCommit* event takes place after importing a number of records, defined in the [CommitRecCount](#) property, or when the import is finished, if the [CommitAfterDone](#) is true. If neither of these properties are defined, the *OnNeedCommit* event will never take place.

Note that you should process the event manually to commit writing data to the dataset.

2.11 TQImport3XLS

2.11.1 TQImport3XLS Reference

Unit

[TQImport3XLS](#)

Description

The *TQImport3XLS* component is intended for importing the MS Excel e-tables to the dataset. MS Office 97-2003 and higher supported.

2.11.2 Properties

▶ Run-time only

 Key properties

[CommitAfterDone](#)

[CommitRecCount](#)

[DataSet](#)

[FieldFormats](#)

[FileName](#)

[Formats](#)

 [ImportRange](#)

[ImportRecCount](#)

 [Map](#)

 [SkipFirstCols](#)

 [SkipFirstRows](#)

2.11.2.1 ImportRange

```
type TQImportRange = (qirMax, qirMin);
```

```
property ImportRange: TQImportRange;
```

Description

As number of records may vary in different columns, you should define when import should be stopped: on importing the longest column or on reaching the last record of the shortest column. Set the *ImportRange* property on *qirMax* or *qirMin* in accordance to define this.

See also:

[Map property](#)

2.11.2.2 Map

property Map: TStrings;

Description

To set the correspondence between the dataset fields and the Excel cells you should define the *Map* property in the following way:

FieldName=CellRange

You can define the CellRange string depending on your needs.

Separate cells

To import separate cells, define the CellRange string as set of cell identifiers, e.g. A1, B2, C3, separated by semicolon.

Example

Field1=A1

Field1=A1;B2;C3;

Column

To import entire column or its part, define the string in the following format:

FieldName=FirstCell-LastCell

Example

Imported cells

Field1=A1-A10

From A1 up to A10

Field1=A10-A1

From A10 down to A1

Field1=COLSTART-

From the first cell with data in the column A up to A10

Field1=A10-

From A10 down to the first cell with data in the column A

Field1=A10-

From A10 up to the last cell with data in the column A

Field1=COLFINISH- From the last cell with data in the column A down to A10

Field1=A-COLFINISH From the first up to the last cell with data in the column A

Field1=A-COLSTART From the last down to the first cell with data in the column A

Rows

To import entire row or its part, define the string in the following format:

FieldName=FirstCell-LastCell

Example

Imported cells

Field1=A1-D1 From A1 up to D1

Field1=D1-A1 From D1 down to D1

Field1=ROWSTART- From the first cell with data in the row 1 up to F1

Field1=F1- From F1 down to the first cell with data in the row 1

Field1=A10- From A10 up to the last cell with data in the row 10

Field1=ROWFINISH- From the last cell with data in the row 10 down to A10

Field1=10- From the first up to the last cell with data in the row 10

Field1=10- From the last down to the first cell with data in the row 10

Defining sheets

To define the specific sheet, use the following string format:

FieldName=[*SheetName*]*FirstCell-LastCell*

or

FieldName=[*:SheetNumber*]*FirstCell-LastCell*

Example **Imported cells**

Field1=[*Sheet1*]*A1*- From A1 up to A10 at the sheet named Sheet1

Field1=[*:3*]*A1-A10* From A1 up to A10 at the sheet number 3

You can mix cell ranges as you need.

Example

Field1=*A1;A3;A10-A15;A15-D15;D15-COLFINISH*;*[Sheet1]COLFINISH-A1*

See also:

[ImportRange property](#)

[Map property](#) (TQImport3)

2.11.2.3 SkipFirstCols

property SkipFirstCols: **integer**;

Description

The *SkipFirstCols* property defines the number of columns, counted from the first in the source table which are not imported to the dataset. E.g. if you set SkipFirstCols=4 then the first four columns of the source Excel table will not be imported.

See also:

[SkipFirstRows property](#)

2.11.2.4 SkipFirstRows

property SkipFirstRows: **integer**;

Description

The *SkipFirstRows* property defines the number of rows counted from the first in the source table which are not imported to the dataset. E.g. if you set SkipFirstRows=4 then the first four rows of the source Excel table will not be imported.

See also:

[SkipFirstCols property](#)

2.11.3 Events

 Key events

[OnAfterImport](#)
[OnBeforeImport](#)
[OnBeforePost](#)
[OnImportError](#)
[OnImportRecord](#)
[OnNeedCommit](#)

2.12 TQImport3XLSx

2.12.1 TQImport3XLSx Reference

Unit

[TQImport3XLSx](#)

Description

The *TQImport3XLSx* component is intended for importing the MS Excel e-tables to the dataset.

2.12.2 Properties

▶ Run-time only

 Key properties

[CommitAfterDone](#)

[CommitRecCount](#)

[DataSet](#)

[FieldFormats](#)

[FileName](#)

[Formats](#)

 [NeedFillMerge](#)

[ImportRecCount](#)

 [Map](#)

 [LoadHiddenSheets](#)

 [SkipFirstRows](#)

 [SheetName](#)

2.12.2.1 Map

property Map: TStrings;

Description

To set the correspondence between the dataset fields and the Excel 2007 table cells you should define the *Map* property in the following way:

FieldName=CellRange

You can define the CellRange string depending on your needs.

Separate cells

To import separate cells, define the CellRange string as set of cell identifiers, e.g. A1, B2, C3, separated by semicolon.

Example

Field1=A1

Field1=A1;B2;C3;

Column

To import entire column or its part, define the string in the following format:

FieldName=FirstCell-LastCell

Example

Imported cells

Field1=A1-A10

From A1 up to A10

Field1=A10-A1

From A10 down to A1

Field1=COLSTART-

From the first cell with data in the column A up to A10

Field1=A10-

From A10 down to the first cell with data in the column A

Field1=A10-

From A10 up to the last cell with data in the column A

Field1=COLFINISH- From the last cell with data in the column A down to A10

Field1=A1- From the first up to the last cell with data in the column A

Field1=A1- From the last down to the first cell with data in the column A

Rows

To import entire row or its part, define the string in the following format:

FieldName=FirstCell-LastCell

Example

Imported cells

Field1=A1-D1 From A1 up to D1

Field1=D1-A1 From D1 down to D1

Field1=ROWSTART- From the first cell with data in the row 1 up to F1

Field1=F1- From F1 down to the first cell with data in the row 1

Field1=A10- From A10 up to the last cell with data in the row 10

Field1=ROWFINISH- From the last cell with data in the row 10 down to A10

Field1=10- From the first up to the last cell with data in the row 10

Field1=10- From the last down to the first cell with data in the row 10

Defining sheets

To define the specific sheet, use the following string format:

FieldName=[SheetName]FirstCell-LastCell

or

FieldName=[:SheetNumber]FirstCell-LastCell

Example

Imported cells

Field1=[Sheet1]A1- From A1 up to A10 at the sheet named Sheet1

Field1=[:3]A1-A10 From A1 up to A10 at the sheet number 3

You can mix cell ranges as you need.

Example

Field1=A1;A3;A10-A15;A15-D15;D15-COLFINISH;[Sheet1]COLFINISH-A1

2.12.2.2 SkipFirstRows

property SkipFirstRows: **integer**;

Description

The *SkipFirstRows* property defines the number of rows, counted from the first in the source table which are not imported to the dataset. E.g. if you set SkipFirstRows=4 then the first four rows of the source Excel 2007 table will not be imported.

2.12.2.3 LoadHiddenSheets

property LoadHiddenSheets: **Boolean**;

Description

The *LoadHiddenSheets* property defines whether hidden sheets should be loaded.

