

# SMImport suite

[Registration](#)

---

**Scalabium**

Scalable solutions that grow with you...

The SMImport suite for Delphi/C++Builder allows to import a data into linked recordset (any TDataset descendents) from:

1. MS Excel spreadsheet (directly without OLE/DDE, including Office 12 files)
2. text delimited file
3. text fixed width file
4. XML file
5. HTML file
6. MS Access database
7. MS Word document
8. Lotus 1-2-3 spreadsheet
9. QuattroPro spreadsheet
10. Paradox table (directly without BDE)
11. DBase/FoxPro table (directly without BDE)
12. any ADO connection
13. Advantage database
14. DBISAM table
15. Clarion tables
16. Windows Clipboard
17. Windows Address Book (WAB)
18. any BDE dataset
19. VCalendar/ICalendar
20. Open Office spreadsheets
21. MS Outlook
22. any dataset component

Also SMImport suite contain the compound component with visual dialogs for easy defining of import process settings. With this visual components you can allows for your end-user to setup an import parameters.

Visit the web site at <http://www.scalabium.com> to see if updated help files is available.

# Registration

[By credit card](#) [By mail](#)

---

## Why Register

Thank you for your interest in SMImport suite.

Registered users will receive the latest registered version of SMImport suite, free on-line support, and the source code.

## Online registration

You can order a product online at:

1. ShareIt: <https://secure.element5.com/register.html?productid=137994>
2. Avangate: <https://secure.avangate.com/order/checkout.php?PRODS=4534078>
3. PayPro: <https://secure.payproglobal.com/orderpage.aspx?products=51385>

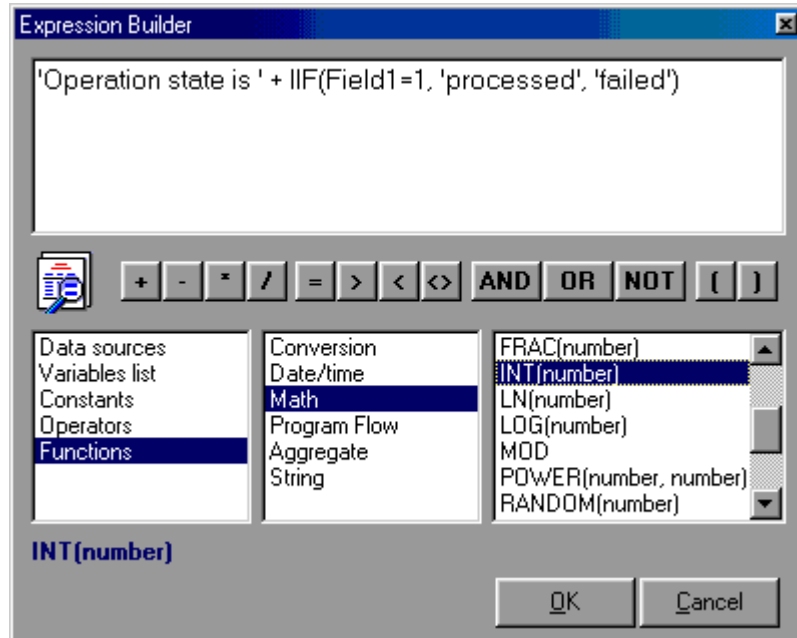
## Questions & Comments

Please refer questions or comments about SMImport suite to:

<http://www.scalabium.com>  
<mailto:mshkolnik@scalabium.com>

# Expressions in Mappings

An internal parser allow to use extended functions and operations in Mappings definitions.



# TSMIDataFormats object

[Properties](#) [Example](#)

---

## Unit

[SMIBase](#)

## Description

The TSMIDataFormats object provides the possibility to customize the formats of loaded data.

You would normally never create an object of this class. This object is created by any import component automatically.

# BooleanFalse property

[See also](#)

---

## Applies to

[TSMIDataFormats](#) object

## Declaration

```
property BooleanFalse: string;
```

## Description

Using this property you can customize a text value of "false".  
By default is a "False" text string.

# BooleanTrue property

[See also](#)

---

## Applies to

[TSMIDataFormats](#) object

## Declaration

```
property BooleanTrue: string;
```

## Description

Using this property you can customize a text value of "true".  
By default is a "True" text string.

# DateOrder property

[See also](#)

---

## Applies to

[TSMIDataFormats](#) object

## Declaration

**property** DateOrder: TSMIDateOrder;

## Description

Using this property you can customize the date order for any date/time values.  
By default is a doMDY.

# DateSeparator property

[See also](#)

---

## Applies to

[TSMIDataFormats](#) object

## Declaration

**property** DateSeparator: Char;

## Description

Using this property you can customize the DateSeparator character for any date/time values.

By default is a current value from Regional settings of Windows.



# DecimalSeparator property

[See also](#)

---

## Applies to

[TSMIDataFormats](#) object

## Declaration

**property** DecimalSeparator: Char;

## Description

Using this property you can customize the DecimalSeparator character for any numeric values.

By default is a current value from Regional settings of Windows.

# FourDigitYear property

[See also](#)

---

## Applies to

[TSMIDataFormats](#) object

## Declaration

**property** FourDigitYear: Boolean;

## Description

Using this property you can define a Y2K support for any date/time values.

By default is a current value from Regional settings of Windows (true if the ShortDateFormat contains 'yyyy').

# LeadingZerosInDate property

[See also](#)

---

## Applies to

[TSMIDataFormats](#) object

## Declaration

**property** LeadingZerosInDate: Boolean;

## Description

Using this property you can switch between masks without zeros in day/month/year and mask with zeros. For example, 'm/d/yyyy' and 'mm/dd/yyyy'.

By default is a current value from Regional settings of Windows (true if the ShortDateFormat contains 'dd').

# TimeSeparator property

[See also](#)

---

## Applies to

[TSMIDataFormats](#) object

## Declaration

**property** TimeSeparator: Char;

## Description

Using this property you can customize the TimeSeparator character for any date/time values.

By default is a current value from Regional settings of Windows.

# TSMImportBaseComponent component

See also [Properties](#) [Methods](#) [Events](#)

---

## Unit

[SMIBase](#)

## Description

This is a base type from which inherited any import component.

# About property

[See also](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** About: TSMIAbout;

## Description

This is design-time property only. If you press an ellipses button in Object Inspector, you'll see a dialog with short information about SMImport suite and authors.

# AnimatedStatus property

[See also](#)

---

## Applies to

[TSMImportBaseComponent](#) component

AnimatedStatus indicates whether the status dialog is displaying during import process.

## Declaration

**property** AnimatedStatus: Boolean;

## Description

Set AnimatedStatus to True if you want to show an animated status dialog during the import process.

Set AnimatedStatus to False if you want to import a data without animated status dialog.

# DataFormats property

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** DataFormats: TSMIDataFormats;

## Description

The DataFormats property provides the possibility to customize the formats of loaded data.



# DataSet property

[See also](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

```
property DataSet: TDataSet;
```

## Description

Use the Dataset property to specify a source for data importing.

You can specify the any TDataSet successor.

For example, the BDE's TTable, TQuery, the multi-tier TClientDataSet, ADO's TADODataset, TADOTable, TADOQuery, IBX's TIBTable, TIBQuery or third-party dataset components (Titan for Btrieve, MemoryTable, OracleDataset etc).

# DatasetKeys property

[See also](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** DatasetKeys: **string**;

## Description

If you want to update a data into dataset with data from external source, you must define key fields. Using this information import process may define that one record from external source is "similar" to record in destination.