2.12.2.4 NeedFillMerge

property NeedFillMerge: **Boolean**;

Description

The *NeedFillMerge* property defines whether the merged cells should be filled.

2.12.2.5 SheetName

property SheetName: string;

Description

The *SheetName* property sets the name of the sheet.

2.12.3 Events

 Key events

[OnAfterImport](#)
[OnBeforeImport](#)
[OnBeforePost](#)
[OnImportError](#)
[OnImportRecord](#)
[OnNeedCommit](#)

2.13 TQImport3XML

2.13.1 TQImport3XML Reference

Unit

[QImport3XML](#)

Description

Use the *TQImport3XLS* component to import your data from the XML files created via the *Advanced Data Export* Component Suite (<http://www.sqlmanager.net/en/products/tools/advancedexport>) or *TClientDataSet* component.

2.13.2 Properties

▶ Run-time only

 Key properties

[CommitAfterDone](#)

[CommitRecCount](#)

[DataSet](#)

[FieldFormats](#)

[FileName](#)

[Formats](#)

[ImportRecCount](#)

 [SkipFirstRows](#)

 [WriteOnFly](#)

2.13.2.1 SkipFirstRows

property SkipFirstRows;

Description

The *SkipFirstRows* property defines the number of rows counted from the first in the source file which are not imported to the dataset. E.g. if you set SkipFirstRows=4 then the first four rows of the source XML file are not imported.

2.13.2.2 WriteOnFly

property WriteOnFly : **boolean**;

Description

The *WriteOnFly* property determines how the source xml file will be read and imported. If *WriteOnFly* is set to *False* then the source file will be read and parsed completely, and after that the prepared data will be imported to the destination object (e.g. TDataSet, TDBGrid etc.). If *WriteOnFly* is set to true each tag will be read, parsed and imported separately without temporary saving to memory.

2.14 TQImport3XMLDoc

2.14.1 TQImport3XMLDoc Reference

Unit

[TQImport3XMLDoc](#)

Description

The *TQImport3XMLDoc* component is intended for importing the any XML documents.

2.14.2 Properties

▶ Run-time only

 Key properties

	CommitAfterDone
	CommitRecCount
	DataSet
	XPath
	FieldFormats
	FileName
	Formats
	ImportRecCount
	Map
	SkipFirstRows
	DataLocation

2.14.2.1 XPath

property XPath: qiString;

Description

The *XPath* property defines the XPath addressing parts of an XML document (path must be specified in the XPath format), e.g. /DATAPACKET/ROWDATA/ROW.

For more information on the XPath please visit <http://www.w3.org/TR/xpath>

2.14.2.2 SkipFirstRows

property SkipFirstRows: **integer**;

Description

The *SkipFirstRows* property defines the number of records counted from the first in the source table which are not imported to the dataset. E.g. if you set SkipFirstRows=4 then the first four records of the source XML document will not be imported.

2.14.2.3 Map

property Map: TStrings;

Description

To set the correspondence between the dataset fields and the XML file column you should define the *Map* property in the following way:

DataSet.FieldName1=1,
DataSet.FieldName2=2, etc.

2.14.2.4 DataLocation

type TXMLDataLocation = (tlAttributes, tlSubNodes);

property DataLocation: TXMLDataLocation;

Description

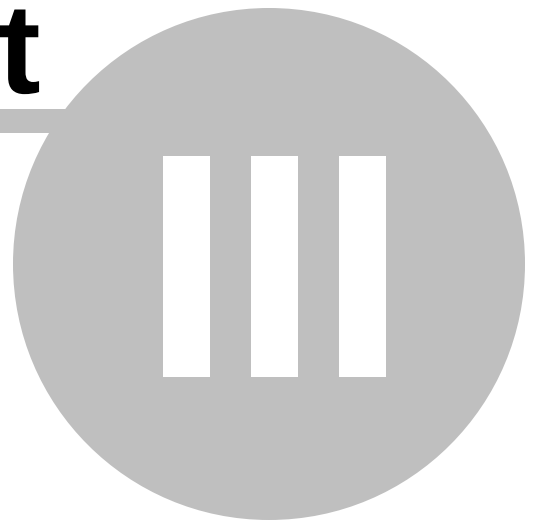
The *DataLocation* property defines the location of data in XML document (attributes and subnodes text). E.g. if you set DataLocation = tlAttributes the data will be taken from the attributes.

2.14.3 Events

 Key events

[OnAfterImport](#)
[OnBeforeImport](#)
[OnBeforePost](#)
[OnImportError](#)
[OnImportRecord](#)
[OnNeedCommit](#)

Part



3 Units

3.1 QImport3 unit

Components

[TQImport3](#)

Objects

[TQImportFormats](#)

[TQImportRow](#)

[TQimportCol](#)

[TQImportFieldFormats](#)

[TQImportFieldFormat](#)

[TQImportReplacements](#)

[TQImportReplacement](#)

3.1.1 TQImportFormats object

Unit

[QImport3](#)

Description

The *TQImportFormats* class is used in all the descendant TQImport3 components and it contains properties, which determine the imported data formats.

3.1.1.1 Properties

▶ Run-time only

 Key properties

	<u>BooleanFalse</u>
	<u>BooleanTrue</u>
	<u>DecimalSeparator</u>
	<u>LongDateFormat</u>
	<u>LongTimeFormat</u>
	<u>NullValues</u>
	<u>ShortDateFormat</u>
	<u>ShortTimeFormat</u>
	<u>ThousandSeparator</u>

3.1.1.1.1 BooleanFalse

property BooleanFalse: TStrings;

Description

The *BooleanFalse* property contains the variants of FALSE value representation in the imported table, e.g. '-', 'No', etc.

See also:

[BooleanTrue property](#)

3.1.1.1.2 BooleanTrue

property BooleanTrue: TStrings;

Description

The *BooleanTrue* property contains the variants of TRUE value representation in the imported table, e.g. '+', 'Yes', etc. When import to the boolean field of the dataset takes place, the imported string will be compared with all the strings of the BooleanTrue property, and if the imported string is not found, then the dataset field value will be false.

See also:

[BooleanFalse property](#)

3.1.1.1.3 DateSeparator

property DateSeparator: **char**;

Description

The *DateSeparator* property defines the character, delimiting months, days and years in the source date fields.

See also:

[TimeSeparator property](#)

3.1.1.1.4 DecimalSeparator

property DecimalSeparator: **char**;

Description

The *DecimalSeparator* property defines the character which denotes the decimal values in the imported float fields, e.g. '.' or ','.

See also:

[ThousandSeparator property](#)

3.1.1.1.5 LongDateFormat

property LongDateFormat: **string**;

Description

The *LongDateFormat* property defines the format of the imported long date fields, e.g. 'mm.dd.yy'.

See also:

[LongTimeFormat property](#)

[ShortDateFormat property](#)

[ShortTimeFormat property](#)

3.1.1.1.6 LongTimeFormat

property LongTimeFormat: **string**;

Description

LongTimeFormat defines the format of the imported long time format, e.g. '20:10:32'.

See also:

[LongDateFormat property](#)

[ShortDateFormat property](#)

[ShortTimeFormat property](#)

3.1.1.1.7 NullValues

property NullValues: TStrings;

Description

Use the *NullValues* property to set a value or a number of values which are interpreted as nulls on import.

3.1.1.1.8 ShortDateFormat

property ShortDateFormat: **string**;

Description

The *ShortDateFormat* property defines the format of the imported long date fields, e.g. '02.28.2002'.