If you have a multi-field key, please enter it in the next format 'Field1;Field2;Field3;...'

If you have unique index, then you can use the index fields as key fields for importing.

PS:

if you'll not define a DatasetKeys property, then you couldn't import a data using imUpdate, imAppendUpdate or imDelete as value for ImportMode property. In this situation you can use the imAppend or imCopy modes only which not requires information about keys.

# FieldDelimiter property

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** FieldDelimiter: TSMIFieldDelimiter;

## Description

You can select one from pre-defined field delimiters:

1. fdNone - your external text file haven't any delimiter between fields
2. fdCustom - you have a custom delimiter which defined in FieldDelimiterCustom
3. fdTab - the delimiter is tabular character
4. fdSemicolon - the delimiter is semicolon (;) character
5. fdComma - the delimiter is comma (,) character
6. fdSpace - the delimiter is space (#32) character

For example, for the next text row in source text file

"Ukraine";"Kiev";"East Europe";603700;52000000

you must define a FieldDelimiter property as fdSemicolon

# FieldDelimiterCustom property

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** FieldDelimiterCustom: Char;

## Description

If you have a some unique field delimiter which are not listed in TSMIFieldDelimiter (tabular, semicolon, comma or space), you can assign it to this property.

If you will assign a some pre-defined delimiter, will automatically changed the FieldDelimiter property.

# Fixed property

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

```
property Fixed: Boolean;
```

## Description

This property allows to switch a mode of import type for text file.

When Fixed is True, the source text file is "Fixed Width": fields are aligned in columns with delimiters between each field.

When Fixed is False, the source text is CSV: characters such as comma or tab (separator symbols) separate each field.

# Mappings property

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

Mappings specifies the column mappings for an import operation.

## Declaration

**property** Mappings: TStrings;

## Description

Set Mappings to specify the correspondence between fields in the Source and fields in the Destination. By default the import components matches fields based on their position in the source and destination tables. That is, the first column in the source is matched with the first column in the destination, and so on. Mappings enables an application to override this default.

Mappings is a list of column mappings (one per line) in one of two forms. To map the column ColName in the source table to the column of the same name in the destination table, use:

ColName

To map the column named SourceColName in the source table to the column named DestColName in the destination table, use:

DestColName = SourceColName

When adding or appending records, fields in Destination which have no entry in Mappings will be set to NULL.

When copying a dataset, fields in Destination which have no entry in Mappings will not appear as columns in the copy of the table.

If source and destination column data types are not the same, import component can either cancel the loading operation or perform a "best fit".

If possible, the field values from the Source will be converted into type of fields in Destination but you can control this process in OnGetCellParams event.

# Mode property

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

```
property Mode: TImportMode;
```

## Description

You can change an import mode which will applied to data loading.  
The description of values view at TImportMode topic.

Important note:

if you use the imUpdate, imAppendUpdate or imDelete mode, don't forget to define a DatasetKeys property.

# Options property

[Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** Options: TSMIOptions;

## Description

This is options which will be applied to import process. For example, you can define a "silence" mode without any interaction with user (no warnings/errors to user).



# RecordSeparator property

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** RecordSeparator: TSMIRecordSeparator;

## Description

You can select one from pre-defined record separators:

1. rsCustom - you have a custom record separator which is defined in RecordSeparatorCustom property
2. rsCRLF - the separator is a DOS/Windows standard separator of text files CR+LF (#13#10)
3. rsCR - the separator is a standard separator of text files (#13)
4. rsLF - the separator is a standard UNIX separator of text files (#10)

# RecordSeparatorCustom property

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** RecordSeparatorCustom: **string**;

## Description

If you want to use a custom separator between rows in text file and this separator is not included in list of pre-defined separators, you must assign it to value of this property.

For example, RecordSeparatorCustom := #11#11. In this case automatically will be changed a RecordSeparator property to rsCustom value.

# RowFirst property

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** RowFirst: Integer;

## Description

This value is a number of record. Starting from this record the next records will be imported. The any records which have a number less than RowFirst will be skipped and will not loaded in destination dataset.

# RowLast property

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** RowLast: Integer;

## Description

This value is a number of record. The any record which have a number less than RowLast will be imported in destination dataset.

The any records which have a number greater than RowLast will be skipped and will not loaded.

# SourceFileName property

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

```
property SourceFileName: string;
```

## Description

This property is a file name with external data which must be imported.

For example, 'C:\INBOX\orders.txt' for text format, 'C:\BACKUPS\products.db' for Paradox table or 'F:\SHARED\invoices.xls' for Excel spreadsheet.

# TextQualifier property

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** TextQualifier: TSMITextQualifier;

## Description

You can select one from pre-defined text qualifiers for text fields:

1. tqNone - your qualifier is an empty (disabled)
2. tqCustom - you have a custom text qualifier which is defined in TextQualifierCustom property
3. tqQuot - you have a standard qualifier as quote ("")
4. tqApos - you have a standard qualifier as apos ('')

When TextQualifier (or TextQualifierCustom) defines a some value as qualifier, then import engine wait each string value as TextQualifierCustom+TextValue+TextQualifierCustom.

For example, for the next text row in source text file

"Ukraine";"Kiev";"East Europe";603700;52000000

you must define a TextQualifier property as tqQuot

# TextQualifierCustom property

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** TextQualifierCustom: Char;

## Description

If you want to use a custom text qualifier (a first and last char in any string data) for text file and this qualifier is not included in list of pre-defined qualifiers, you must assign it to value of this property.

For example, TextQualifierCustom := '-'. In this case automatically will be changed a TextQualifier property to tqCustom value.

# TitleStatus property

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

```
property TitleStatus: string;
```

## Description

If you want, here you can define a custom caption for animated status dialog.

If you'll not define it, then will be used a default "Importing..." text (depends from translated multilingual resources).



# LoadSpecification method

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

```
class procedure LoadSpecification(strFileName: string);
```

## Description

This method allow to load a some prepared specification with all pre-defined settings for importing.

For example, this specification can be created in SaveSpecification method or saved in wizard component.

Also this specification file can be prepared during export process.

# Extension method

[See also](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

```
function Extension: string; virtual;
```

## Description

This virtual method allow to receive a default extension of file. For example, the 'XLS' for MS Excel spreadsheets or 'XML' for XML-file with data.

# FillFileFilters method

[See also](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

```
function FillFileFilters: string; virtual;
```

## Description

This method returns a list of strings and file extensions which are supported by import component. For example, the TSMImportFromText component support the next file filters:

1. Text files (\*.txt)
2. Comma-delimited file (\*.csv)
3. Data file (\*.dat)
4. Text file (\*.prn)
5. Tabular file (\*.tab)
6. ASCII file (\*.asc)

This method will be executed before any open/save dialog opening when will be generated a values for filters.

# AboutSMI method

[See also](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**procedure** AboutSMI;

## Description

To call this method if you want to see a dialog with short information about SMIImport suite and authors. Also here you can check a version of used SMIImport version.

# Execute method

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

Execute method performs the import operation specified by other properties.

## Declaration

```
procedure Execute;
```

## Description

When you finished to set the properties with import settings, call Execute to perform the operation. As a minimum, the Dataset, SourceFileName and ImportMode properties must be defined.

After calling Execute, the data from external source file will be loaded in dataset.

# SaveSpecification method

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

```
procedure SaveSpecification(SpecName, FileName: string; ShowDialog:  
Boolean);
```

## Description

This method allow to save the current settings of import component into some file with specified name (it's an user friendly caption which he/she which will view in list of specifications).

If an external file with the same name already exists, it is deleted and a new file with specification is created in its place.

# OnAfterExecute event

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** OnAfterExecute: TNotifyEvent;

## Description

This event will be called after each import process so using it you can activate some own actions which must be executed after import.

The Sender parameter is an import component which finished a data loading.

# OnAfterRecordEvent event

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** OnAfterRecordEvent: TAfterRecordEvent;

## Description

This event allow to control an import process - this fires after each imported rows (no matter - successful or not).

Using this event you can cancel an import process in any moment or log any action.



# OnBeforeExecute event

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** OnBeforeExecute: TNotifyEvent;

## Description

This event will be called before each import process activation. So here you can activate some own actions which must be executed before import.

The Sender parameter is an import component which will start a data loading.

# OnBeforeRecordEvent event

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** OnBeforeRecordEvent: TBeforeRecordEvent;

## Description

This event allow to define global values for each parsed value before applying to fields. Also here you can skip some row from loading (just by your custom condition).

The Fields parameter is a list of parsed field names.

The Values parameter is a variant array with parsed value for each field name.

To skip a row from loading just assign a False value to Accept parameter. By default the Accept is True.

# OnErrorEvent event

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** OnErrorEvent: TOnErrorEvent;

## Description

In this event you can control any error during importing.

The Sender is a current import component where was raised the exception.  
The Error is standard exception type.

If you want to stop a process, you must return True in Abort parameter.  
But if you want to ignore this error and continue the import, just return a False.

In this event you can handle the any errors and, for example, to save the "bad" values in some buffer for next modification and re-loading.

# OnGetCellParams event

[See also Example](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

**property** OnGetCellParams: TGetCellParamsEvent;

## Description

In this event you can control import process in own hands. The event will be called for each imported value (each field processing for any records).

If you want to change a result value which will be imported in destination dataset, you must change a Value parameter.

The Field parameter will show a field of destination dataset where will be placed a value.

The Sender is an import component which was used for data loading and parsing.

# OnCreateStructure event

[See also Example](#)

---

## Unit

[SMIBase](#)

## Declaration

**property** OnCreateStructure: TSMIOnCreateStructure;

## Description

If you want to create a dataset with structure from parsed external file and only after that to load data, then in this event you can do it.

Here you will receive a collection of parsed columns with all required information - field name, width, data type etc

This event is called before real import process will be started.

# TSpreadSheetCells component

[See also](#) [Properties](#) [Methods](#)

---

## Unit

[SMCells](#)

## Description

This class is an analogue of virtual array - the unlimited list with access through columns like array.

The SMIImport suite use this internal class for support of "spreadsheet" files - possibility to read/load a spreadsheet into virtual array.

# ColCount property

[See also](#)

---

## Applies to

[TSpreadSheetCells](#) component

## Declaration

**property** ColCount: Word;

## Description

Return a number of columns in virtual array of spreadsheet.

# RowCount property

[See also](#)

---

## Applies to

[TSpreadSheetCells](#) component

## Declaration

**property** RowCount: Word;

## Description

Return a number of rows in virtual array of spreadsheet.



# AddRow method

[See also](#)

---

## Applies to

[TSpreadSheetCells](#) component

## Declaration

```
function AddRow(i: Integer): Integer;
```

## Description

This method allow to add a row with some specific index into virtual array.

# GetRow method

[See also](#)

---

## Applies to

[TSpreadSheetCells](#) component

## Declaration

```
function GetRow(Row: Integer; NeedAdd: Boolean): Integer;
```

## Description

This method allow to read a row from virtual array.

If NeedAdd parameter is True and row is not exist in virtual array, then row will be added and returned as result.

# GetValue method

[See also](#)

---

## Applies to

[TSpreadSheetCells](#) component

## Declaration

```
function GetValue(Col, Row: Integer): Variant;
```

## Description

This method allow to read a value from virtual array by specific row and specific column number.

# RemoveRow method

[See also](#)

---

## Applies to

[TSpreadSheetCells](#) component

## Declaration

```
procedure RemoveRow(i: Integer);
```

## Description

This method allow to remove a row from virtual array.

# SetValue method

[See also](#)

---

## Applies to

[TSpreadSheetCells](#) component

## Declaration

```
procedure SetValue(Col, Row: Integer; Value: Variant);
```

## Description

This method allow to write a value into virtual array by specific row and specific column number.

# TCustomSpreadSheet component

[See also Properties](#)

---

## Unit

[SMCells](#)

## Description

This is an internal class for support of "spreadsheet" files - possibility to read/load a spreadsheet into virtual array using events.

The components for MS Excel, Lotus 1-2-3, QuattroPro spreadsheets are inherited from this basic type.

# FileName property

---

## Applies to

[TCustomSpreadSheet](#) component

## Declaration

```
property FileName: string;
```

## Description

Defines a file name from which can be loaded a spreadsheet.

# OnCellValue property

---

## Applies to

[TCustomSpreadSheet](#) component

## Declaration

```
property OnCellValue: TOnCellValue;
```

## Description

Using this event you can load a value to any cell of virtual array.



# OnColumnWidth property

---

## Applies to

[TCustomSpreadSheet](#) component

## Declaration

**property** OnColumnWidth: TOnColumnWidth;

## Description

Using this event you can change a width of any column from virtual array.

# OnDimensions property

---

## Applies to

[TCustomSpreadSheet](#) component

## Declaration

```
property OnDimensions: TOnDimensions;
```

## Description

Using this event you can define the row and column numbers in spreadsheet.

# OnRowHeight property

---

## Applies to

[TCustomSpreadSheet](#) component

## Declaration

```
property OnRowHeight: TOnRowHeight;
```

## Description

Using this event you can define a height of any row from virtual array.

# TSpreadSheet component

[See also](#) [Methods](#)

---

## Unit

[SMCells](#)

## Description

This is an internal class for support of "spreadsheet" files - possibility to read/load a spreadsheet into virtual array from external file.

The components for MS Excel, Lotus 1-2-3, QuattroPro spreadsheets are inherited from this basic type.

# LoadFromFile method

---

## Applies to

[TSpreadSheet](#) component

## Declaration

```
procedure LoadFromFile(const FileName: string); virtual;
```

## Description

This is a virtual method which allow to realize a file reading and filling of virtual array using events.

# TSMImportFromBDE component

[See also](#) [Properties](#) [Methods](#) [Events](#)

---

## Unit

[SMI2BDE](#)

## Description

This component allows to load a data from external Paradox or DBase table.

You must understand that this component uses a BDE so if you'll use it, then on client computers must be installed BDE.

# TableType property

---

## Applies to

[TSMImportFromBDE](#) component

## Declaration

```
property TableType: TTableType;
```

## Description

When you use a BDE loading you must define which table format you want to use: Paradox or DBase.



# TSMImportFromDataSet component

[See also](#) [Properties](#) [Events](#)

---

## Unit

[SMI2DS](#)

## Description

This component allows to load a data from one dataset to other.



# SourceDataset property

[See also](#)

---

## Applies to

[TSMImportFromDataSet](#) component

## Declaration

**property** SourceDataset: TDataSet;

## Description

Using this property you can define a dataset which will "deploy" a data for loading. The records from this dataset will be imported into dataset which defined in Dataset property.

# TMSExcel component

[Properties](#) [Methods](#)

---

## Unit

[SMXLS](#)

## Description

This basic component allow to read the MS Excel's spreadsheet into virtual array (unlimited list of cells). The reading is a direct and doesn't use any external libraries, OLE, DDE etc

This component can be used for MS Excel loading into TStringGrid component.