See also:

[LongDateFormat property](#)

[LongTimeFormat property](#)

[ShortTimeFormat property](#)

3.1.1.1.9 ShortTimeFormat

property ShortTimeFormat: **string**;

Description

The *ShortTimeFormat* property defines the format of the imported short time fields, e.g. '20:28'.

See also:

[LongDateFormat property](#)

[LongTimeFormat property](#)

[ShortDateFormat property](#)

3.1.1.1.10 ThousandSeparator

property ThousandSeparator: **char**;

Description

The *ThousandSeparator* property defines the character which separates the digit groups in the imported fields, e.g. '.' or ','.

See also:

[DecimalSeparator property](#)

3.1.1.1.11 TimeSeparator

property TimeSeparator: **char**;

Description

The *TimeSeparator* property defines the character delimiting minutes and hours in the source date fields.

See also:

[DateSeparator property](#)

3.1.2 TQImportRow object

Unit

[QImport3](#)

Description

The *TQImportRow* object contains the currently imported table row which was already delivered from the source table, but have not yet been posted to the dataset. This object is used in the [OnBeforePost](#) event.

3.1.2.1 Properties

▶ Run-time only  Key properties

▶  [Items](#)

3.1.2.1.1 Items

property Items: [TQImportCol](#);

Description

The *Items* property contains fields of the currently imported table row which was already delivered from the source table, but have not yet been posted to the dataset. These data are stored as a set of [TQImportCol](#) values, which contain information of each certain field of the table row.

Use this property in processing the [OnBeforePost](#) and [OnUserDefinedImport](#) events.

See also:

[TQImportCol object](#)

[OnBeforePost event](#)

3.1.3 TQImportCol object

Unit

[QImport3](#)

Description

Each *TQImportCol* object contains one certain field of the currently imported table row ([TQImportRow](#) object).

3.1.3.1 Properties

▶ Run-time only

 Key properties

▶  [Name](#)

▶  [Value](#)

3.1.3.1.1 Name

property Name: **string**;

Description

The *Name* property determines the [TQImportCol](#) field as it contains the column name of the currently imported table row ([TQImportRow](#) object).

See also:

[TQImportRow component](#)

[Value property](#)

3.1.3.1.2 Value

property Value: {\$IFDEF UNICODE}WideString{\$ELSE}string{\$ENDIF};

Description

The *Value* property contains the string value of the [TQImportCol](#) field.

See also:

[TQImportRow component](#)

[Name property](#)

3.1.4 TQImportFieldFormat object

Unit

[QImport3](#)

Description

TQImportFieldFormat represents an item in the [TQImportFieldFormats](#) collection. The properties of this object allow you to set various data formats for the certain dataset field. These properties are read in the definite order: generator properties ([GeneratorValue](#) and [GeneratorStep](#)), [ConstantValue](#), Null properties ([NullValue](#) and [DefaultValue](#)), Quotation properties ([QuoteAction](#), [LeftQuote](#) and [RightQuote](#)) and String Conversion properties ([CharCase](#) and [CharSet](#)).

Thus, if you set *GeneratorStep* to a value other than 0, no other properties are taken into consideration; if you set *ConstantValue* or *NullValue* and *DefaultValue* properties, then you can set only Quotation properties and String Conversion properties.

3.1.4.1 Properties

▶ Run-time only

 Key properties

	CharCase
	CharSet
	ConstantValue
	DefaultValue
	FieldName
	GeneratorStep
	GeneratorValue
	NullValue
	QuoteAction
	LeftQuote
	RightQuote
	Replacements

3.1.4.1.1 CharCase

```
type TQImportCharCase = (iccNone, iccUpper, iccLower, iccUpperFirst, iccUpperFirstWord);
```

```
property CharCase: TQImportCharCase;
```

Description

The *CharCase* property defines the character case of the field data. The following values are available:

<i>iccNone</i>	keep the original character case
<i>iccUpper</i>	set the whole string to upper case
<i>iccLower</i>	set the whole string to lower case
<i>iccUpperFirst</i>	set the first letter of the string to upper case
<i>iccUpperFirstWord</i>	set the first letter of each word to upper case

See also:

[CharSet property](#)

3.1.4.1.2 CharSet

```
type TQImportCharSet = (icsNone, icsAnsi, icsOem);
```

```
property CharSet: TQImportCharSet;
```

Description

The *CharSet* property defines the character set of the field data. The following values are available:

<i>icsNone</i>	save the original character set
<i>icsAnsi</i>	set the field character set to ANSI
<i>icsOem</i>	set the field character set to OEM

See also:

[CharCase property](#)

3.1.4.1.3 ConstantValue

property ConstantValue: {\$IFDEF UNICODE}WideString{\$ELSE}string{\$ENDIF};

Description

The *ConstantValue* property defines the value of the constant field.

3.1.4.1.4 DefaultValue

property DefaultValue: {\$IFDEF UNICODE}WideString{\$ELSE}string{\$ENDIF};

Description

Use *DefaultValue* to define the value to set when the value set as [NullValue](#) is imported.

See also:

[NullValue property](#)

3.1.4.1.5 *FieldName*

property *FieldName*: **string**;

Description

The *FieldName* property contains the name of the field to apply the formats to.

3.1.4.1.6 GeneratorStep

property GeneratorStep: **integer**;

Description

The *GeneratorStep* property defines the step of the autoincrement field.

See also:

[GeneratorValue property](#)

3.1.4.1.7 GeneratorValue

property GeneratorValue: **integer**;

Description

The *Generator* value defines the initial value of the autoincrement field.

See also:

[GeneratorStep property](#)

3.1.4.1.8 NullValue

property NullValue: {\$IFDEF UNICODE}WideString{\$ELSE}string{\$ENDIF};

Description

Use *NullValue* to set the value understood as Null in the imported string. If such value is imported, then the value of the [DefaultValue](#) property will be set.

See also:

[ConstantValue property](#)

[DefaultValue property](#)

3.1.4.1.9 QuoteAction

```
type TQuoteAction = (qaNone, qaAdd, qaRemove);
```

```
property QuoteAction: TQuoteAction;
```

Description

The *QuoteAction* property defines the action taken on importing the quotation string to the field. If this property is *qaNone* then no action is taken, if it is *qaAdd* then quotation marks will be added to the imported string, and if it is *qaRemove* then the quotation marks will be removed from the imported quotation string.

See also:

[LeftQuote property](#)

[RightQuote property](#)

3.1.4.1.10 LeftQuote

property LeftQuote: {\$IFDEF UNICODE}WideString{\$ELSE}string{\$ENDIF};

Description

The *LeftQuote* property defines the characters which denote quoting in the imported string, e.g. '<<'.

See also:

[QuoteAction property](#)

[RightQuote property](#)

3.1.4.1.11 RightQuote

property RightQuote: {\$IFDEF UNICODE}WideString{\$ELSE}string{\$ENDIF};

Description

The *RightQuote* property defines the characters which denote quoting in the imported string, e.g. '>>'.

See also:

[QuoteAction property](#)

[LeftQuote property](#)

3.1.4.1.12 Replacements

property Replacements: [TQImportReplacements](#);

Description

Use this property to define the imported field values that are to be replaced and the new values for this fields. Use the [TQImportReplacements](#) collection items for to define the text to find and to replace it with, and also the case-sensitivity.

See also:

[TQImportReplacements object](#)

3.1.5 TQImportFieldFormats object

Unit

[QImport3](#)

Description

The *TQImportFieldFormats* object is the collection of [TQImportFieldFormat](#) objects, which allow you to define data formats for each dataset field.