# SheetIndex property

---

## Applies to

[TMSExcel](#) component

## Declaration

```
property SheetIndex: Integer;
```

## Description

This property is readonly only and is used for access to data in specific sheet of spreadsheet file.

# Version property

[See also](#)

---

## Applies to

[TMSExcel](#) component

## Declaration

```
property Version: TExcelVersion;
```

## Description

This property is readonly. You can check a version of MS Excel's spreadsheet that was loaded from file.

Keeps a version of last loaded spreadsheet.

# TQuattroPro component

[Properties](#) [Methods](#)

---

## Unit

[SMWQ](#)

## Description

This basic component allow to read the QuattroPro's spreadsheet into virtual array (unlimited list of cells). The reading is a direct and doesn't use any external libraries, OLE, DDE etc

This component can be used for QuattroPro loading into TStringGrid component.

# Version property

[See also](#)

---

## Applies to

[TQuattroPro](#) component

## Declaration

**property** Version: TWQVersion;

## Description

This property is readonly. You can check a version of QuattroPro's spreadsheet that was loaded from file.

Keeps a version of last loaded spreadsheet.

# TLotus123 component

[Properties](#) [Methods](#)

---

## Unit

[SMWKS](#)

## Description

This basic component allow to read the Lotus 1-2-3's spreadsheet into virtual array (unlimited list of cells). The reading is a direct and doesn't use any external libraries, OLE, DDE etc

This component can be used for Lotus 1-2-3 loading into TStringGrid component.

# Version property

[See also](#)

---

## Applies to

[TLotus123](#) component

## Declaration

```
property Version: TWKSVersion;
```

## Description

This property is readonly. You can check a version of Lotus 1-2-3's spreadsheet that was loaded from file.

Keeps a version of last loaded spreadsheet.



# TSMImportFromCell component

[Properties](#) [Methods](#) [Events](#)

---

## Unit

[SMI2Cell](#)

## Description

This component is declared as basic for loading spreadsheets (MS Excel, QuattroPro, Lotus 1-2-3 etc) into dataset.



# TSMImportFromHTML component

[See also](#) [Properties](#) [Methods](#) [Events](#)

---

## Unit

[SMI2HTML](#)

## Description

This component allows to load a data from external HTML file.

The import process does not use the MS Internet Explorer application for data transformation. The TSMImportFromHTML component will read a first table from HTML file directly.

Supported the HTML files which are generate by SMExport suite.



# TSMImportFromText component

[See also](#) [Properties](#) [Methods](#) [Events](#)

---

## Unit

[SMI2TXT](#)

## Description

This component allows to load a data from external text file.

Supported a lot of custom settings like file format (fixed or CSV), record separator, field delimiter, text qualifier etc

Also automatically supported the field mapping in case when first row in text file contains field names.



# TSMImportFromWKS component

[See also](#) [Methods](#) [Events](#)

---

## Unit

[SMI2WKS](#)

## Description

This component allows to load a data from external Lotus 1-2-3 spreadsheet.

The import process does not use the Lotus 1-2-3 application for data reading so on client computers the Lotus 1-2-3 is not necessary. The TSMImportFromWKS component will read a spreadsheet directly.

Supported any Lotus 1-2-3 versions: 1.0, 2.0 and WR1.



# TSMImportFromQuattro component

[See also](#) [Methods](#) [Events](#)

---

## Unit

[SMI2WQ](#)

## Description

This component allows to load a data from external QuattroPro spreadsheet.

The import process does not use the Corel QuattroPro application for data reading so on client computers the QuattroPro is not necessary. The TSMImportFromQuattro component will read a spreadsheet directly.

Supported any QuattroPro versions: 1.0 and 2.0.



# TSMImportFromXLS component

[See also](#) [Methods](#) [Events](#)

---

## Unit

[SMI2XLS](#)

## Description

This component allows to load a data from external MS Excel spreadsheet.

The import process does not use the MS Excel application for data reading so on client computers the MS Excel is not necessary. The TSMImportFromXLS component will read a spreadsheet directly.

Supported any MS Excel versions: 2.0, 3.0, 4.0, 5.0, 7.0 and 8.0.



# TSMImportFromXML component

[See also](#) [Properties](#) [Methods](#) [Events](#)

---

## Unit

[SMI2XML](#)

## Description

This component allows to load a data from external XML file.

The import process does not use the MS Internet Explorer application for data transformation. The TSMImportFromXML component will read a XML file directly.

Supported the XML files which are generate by TClientDataset component (see SaveToFile method) and SMExport suite as well.

# TSMIUserAccess object

[See also](#) [Properties](#) [Methods](#)

---

## Unit

[SMIWiz](#)

## Description

This type declared for support of restrictions in wizard component. Using properties of this type you can control end-user's control to any parameter of import.

For example, you can disable a changing of type for source file and set a field mappings as read only.



# FieldAdjustment property

[See also](#)

---

## Applies to

[TSMIUserAccess](#) object

## Declaration

**property** FieldAdjustment: TSMIRestriction;

## Description

This property declared for restrictions which you can place for your end-users in wizard dialog.

1. irDisabled mean that user can't access to field adjustments. For example, he/she can't define the breaks between fields in text file
2. irReadOnly mean that user can view field adjustments but can't change
3. irReadWrite mean that user have a full access to field adjustments without any restrictions

# FieldDelimiter property

[See also](#)

---

## Applies to

[TSMIUserAccess](#) object

## Declaration

**property** FieldDelimiter: TSMIRestriction;

## Description

This property declared for restrictions which you can place for your end-users in wizard dialog.

1. irDisabled mean that user can't access to a field delimiter setting
2. irReadOnly mean that user can view a current setting for a field delimiter but can't change
3. irReadWrite mean that user have a full access to a field delimiter without any restrictions

# FieldMapping property

[See also](#)

---

## Applies to

[TSMIUserAccess](#) object

## Declaration

**property** FieldMapping: TSMIRestriction;

## Description

This property declared for restrictions which you can place for your end-users in wizard dialog.

1. irDisabled mean that user can't access to value of grid with mappings between fields in destination and source data
2. irReadOnly mean that user can view a current field mappings but can't change
3. irReadWrite mean that user have a full access to grid with field mappings without any restrictions

# FirstRow property

[See also](#)

---

## Applies to

[TSMIUserAccess](#) object

## Declaration

**property** FirstRow: TSMIRestriction;

## Description

This property declared for restrictions which you can place for your end-users in wizard dialog.

1. irDisabled mean that user can't access to value of edit box where can be changed a first row value
2. irReadOnly mean that user can view edit box with first row value but can't change
3. irReadWrite mean that user have a full access to edit box with first row value without any restrictions

# ImportMode property

[See also](#)

---

## Applies to

[TSMIUserAccess](#) object

## Declaration

**property** ImportMode: TSMIRestriction;

## Description

This property declared for restrictions which you can place for your end-users in wizard dialog.

1. irDisabled mean that user can't access to value of radio box where can be changed an import mode
2. irReadOnly mean that user can view group box with import modes but can't change
3. irReadWrite mean that user have a full access to group box with import modes without any restrictions

# LastRow property

[See also](#)

---

## Applies to

[TSMIUserAccess](#) object

## Declaration

**property** LastRow: TSMIRestriction;

## Description

This property declared for restrictions which you can place for your end-users in wizard dialog.

1. irDisabled mean that user can't access to value of edit box where can be changed a last row value
2. irReadOnly mean that user can view edit box with last row value but can't change
3. irReadWrite mean that user have a full access to edit box with last row value without any restrictions

# PreviewData property

[See also](#)

---

## Applies to

[TSMIUserAccess](#) object

## Declaration

**property** PreviewData: TSMIRestriction;

## Description

This property declared for restrictions which you can place for your end-users in wizard dialog.

1. irDisabled mean that user can't access to a mode of preview grid with example of loaded data
2. irReadOnly mean that user can view a current setting for a mode of preview grid with example of loaded data but can't change
3. irReadWrite mean that user have a full access to a mode of preview grid with example of loaded data without any restrictions

# RecordSeparator property

[See also](#)

---

## Applies to

[TSMIUserAccess](#) object

## Declaration

**property** RecordSeparator: TSMIRestriction;

## Description

This property declared for restrictions which you can place for your end-users in wizard dialog.

1. irDisabled mean that user can't access to record separator setting
2. irReadOnly mean that user can view a current setting for record separator but can't change
3. irReadWrite mean that user have a full access to record separator without any restrictions



# SourceFileName property

[See also](#)

---

## Applies to

[TSMIUserAccess](#) object

## Declaration

**property** SourceFileName: TSMIRestriction;

## Description

This property declared for restrictions which you can place for your end-users in wizard dialog.

1. irDisabled mean that user can't access to a name of loaded external file
2. irReadOnly mean that user can view a current setting for a name of loaded external file but can't change
3. irReadWrite mean that user have a full access to a name of loaded external file without any restrictions

# Specification property

[See also](#)

---

## Applies to

[TSMIUserAccess](#) object

## Declaration

**property** Specification: Boolean;

## Description

This property declared for restrictions which you can place for your end-users in wizard dialog.

If you will set Specification as True then in wizard dialog will be available mode where end-user can load/save/delete the specifications.  
Else end-user couldn't access to this mode.

# TableType property

[See also](#)

---

## Applies to

[TSMIUserAccess](#) object

## Declaration

**property** TableType: TSMIRestriction;

## Description

This property declared for restrictions which you can place for your end-users in wizard dialog.

1. irDisabled mean that user can't access to a format of loaded external file
2. irReadOnly mean that user can view a current setting for format of loaded external file but can't change
3. irReadWrite mean that user have a full access to format of loaded external file without any restrictions

# TextQualifier property

[See also](#)

---

## Applies to

[TSMIUserAccess](#) object

## Declaration

**property** TextQualifier: TSMIRestriction;

## Description

This property declared for restrictions which you can place for your end-users in wizard dialog.

1. irDisabled mean that user can't access to a text qualifier
2. irReadOnly mean that user can view a current setting for a text qualifier but can't change
3. irReadWrite mean that user have a full access to a text qualifier without any restrictions

# TextType property

[See also](#)

---

## Applies to

[TSMIUserAccess](#) object

## Declaration

**property** TextType: TSMIRestriction;

## Description

This property declared for restrictions which you can place for your end-users in wizard dialog.

1. irDisabled mean that user can't access to a type of text file
2. irReadOnly mean that user can view a current setting for type of a text file but can't change
3. irReadWrite mean that user have a full access to a type of text file without any restrictions

# Create method

---

## Applies to

[TSMIUserAccess](#) object

Creates and instance of a TSMIUserAccess object.

## Declaration

**constructor** Create(AOwner: TComponent); **virtual**;

## Description

You would normally never create an object of this class. This object is created by and bound to a TSMIWizardDlg component.



# TSMIWizardDlg component

[See also](#) [Properties](#) [Methods](#) [Events](#)

---

## Unit

[SMIWiz](#)

## Description

The TSMIWizardDlg is a compound component which allow to load a data from any supported external files.

The useful wizard form with friendly interface and step-by-step instructions helps to solve any task.

Also component can be used as compound importing without any visualization and interaction with end-user.

If you will give a wizard to your end-user, you can still control a process in own hands using strong restrictions in access to settings.

# Picture property

[See also Example](#)

---

## Applies to

[TSMIWizardDlg](#) component

## Declaration

**property** Picture: TPicture;

## Description

You can assign own logo which will be displayed in wizard component. If you'll not assign a picture, will be displayed a default picture.



# TableType property

[See also](#)

---

## Applies to

[TSMIWizardDlg](#) component

## Declaration

**property** TableType: TTableTypeImport;

## Description

You can define a default format of external file for wizard component. According to user's permissions, you can enable/disable a format changing.

IMPORTANT:

If you'll set a teParadox or teDBase as default but on client computer the BDE is not installed, then your default TableType will be changed on first from available formats.

# Title property

[See also Example](#)

---

## Applies to

[TSMIWizardDlg](#) component

## Declaration

```
property Title: TCaption;
```

## Description

If you want, here you can define a custom caption for wizard dialog.

If you'll not define it, then will be used a default "Import Wizard" text (depends from translated multilingual resources).

# UseDisplayNames property

---

## Applies to

[TSMIWizardDlg](#) component

## Declaration

**property** UseDisplayNames: Boolean;

## Description

Using this property you can define to show your customized field names for displaying to user.

If the UseDisplayNames is True, then in grid where user can define a field mappings, he/she will see a DisplayName property of each field of your dataset. Else will be displayed the FieldName property for fields.

# UserAccess property

[See also](#)

---

## Applies to

[TSMIWizardDlg](#) component

## Declaration

**property** UserAccess: TSMIUserAccess;

## Description

This property allow to restrict an end-user access to properties in wizard component. Using properties of this type you can control end-user's control to any parameter of import.

For example, you can disable a changing of type for source file and set a field mappings as read only.

# Execute method

[See also Example](#)

---

## Applies to

[TSMIWizardDlg](#) component

## Declaration

```
function Execute: TModalResult;
```

## Description

This method allow to execute the importing with wizard displaying.

After running will be displayed a visual dialog where user can define any import property step-by-step in wizard mode. After definition of any properties end-user can activate the data loading.

You can control any step in own hands: to restrict any property, define the default values etc

# ExecuteWithoutDialog method

[See also Example](#)

---

## Applies to

[TSMIWizardDlg](#) component

## Declaration

```
procedure ExecuteWithoutDialog;
```

## Description

This method allow to execute the importing without wizard displaying.

It's useful when you want to work with compound import component (and not have a lot of components for each import type) but you want to control all process in own hands without interaction with end-user.

Just set the properties of importing and call this method for data loading.

# OnGetSpecifications event

[See also Example](#)

---

## Applies to

[TSMIWizardDlg](#) component

## Declaration

**property** OnGetSpecifications: TSMIGetSpecificationsEvent;

## Description

If you want to customize a list of specifications which will use the end-user in wizard component, then in this event you can do it.

Here you can load own list of specifications instead default. By default (without code in this event) will be loaded smi-files from application folder.

As result you must fill the `IstFiles` list by the next rule:

the name of specification which will be displayed in list box is item of list. The file name with this specification must be loaded as Object to this item.

# Registration By credit card

**Visa/Discover/MasterCard/AmericanExpress 50 EUR for full package with sources**

---

**For technical support or comments about this program, you may contact Mike Shkolnik at:** <mailto:mshkolnik@scalabium.com>

For your convenience we have contracted another companies (registrators), RegSoft.Com, USA and ShareIT, Germany to process any orders you may wish to place with your Visa, Discover, MasterCard or other credit cards.