3.1.5.1 Properties

▶ Run-time only  Key properties

▶  [Items](#)

3.1.5.1.1 Items

property Items: TQImportFieldFormat;

Description

Use *Items* to access individual items in the collection. The value of the *Index* parameter corresponds to the Index property of [TQImportFieldFormat](#). It represents the position of the item in the collection.

3.1.6 TQImportReplacement object

Unit

[QImport3](#)

Description

Use the *TQImportReplacement* object to set the replacement lists for each imported field.

3.1.6.1 Properties

▶ Run-time only

 Key properties

 [IgnoreCase](#)
 [ReplaceWith](#)
 [TextToFind](#)

3.1.6.1.1 IgnoreCase

property IgnoreCase: **boolean**;

Description

The *IgnoreCase* property defines if search for the [TextToFind](#) text is case-sensitive or not.

3.1.6.1.2 ReplaceWith

property ReplaceWith: {\$IFDEF UNICODE}WideString{\$ELSE}string{\$ENDIF};

Description

The *ReplaceWith* property contains value to replace imported text from the [TextToFind](#) property.

See also:

[TextToFind](#) property

3.1.6.1.3 TextToFind

property TextToFind: {\$IFDEF UNICODE}WideString{\$ELSE}string{\$ENDIF};

Description

The *TextToFind* property defines the text to be found during import and replaced with the [ReplaceWith](#) property value.

See also:

[ReplaceWith property](#)

3.1.7 TQImportReplacements object

Unit

[QImport3](#)

Description

The *TQImportReplacements* object is the collection of [TQImportReplacement](#) objects, which allow you to set the list of the replacement values for each imported field.

3.1.7.1 Properties

▶ Run-time only  Key properties

▶  [Items](#)

3.1.7.1.1 Items

property Items[Index: **integer**]: [TQImportReplacement](#);

Description

Use *Items* to access individual items in the collection. The value of the *Index* parameter corresponds to the Index property of [TQImportReplacement](#). It represents the position of the item in the collection.

3.2 QImport3Access unit

Components

[TQImport3Access](#)

3.3 QImport3ASCII unit

Components

[TQImport3ASCII](#)

3.4 QImport3DataSet unit

Components

[TQImport3DataSet](#)

3.5 QImport3DBF unit

Components

[TQImport3DBF](#)

3.6 QImport3Docx unit

Components

[TQImport3Docx](#)

3.7 QImport3HTML unit

Components

[TQImport3HTML](#)

3.8 QImport3ODS unit

Components

[TQImport3ODS](#)

3.9 QImport3ODT unit

Components

[TQImport3ODT](#)

3.10 QImport3Wizard unit

Components

[TQImport3Wizard](#)

3.11 QImport3XLS unit

Components

[TQImport3XLS](#)

3.12 QImport3XLSx unit

Components

[TQImport3XLSx](#)

3.13 QImport3XML unit

Components

[TQImport3XML](#)

3.14 QImport3XMLDoc unit

Components

[TQImport3XMLDoc](#)

Part

IV

4 Advanced Data Import Wizard Guide

EMS Advanced Data Import Wizard guides you through the process of importing data to the dataset.

First specify the format of the source file. For details refer to [Supported file formats](#). If you import data from the CSV file you should also define the character, delimiting columns in the source table, and characters, which stand for left and right quotation marks in the source table.

Import data wizard - Import from country.xls

Import from

- ☒ MS Excel 97-2003
- ☐ MS Excel
- ☐ MS Word
- ☐ ODS file
- ☐ ODT file
- ☐ DBF
- ☐ XML

XML document type

- ☐ Text file
- ☐ CSV file

Comma Quote

- ☐ HTML file
- ☐ XMLDoc
- ☐ MS Access 97-2003
- ☐ MS Access

Password

Source file name

C:\local\workspace\d10.4\vd-dataimport\demos\data\country.xls

☐ Skip Invisible Target Columns

Template Options

- ☐ Go to the last page after loading template
- ☐ Save template automatically when wizard closes

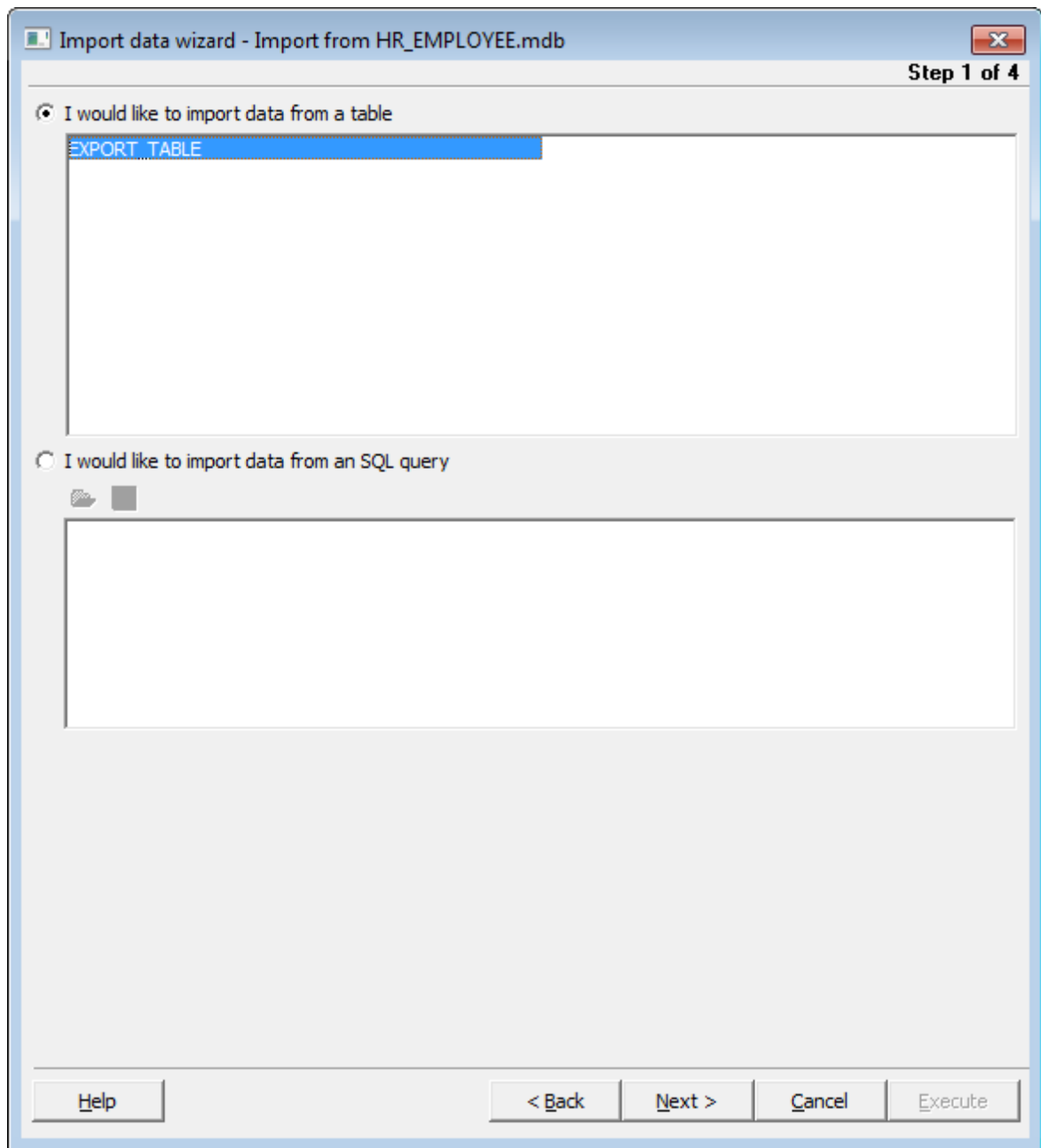
Set the name of the file to import data from in the **Source file name** edit field.