Registrators can be easily contacted **for orders only** via any of the following methods:

## ONLINE ORDERS

You may register SMImport suite via an online order form by pointing your browser to:

1. ShareIt: <https://secure.element5.com/register.html?productid=137994>
2. Avangate: <https://secure.avangate.com/order/checkout.php?PRODS=4534078>
3. PayPro: <https://secure.payproglobal.com/orderpage.aspx?products=51385>

## PHONED ORDERS

ShareIt: +49-221-2407279

(US and Canada customers please call 1-800-903-4152)

## FAXED ORDERS

Available 24 hours. International & business orders are encouraged.

ShareIt: 724-850-8187 (US customers)

(customers outside the US: +49-221-2407278)

## US CHECKS AND CASH ORDERS

ShareIt:

European office: US office:

element 5 AG ShareIt! Inc.

Vogelsanger Strasse 78 PO Box 844

Greensburg, PA 15601-0844

D-50823 Koln USA

Germany

tel. +49-221-2407279 tel. 724-850-8186

fax. +49-221-2407278 fax. 724-850-8187

## EMAILED ORDERS

ShareIt: [register@shareit.com](mailto:register@shareit.com)

**Please make sure to include the program ID:**



ShareIt: 137994

**Please provide (or be prepared to provide) the following information:**

- + The program you are registering.
- + Your mailing address.
- + Your Visa, Discover, or MasterCard # and it's expiration date (if using credit card).
- + Your EMail address (so registrator can send you email confirming your order and so we can contact you easily with important follow-up information or upgrade announcements).

# Registration by mail order

Select **Print Topic...** from the **File** menu to print this form.

---

**Item:** SMImport suite for Delphi/C++Builder

**Price:** 50 EUR with sources or 35 EUR without sources

Please register my copy of SMImport suite for Delphi/C++Builder,  
I am sending a check or money order for \$\_\_\_\_\_

**Name:** \_\_\_\_\_

**Company:** \_\_\_\_\_

**Address1:** \_\_\_\_\_

**Address2:** \_\_\_\_\_

**City:** \_\_\_\_\_

**State:** \_\_\_\_\_ **Zip:** \_\_\_\_\_

**Country:** \_\_\_\_\_

**Phone:** \_\_\_\_\_ optional

**Email:** \_\_\_\_\_

Please send completed form with payment to:

Mike Shkolnik

ul.Pragskaya 4, kv.39

Kiev, 02090

Ukraine

## Properties

- ▶ Run-time only    ■ Key properties
  - [BooleanFalse](#)
  - [BooleanTrue](#)
  - [DateOrder](#)
  - [DateSeparator](#)
  - [DecimalSeparator](#)
  - [ThousandSeparator](#)
  - [FourDigitYear](#)
  - [LeadingZerosInDate](#)
  - [TimeSeparator](#)

## TSMIDataFormats - Example

```
procedure ApplyFormats(smi: TSMImportBaseComponent);  
begin  
  smi.DataFormats.BooleanFalse := 'No';  
  smi.DataFormats.BooleanFalse := 'Yes';  
  
  smi.DataFormats.DateOrder := doYMD; //yy-mm-dd  
  smi.DataFormats.DateSeparator := '-';  
  smi.DataFormats.FourDigitYear := False;  
  smi.DataFormats.LeadingZeroInDate := True;  
  
  smi.DataFormats.TimeSeparator := ':';  
  
  smi.DataFormats.DecimalSeparator := '.';  
end;
```

# SMIBase unit

---

In this unit was declared a base types and base TSMIImportBaseComponent component.

## Components

[TSMIImportBaseComponent](#)

## Objects

[TSMIDataFormats](#)

## Types

[TAfterRecordEvent](#)

[TBeforeRecordEvent](#)

[TErrorEvent](#)

[TGetCellParamsEvent](#)

[TImportFormatTypes](#)

[TImportMode](#)

[TSMIAbout](#)

[TSMIDateOrder](#)

[TSMIFieldDelimiter](#)

[TSMIOption](#)

[TSMIOptions](#)

[TSMIRecordSeparator](#)

[TSMIStatistic](#)

[TSMITextQualifier](#)

[TTableTypeImport](#)

## Routines

[AboutSMIImport](#)

## **See also**

[TSMIDataFormats](#)

[BooleanTrue](#)

[DateOrder](#)

[DateSeparator](#)

[DecimalSeparator](#)

[ThousandSeparator](#)

[FourDigitYear](#)

[LeadingZeroInDate](#)

[TimeSeparator](#)

## **See also**

[TSMIDataFormats](#)

[BooleanFalse](#)

[DateOrder](#)

[DateSeparator](#)

[DecimalSeparator](#)

[ThousandSeparator](#)

[FourDigitYear](#)

[LeadingZeroInDate](#)

[TimeSeparator](#)

## **See also**

[TSMIDateOrder](#)  
[TSMIDataFormats](#)

[BooleanFalse](#)  
[BooleanTrue](#)  
[DateSeparator](#)  
[DecimalSeparator](#)  
[ThousandSeparator](#)  
[FourDigitYear](#)  
[LeadingZeroInDate](#)  
[TimeSeparator](#)



## See also

[TSMIDataFormats](#)

[BooleanFalse](#)

[BooleanTrue](#)

[DateOrder](#)

[DecimalSeparator](#)

[ThousandSeparator](#)

[FourDigitYear](#)

[LeadingZeroInDate](#)

[TimeSeparator](#)

## See also

[TSMIDataFormats](#)

[BooleanFalse](#)

[BooleanTrue](#)

[DateOrder](#)

[DecimalSeparator](#)

[ThousandSeparator](#)

[FourDigitYear](#)

[LeadingZeroInDate](#)

[TimeSeparator](#)

## **See also**

[TSMIDataFormats](#)

[BooleanFalse](#)

[BooleanTrue](#)

[DateOrder](#)

[DateSeparator](#)

[DecimalSeparator](#)

[ThousandSeparator](#)

[LeadingZeroInDate](#)

[TimeSeparator](#)

## **See also**

[TSMIDataFormats](#)

[BooleanFalse](#)

[BooleanTrue](#)

[DateOrder](#)

[DateSeparator](#)

[DecimalSeparator](#)

[ThousandSeparator](#)

[FourDigitYear](#)

[TimeSeparator](#)

## **See also**

[TSMIDataFormats](#)

[BooleanFalse](#)

[BooleanTrue](#)

[DateOrder](#)

[DateSeparator](#)

[DecimalSeparator](#)

[ThousandSeparator](#)

[FourDigitYear](#)

[LeadingZeroInDate](#)

## **See also**

[TSMIWizardDlg](#)

[TSMImportFromText](#)

[TSMImportFromXLS](#)

[TSMImportFromWKS](#)

[TSMImportFromQuattro](#)

[TSMImportFromXML](#)

[TSMImportFromHTML](#)

[TSMImportFromBDE](#)

[TSMImportFromDataset](#)

## Properties

▶ Run-time only    ■ Key properties

- [About](#)
- [AnimatedStatus](#)
- [DataFormats](#)
- [DataSet](#)
- [DatasetKeys](#)
- ▶ ■ [FieldDelimiter](#)
- ▶ ■ [FieldDelimiterCustom](#)
- ▶ ■ [Fixed](#)
- [Mappings](#)
- [Mode](#)
- [Options](#)
- ▶ ■ [RecordSeparator](#)
- ▶ ■ [RecordSeparatorCustom](#)
- ▶ ■ [RowFirst](#)
- ▶ ■ [RowLast](#)
- ▶ ■ [SourceFileName](#)
- ▶ ■ [Statistic](#)
- ▶ ■ [TextQualifier](#)
- ▶ ■ [TextQualifierCustom](#)
- [TitleStatus](#)