Load template from file - use this button to fasten the process of configuring your current import by loading previously saved template with various import options (source filename, field correspondence, format options, etc) from file.

Click 'Next' to proceed to the first step.

4.1 Selecting Data Source

This step of the wizard is only available when you are importing data from *MS Access*. First select the data source for import - MS Access table or SQL query. If you choose import from a table, then you should select a table name from the list, if you choose to import from a query, you should set the query SQL text in the lower area, e.g.
`SELECT * FROM COUNTRY WHERE CONTINENT='South America'.`



Click 'Next' to proceed to the next step.

4.2 Setting Correspondence

This step of the wizard allows you **to set correspondence** between columns of the source file and fields of the table they are imported to.

- [MS Excel](#)
- [MS Access](#)
- [DBF](#)
- [TXT](#)
- [CSV](#)

To get more information about the file formats, see the [Supported file formats](#) page.

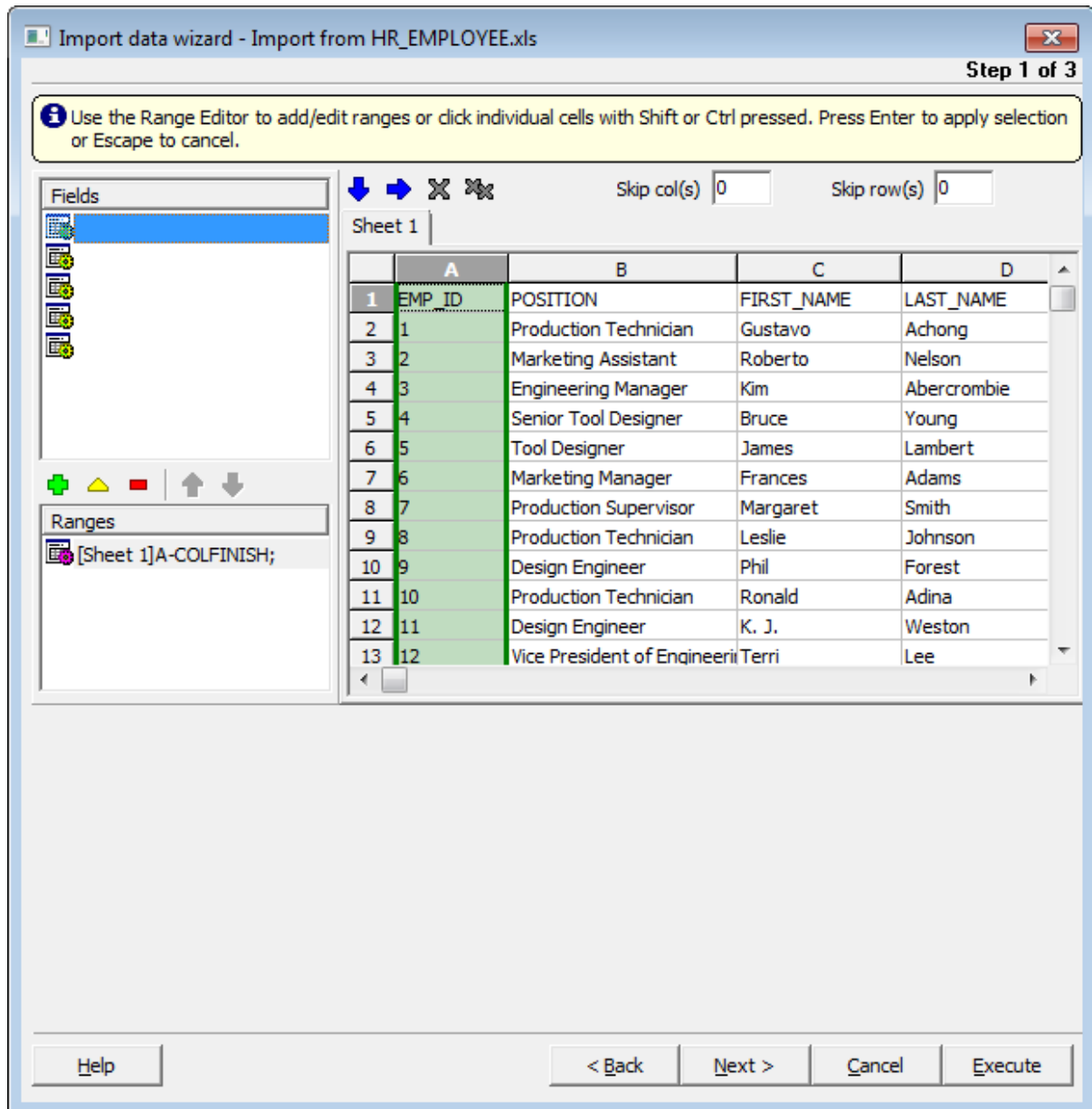
Click 'Next' to continue or click 'Back' to return to the previous step.

4.2.1 MS Excel

First select the dataset field from the 'Fields' drop-down list. Then select the corresponding cells by clicking row or column caption (to select the whole row or column) or clicking the individual cells, using Shift and Ctrl keys.

You can also define the corresponding cells manually in the 'Cells' edit field. Use semicolon to separate multiple cells.

After you select all the corresponding cells for the current table field, proceed to another field and repeat all these operations for each dataset field.



If you don't want some first rows or columns of the source table to be imported, set the number of such rows in the 'Skip ... first row(s)' and 'Skip first ... col(s)' edit fields.

Auto fill cols - use this button to set the correspondence between the source table

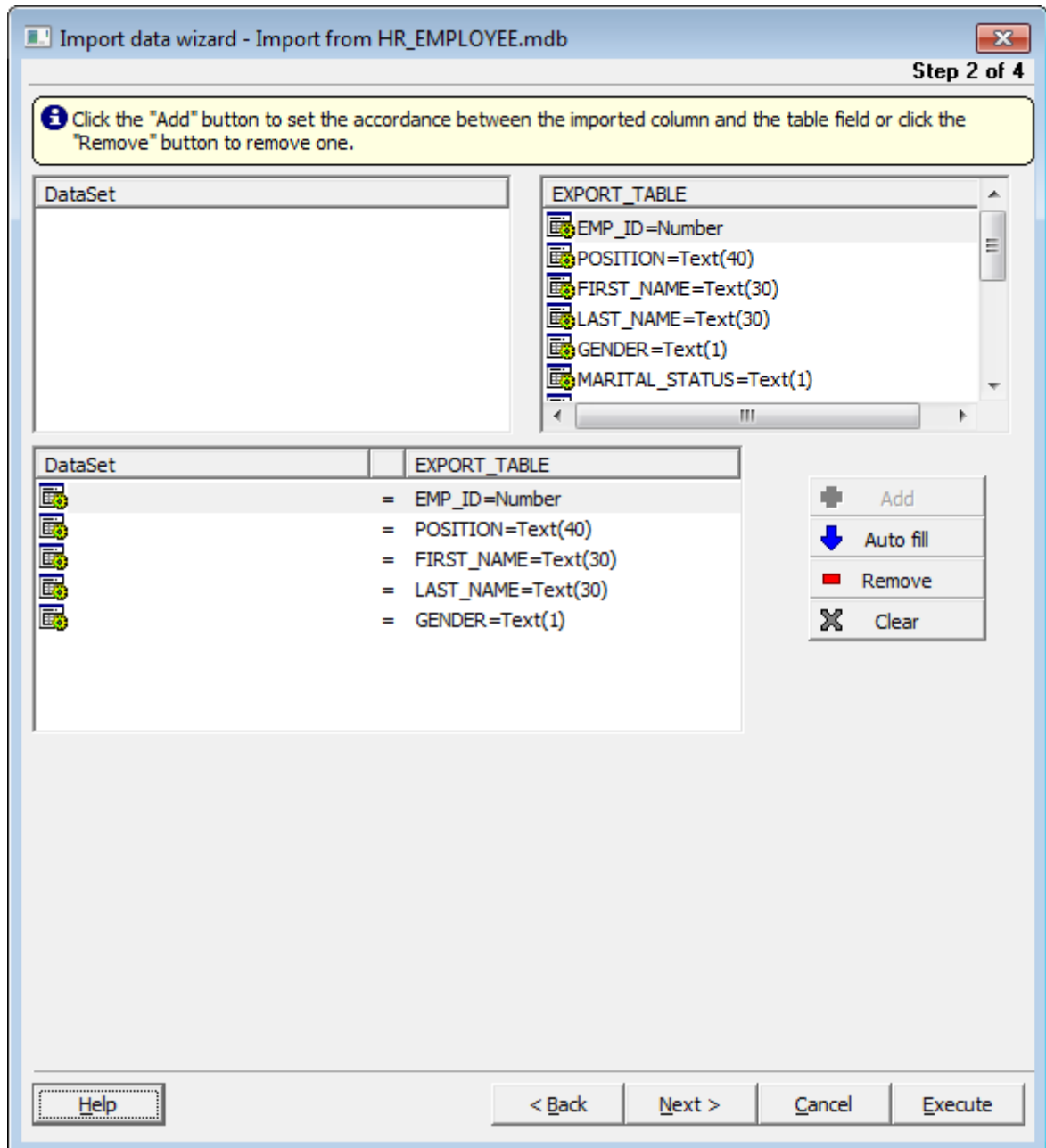
columns and the dataset fields automatically. It is convenient if they are ordered in the same way. First table column will correspond to the first dataset field, second column to the second field, etc. If quantity of the table columns exceeds quantity of the dataset fields, then the last columns will have no correspondence.

Auto fill rows - use this button to set the correspondence between the source table rows and the dataset fields automatically. It is convenient if they are ordered in the same way. First table row will correspond to the first dataset field, second row to the second field, etc. If quantity of the table rows exceeds quantity of the dataset fields, then the last rows will have no correspondence.

4.2.2 MS Access

First select the dataset field from the 'DataSet' list. Then select the corresponding field in the '<TABLE_NAME>' or 'Custom query' list.

Click button 'Add' to link these fields. These fields will be added to the list at the bottom of the window. Repeat these operations for each dataset field. If you want to remove the accordance you set, select the linked fields in the bottom list and click button 'Remove'.



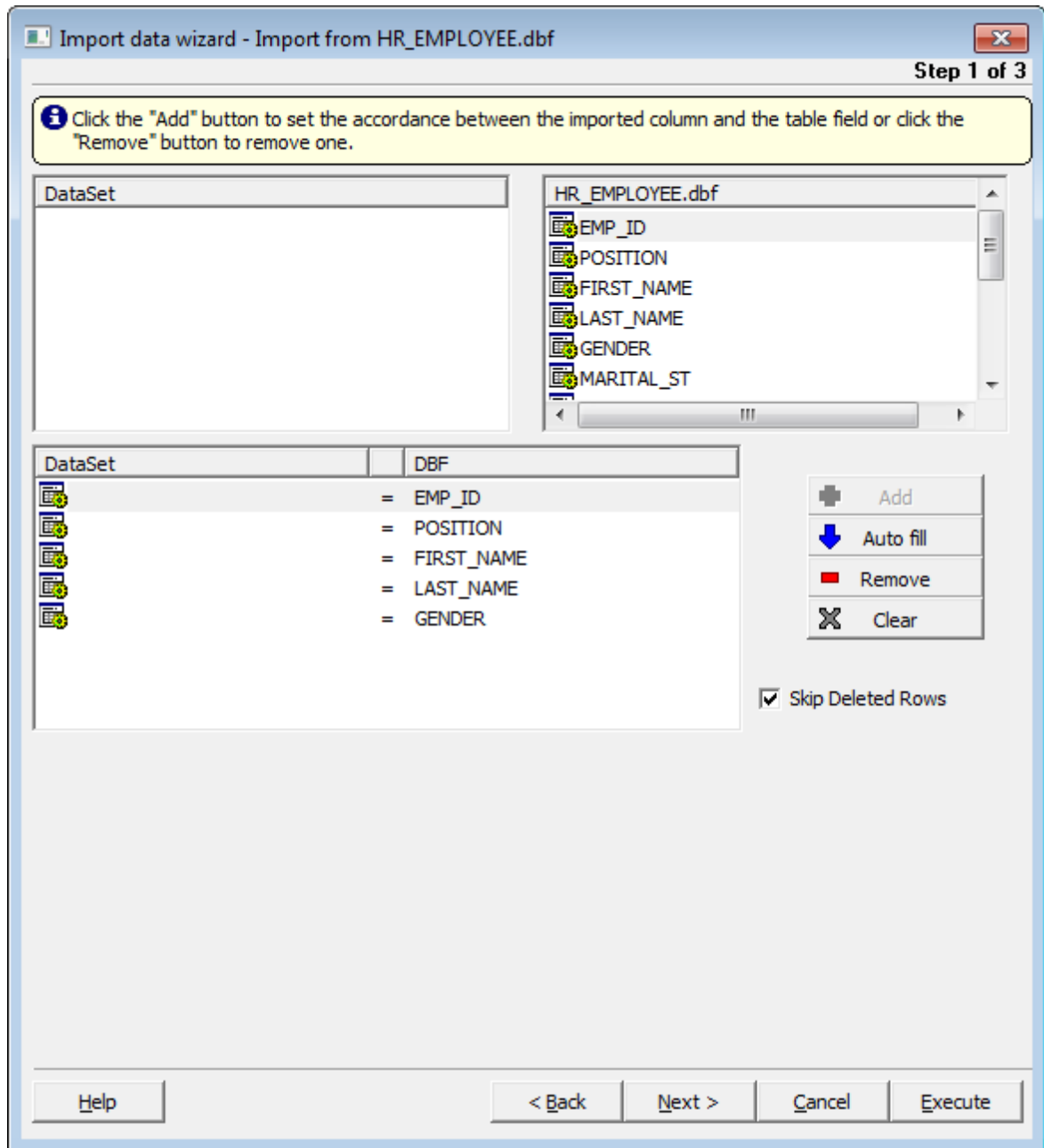
Auto fill - use this button to set the correspondence between the source table fields and the dataset fields automatically. It is convenient if they are ordered in the same way. First table field will correspond to the first dataset field, second field to the second field, etc. If quantity of the table fields exceeds quantity of the dataset fields, then the last

fields will have no correspondence.

4.2.3 DBF

First select the dataset field from the 'DataSet' list. Then select the corresponding field in the '<TABLE_NAME>.DBF' list.

Click button 'Add' to link these fields. These fields will be added to the list at the bottom of the window. Repeat these operations for each dataset field. If you want to remove the accordance you set, select the linked fields in the bottom list and click button 'Remove'.