## Methods

- Key methods

- [LoadSpecification](#)

Create{linkDelphi=Create\_Method}

Destroy{linkDelphi=Destroy\_Method}

- [Extension](#)

- [FillFileFilters](#)

- [AboutSMI](#)

- [Execute](#)

- [SaveSpecification](#)



## Events

- Key events
- [OnAfterExecute](#)
- [OnAfterRecordEvent](#)
- [OnBeforeExecute](#)
- [OnBeforeRecordEvent](#)
- [OnErrorEvent](#)
- [OnGetCellParams](#)
- [OnCreateStructure](#)

## See also

[AboutSMI](#)

[AboutSMImport](#)

## **See also**

[TSMIWizardDlg](#)

[TSMImportFromText](#)

[TSMImportFromXLS](#)

[TSMImportFromWKS](#)

[TSMImportFromQuattro](#)

[TSMImportFromXML](#)

[TSMImportFromHTML](#)

[TSMImportFromBDE](#)

[TSMImportFromDataset](#)

**See also**

[TSMIDataFormats](#)

## **See also**

[TSMIWizardDlg](#)

[TSMImportFromText](#)

[TSMImportFromXLS](#)

[TSMImportFromWKS](#)

[TSMImportFromQuattro](#)

[TSMImportFromXML](#)

[TSMImportFromHTML](#)

[TSMImportFromBDE](#)

[TSMImportFromDataset](#)

## See also

[Dataset](#)

[Mappings](#)

[Mode](#)

## **See also**

[TSMIFieldDelimiter](#)

[FieldDelimiterCustom](#)

[Fixed](#)

[RecordSeparator](#)

[RecordSeparatorCustom](#)

[TextQualifier](#)

[TextQualifierCustom](#)

### FieldDelimiter property example

FieldDelimiter := fdComma;



## **See also**

[FieldDelimiter](#)

[Fixed](#)

[RecordSeparator](#)

[RecordSeparatorCustom](#)

[TextQualifier](#)

[TextQualifierCustom](#)

## FieldDelimiterCustom property example

FieldDelimiterCustom := '|';

## **See also**

[FieldDelimiter](#)

[FieldDelimiterCustom](#)

[RecordSeparator](#)

[RecordSeparatorCustom](#)

[TextQualifier](#)

[TextQualifierCustom](#)

### Fixed property example

```
smi.Fixed := True;  
smi.Mappings.Clear;  
smi.Mappings.Add('Account=Text1-10');  
smi.Mappings.Add('Amount=Text12-15');
```

## **See also**

[FieldDelimiter](#)

[FieldDelimiterCustom](#)

[Fixed](#)

[RecordSeparator](#)

[RecordSeparatorCustom](#)

[TextQualifier](#)

[TextQualifierCustom](#)

[Expressions in Mappings](#)

## Mappings property example

The correct Mappings property is most important key of success data loading.

The field mappings are correspondence between fields in Dataset (destination) and fields in text file (source data). So when you dropped an import component on own form, select this component and in Object Inspector you must define a Mappings property by the next rule:

1. if you have a comma-delimited text (the each field value is delimited by some character - tabular or comma), then you must define as

AccountNo=Field1

Amount=Field2

...

Here number (after 'Field' string) is a number of field in each line of text and

AccountNo/Amount/... are fields in your table. Also don't forget to set a Fixed property in False.

2. If you have a "fixed width" text file (when each field in any row have a same width), then you must describe a mapping in the next format:

Account=Text1-10

Amount=Text12-15

...

Here the first number (after 'Text' string) is start position of your field.

The second number (after '-' character) is end position of your field.

## See also

[Dataset](#)

[DatasetKeys](#)

[TImportMode](#)

### Mode property example

```
smi.Mode := imAppendUpdate;  
smi.DatasetKeys := 'ID_CUSTOMER';
```



### Options property example

{define an interaction with end-user (status dialog, error messages etc)}  
smi.Options := smi.Options + [soShowMessage];

{define a silence mode without any interaction with end-user}  
smi.Options := smi.Options - [soShowMessage];

## **See also**

[TSMIRecordSeparator](#)

[FieldDelimiter](#)

[FieldDelimiterCustom](#)

[Fixed](#)

[RecordSeparatorCustom](#)

[TextQualifier](#)

[TextQualifierCustom](#)

### **RecordSeparator property example**

RecordSeparator := rsCRLF;

## See also

[FieldDelimiter](#)

[FieldDelimiterCustom](#)

[Fixed](#)

[RecordSeparator](#)

[TextQualifier](#)

[TextQualifierCustom](#)

## RecordSeparatorCustom property example

RecordSeparatorCustom := #12;

**See also**

[RowLast](#)

### RowFirst property example

```
{load a data starting from second row}  
RowFirst := 2;
```

**See also**

[RowFirst](#)



### RowLast property example

```
{limit up to 150 rows only}  
RowLast := 150;
```

## **See also**

[TSMITextQualifier](#)

[FieldDelimiter](#)

[FieldDelimiterCustom](#)

[Fixed](#)

[RecordSeparator](#)

[RecordSeparatorCustom](#)

[TextQualifierCustom](#)

## TextQualifier property example

TextQualifier := tqApos;

## **See also**

[FieldDelimiter](#)

[FieldDelimiterCustom](#)

[Fixed](#)

[RecordSeparator](#)

[RecordSeparatorCustom](#)

[TextQualifier](#)

## TextQualifierCustom property example

TextQualifierCustom := '#';

**See also**

[Picture](#)  
[Title](#)

### TitleStatus property example

TitleStatus := 'Progress of data loading';

**See also**

[SaveSpecification](#)

[TOnGetSpecifications](#)



### **LoadSpecification method example**

{1. load a specification with pre-defined settings}  
`smi.LoadSpecification('C:\ForLoad\remote_office.smi');`

{2. start an import process}  
`smi.Execute;`

**See also**

[TableType](#)

## See also

[Extension](#)  
[TableType](#)

## See also

[About](#)

[AboutSMImport](#)

**See also**

[ExecuteWithoutDialog](#)

### **Execute method example**

{1. load a specification with pre-defined settings}  
smi.LoadSpecification('C:\ForLoad\remote\_office.smi');

{2. start an import process}  
smi.Execute;

## See also

[LoadSpecification](#)

[TOnGetSpecifications](#)

### SaveSpecification method example

{1. define the import properties}

```
smi.Fixed := True;  
smi.Mappings.Clear;  
smi.Mappings.Add('Account=Text1-10');  
smi.Mappings.Add('Amount=Text12-15');
```

{2. save a specification}

```
smi.SaveSpecification('C:\ForLoad\remote_office.smi');
```

{3. start an import process}

```
smi.Execute;
```



**See also**

[OnBeforeExecute](#)

[OnAfterRecordEvent](#)

[OnBeforeRecordEvent](#)

[OnErrorEvent](#)

### OnAfterExecute event example

begin

{show a text in status bar}

StatusBar.Text := 'Imported.';

ShowMessage(Format('Import is completed. Result statistic: %d processed, %d added, %d updated, %d deleted, %d errors', [SMIComponent.Statistic.TotalImported, SMIComponent.Statistic.TotalAdded, SMIComponent.Statistic.TotalUpdated, SMIComponent.Statistic.TotalDeleted, SMIComponent.Statistic.TotalErrors])  
end;

**See also**

[OnBeforeRecordEvent](#)  
[TAfterRecordEvent](#)

[OnAfterExecute](#)  
[OnBeforeExecuteEvent](#)  
[OnErrorEvent](#)