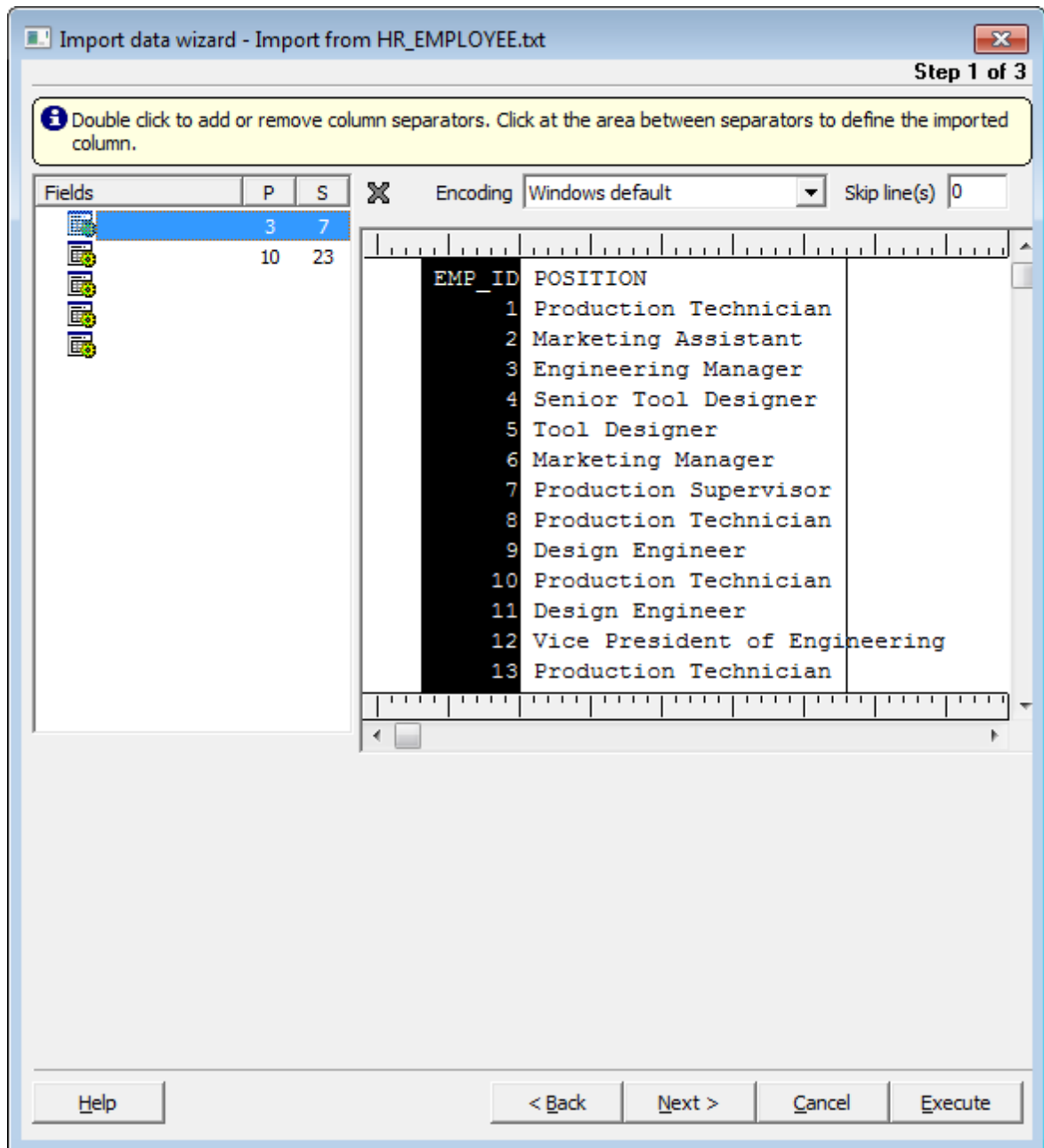


Auto fill - use this button to set the correspondence between the source table fields and the dataset fields automatically. It is convenient if they are ordered in the same way. First table field will correspond to the first dataset field, second field to the second field,

etc. If quantity of the table fields exceeds quantity of the dataset fields, then the last fields will have no correspondence.

4.2.4 TXT

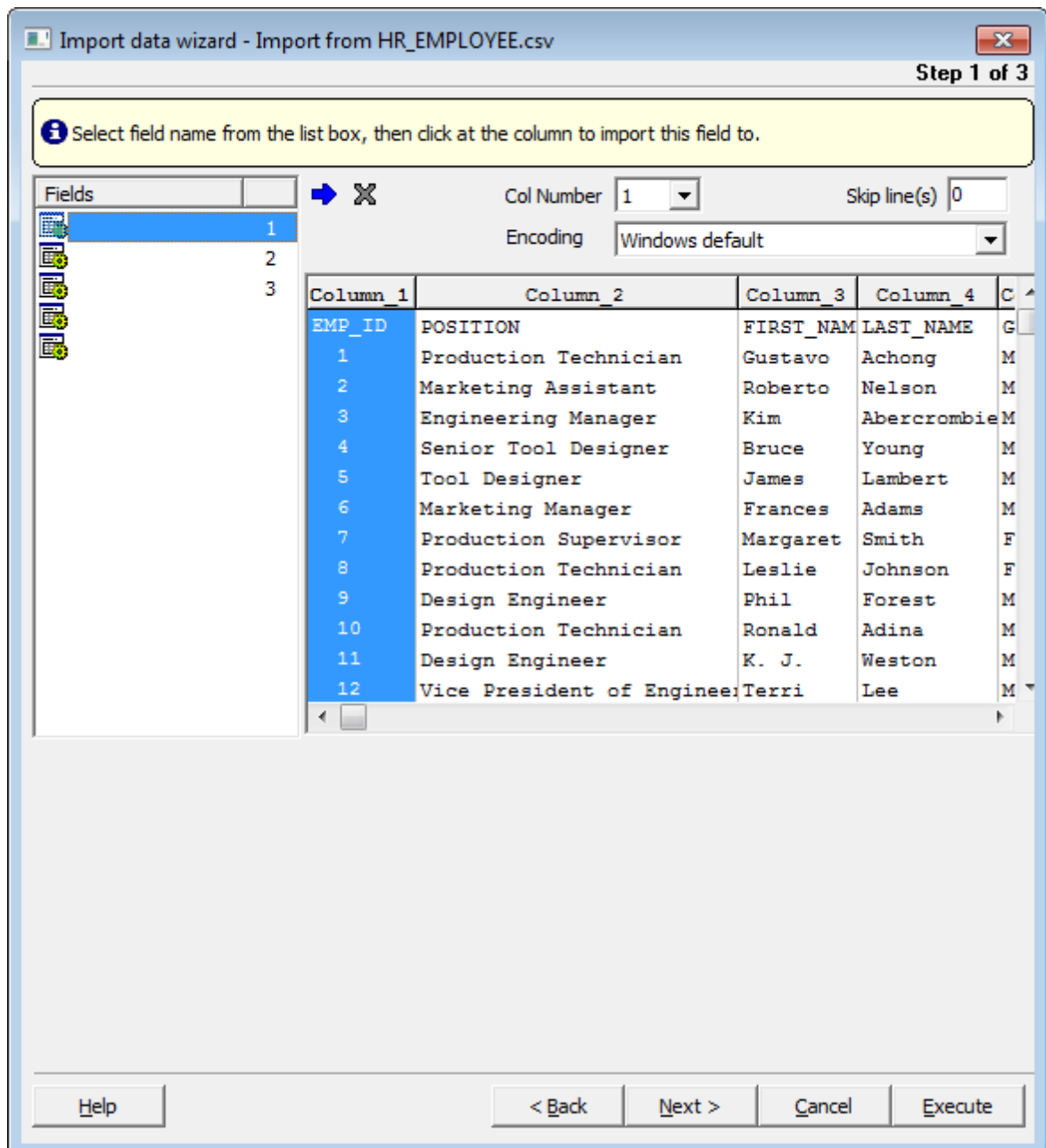
First select the dataset field from the 'Fields' drop-down list. Then set two separator lines to delimit the source table column. Click to add a separator, double-click to delete one. Drag separators to change the column width. You can also set the column starting position and the column width manually in the edit fields 'Pos' and 'Size'. When you set the separators correctly, proceed to another field and repeat these operations for each dataset field.



If you don't want some first rows of the source table to be imported set the number of such rows in the 'Skip ... first line(s)' edit field.

4.2.5 CSV

If the delimiter you have defined on the first step was found in the source table, then you will find the table columns already separated and delimited. Select the dataset field from the 'Fields' drop-down list. Then click the corresponding source table column or set the 'Col' value manually. Repeat these operations for each dataset field.



If you don't want some first rows of the source table to be imported set the number of such rows in the 'Skip ... first line(s)' edit field.

Auto fill - use this button to set the correspondence between the source table columns and the dataset fields automatically. It is convenient if they are ordered in the same way.

First table column will correspond to the first dataset field, second column to the second field, etc. If quantity of the table columns exceeds quantity of the dataset fields, then the last columns will have no correspondence.

4.3 Specifying Base Formats

Regional settings

Decimal separator - set a character, which delimits the decimal parts of the imported numbers.

Thousand separator - set a character, which separates the digit groups in the imported numbers.

Short date format, Long date format, Short time format, Long time format - use these edit fields to set the date and time formats.

Format options

Left quotation - set a character or a number of characters, which denote quoting in the imported strings.

Right quotation - set a character or a number of characters, which denote unquoting in the imported strings.

Quotation action - you can select 'Add' to add quotation marks to each imported string or 'Remove' to remove all the quotation marks from the imported strings.

Import data wizard - Import from HR_EMPLOYEE.xls

Step 2 of 3

Base formats | Data formats

Date and Time formats

Short date: dd/MM/yyyy

Long date: d MMMM yyyy 'r/'

Short time: h:mm

Long time: h:mm:ss

Separators

Decimal: ,

Thousand: #160

Date: .

Time: :

Boolean true

True

Boolean false

False

Null Values

Null

Help < Back Next > Cancel Execute

Boolean true - set some variants of TRUE value representation in the imported table, e. g. 'Yes' or '+'. Use new line for each new variant.

Boolean false - set some variants of FALSE value representation in the imported table, e.g. 'No' or '-'. Use new line for each new variant.

Data Formats

On this tab you can customize the format of each imported field in case when additional formatting is required. Select the field in the 'Field Name' list and set its format in the proper edit fields.

Tuning

Generator Value - use this edit field to set the initial value of the autoincrement field.

Generator Step - set the step of the autoincrement field. If it is 0 then the value of the generator will be ignored.

Constant Value - use this edit field to set the constant value of the field.

Null Value - set the value, which will be understood as NULL to set the default value.

Default Value - set the default value of the NULL field.

Left quotation - set a character or a number of characters, which denote quoting in the imported string.

Right quotation - set a character or a number of characters, which denote unquoting in the imported string.

Quotation action - you can select 'Add' to add quotation marks to the imported string, 'Remove' to remove all the quotation marks from the imported string or 'As is' to save the original quotation marks.

Char case - set the case of the imported string. 'As is' saves the original string, 'Upper' sets the whole string to upper case, 'Lower' sets the whole string to lower case, 'UpperFirst' sets the first letter of the string to upper case, 'UpperFirstWord' sets the first letter of each word to upper case.

Char set - set the char set of the imported string to ANSI or OEM. 'As is' saves the original string char set.

Replacements

Use this tab to set the replacement list for the selected field. Fill the list in the following format:

<Value-to-find>=<Replace-with-Value>.

E.g., you set the following replacements for the field 'Continent':

'South America'='S. America'

'North America'='N. America',

that means, that all the values 'South America' of the field 'Continent' will be replaced with values 'S. America', and values 'North America' will be replaced with 'N. America' respectively.

Click 'Next' to continue or click 'Back' to return to the previous step.

4.4 Setting Import Options

Commit

Commit after done - check this option to commit the transaction after import is finished.

Commit after ... records - set a number of records, after importing which the transaction shall be committed.

Import data wizard - Import from HR_EMPLOYEE.xls

Step 3 of 3

Commit Options | Advanced

Commit

Commit after each record(s)

☐ Commit when done

Record count

☒ Import all records

Import only first record(s)

Error log

☒ Enable error log

Error Log File Name

☒ Rewrite error log file if it exists

☐ Show error log after import

☐ Close wizard after import

Save template to file...

Help < Back Next > Cancel Execute

Record count

Import all records - check this option to import all records from the source table.

Import only ... first record(s) - if you don't want all the records to be imported, set a number of records to import them from the source file. In this case only this number of records (beginning from the first one) will be imported.

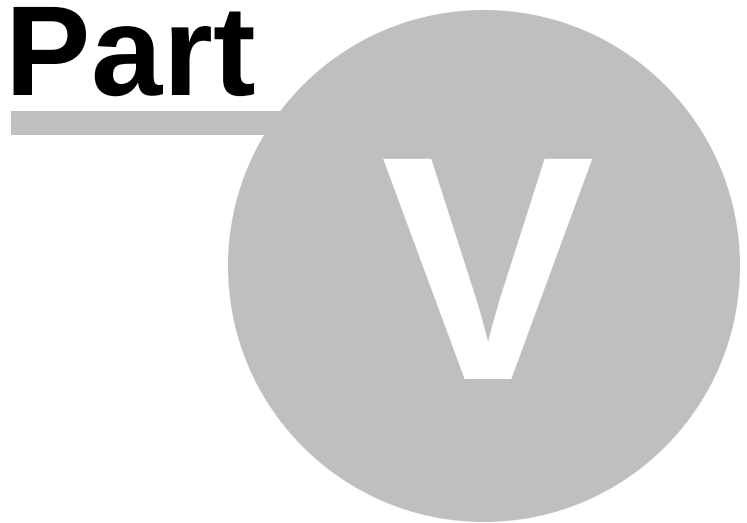
Miscellaneous

Add Type - select the method of adding data to the dataset: Append or Insert.

Save template to file - use this button to save current import options (source filename, field correspondence, format options, etc) to file to fasten the process of configuring your next import.

Click 'Execute' when you are done to start import, click 'Back' to return to any step of preparing import or click 'Cancel' to cancel import.

Part



5 Appendix

5.1 Supported file formats

- **MS Excel 97-2003**

The most popular e-table format used by Microsoft® Excel (*.xls). The result files are fully compatible with Microsoft® Excel versions 97-2003.

- **MS Access**

File of Microsoft® Access format (*.mdb, *.accdb) with an ADO connection used.

- **MS Word**

One of the most popular text processing formats used by Microsoft® Word (*.docx). The result files are fully compatible with Microsoft® Word.

- **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).

• **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.

Credits

Software Developers:

Alexey Butalov

Alex Paclin

Alexey Saybel

Dmitry Ziborov

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