### OnAfterRecordEvent event example

```
begin
{stop an import process if added 50 records}
if SMIComponent.Statistic.TotalImported > 50 then
Abort := True
end;
```

**See also**

[OnAfterExecute](#)

[OnAfterRecordEvent](#)

[OnBeforeRecordEvent](#)

[OnErrorEvent](#)

### OnBeforeExecute event example

```
begin  
  {show a text in status bar}  
  StatusBar.Text := 'Importing...';  
end;
```

**See also**

[OnAfterRecordEvent](#)  
[TBeforeRecordEvent](#)

[OnAfterExecute](#)  
[OnBeforeExecuteEvent](#)  
[OnErrorEvent](#)

### OnBeforeRecordEvent event example

```
begin  
  {update a progress bar}  
  ProgressBar.Position := SMIComponent.Statistic.TotalImported;  
end;
```



**See also**

[TErrorEvent](#)

[OnBeforeRecordEvent](#)

[OnAfterRecordEvent](#)

[OnAfterExecute](#)

[OnBeforeExecuteEvent](#)

### OnErrorEvent event example

```
procedure TForm1.SMIWizardDlgErrorEvent(Sender: TObject;  
Error: Exception; var Abort: Boolean);  
begin  
  {add to internal log}  
  lbErrorLog.Items.Add(Error.Message);  
  
  {if we have more than 100 errors, then cancel an import process}  
  if (SMIWizardDlg.Statistic.TotalErrors > 100) then  
    Abort := True;  
end;
```

**See also**

[TGetCellParamsEvent](#)  
[OnErrorEvent](#)

### OnGetCellParams event example

```
procedure TForm1.SMImportFromText(Sender: TObject; Field: TField; var Value:
Variant);
begin
if Assigned(Field) and
(Field.FieldName = 'CustomerID') then
begin
Value := 'UA' + VarToStr(Value);
end
end;
```

## OnCreateStructure event - See also

[TSMIColumn](#)

[TSMIColumns](#)

[TSMIOnCreateStructure](#)

## OnCreateStructure event - Example

```
procedure TForm1.SMImportFromCSVCreateStructure(Sender: TObject;
Columns: TSMIColumns);
var
i: Integer;
begin
{delete all current dataset fields}
ClientDataSet1.Close;
ClientDataSet1.FieldDefs.Clear;

{create a structure by parsed columns from CSV}
for i := 0 to Columns.Count-1 do
begin
if Columns[i].Size = 0 then
Columns[i].Size := 30;

case Columns[i].DataType of
itString: ClientDataSet1.FieldDefs.Add(Columns[i].FieldName, ftString, Columns[i].Size,
False);
itInteger: ClientDataSet1.FieldDefs.Add(Columns[i].FieldName, ftInteger, 0, False);
itFloat: ClientDataSet1.FieldDefs.Add(Columns[i].FieldName, ftFloat, 0, False);
itDateTime: ClientDataSet1.FieldDefs.Add(Columns[i].FieldName, ftDateTime, 0, False);
itDate: ClientDataSet1.FieldDefs.Add(Columns[i].FieldName, ftDate, 0, False);
itTime: ClientDataSet1.FieldDefs.Add(Columns[i].FieldName, ftTime, 0, False);
itBoolean: ClientDataSet1.FieldDefs.Add(Columns[i].FieldName, ftBoolean, 0, False);
else
ClientDataSet1.FieldDefs.Add(Columns[i].FieldName, ftString, 10, False);
end;
end;
ClientDataSet1.CreateDataset;

{open a dataset with created fields (from text file)}
ClientDataSet1.Active := True;

{fill default correspondence between fields in dataset and columns in CSV}
SMImportFromCSV.Mappings.Clear;
SMImportFromCSV.Columns2Mapping;
end;
```

## See also

[PSpreadSheetRow](#)

[TCustomSpreadCols](#)

[TCustomSpreadSheet](#)

[TSpreadSheet](#)

## Properties

▶ Run-time only    ■ Key properties

▶ ■ [ColCount](#)

▶ ■ [RowCount](#)



## Methods

- Key methods

~~Destroy~~{linkDelphi=Destroy\_Method}

- [AddRow](#)

- [GetRow](#)

- [GetValue](#)

~~Clear~~{linkDelphi=Clear\_Method}

- [RemoveRow](#)

- [SetValue](#)

# SMCells unit

[See also](#)

---

<<< Description of the unit >>>

## Components

[TSpreadSheetCells](#)

[TCustomSpreadSheet](#)

[TSpreadSheet](#)

## Types

[PSpreadSheetRow](#)

[TOnCellValue](#)

[TOnColumnWidth](#)

[TOnDimensions](#)

[TOnRowHeight](#)

[TSpreadSheetCols](#)

[TSpreadSheetRow](#)

## Constants

MAXCol

MAXRow

**See also**

[RowCount](#)

**See also**

[ColCount](#)

## See also

[GetRow](#)

[RemoveRow](#)

## See also

[AddRow](#)

[RemoveRow](#)

**See also**

[SetValue](#)

## See also

[AddRow](#)

[GetRow](#)



**See also**

[GetValue](#)

**See also**

[TSpreadSheet](#)

[TMSExcels](#)

[TQuattroPro](#)

[TLotus123](#)

## Properties

- ▶ Run-time only    ■ Key properties
  - [FileName](#)
  - [OnCellValue](#)
  - [OnColumnWidth](#)
  - [OnDimensions](#)
  - [OnRowHeight](#)

**See also**

[TCustomSpreadSheet](#)

## Methods

- Key methods

Create{linkDelphi=Create\_Method}

Destroy{linkDelphi=Destroy\_Method}

- [LoadFromFile](#)

## See also

[TSMImportBaseComponent](#)

[TSMIWizardDlg](#)

[TSMImportFromText](#)

[TSMImportFromXLS](#)

[TSMImportFromWKS](#)

[TSMImportFromQuattro](#)

[TSMImportFromXML](#)

[TSMImportFromHTML](#)

[TSMImportFromDataset](#)

## Properties

- Run-time only
- Key properties
- [TableType](#)

[About](#)

[AnimatedStatus](#)

[DataFormats](#)

[DataSet](#)

[DatasetKeys](#)

[Mappings](#)

[Mode](#)

[Options](#)

[RowFirst](#)

[RowLast](#)

[Statistic](#)

[TitleStatus](#)

## Methods

- Key methods

Create{linkDelphi=Create\_Method}

[LoadSpecification](#)

~~Destroy~~{linkDelphi=Destroy\_Method}

[Extension](#)

[FillFileFilters](#)

[AboutSMI](#)

[Execute](#)

[SaveSpecification](#)



## Events

[OnAfterExecute](#)

[OnAfterRecordEvent](#)

[OnBeforeExecute](#)

[OnBeforeRecordEvent](#)

[OnErrorEvent](#)

[OnGetCellParams](#)

# SMI2BDE unit

[See also](#)

---

In this unit was declared a TSMImportFromBDE component for importing from Paradox and DBase tables.

## Components

[TSMImportFromBDE](#)

## Routines

[GetAvailableImportFormats](#)

## **See also**

[TSMImportBaseComponent](#)

[TSMIWizardDlg](#)

[TSMImportFromText](#)

[TSMImportFromXLS](#)

[TSMImportFromWKS](#)

[TSMImportFromQuattro](#)

[TSMImportFromXML](#)

[TSMImportFromHTML](#)

[TSMImportFromBDE](#)

## Properties

- Run-time only
- Key properties
- [SourceDataset](#)

[About](#)

[AnimatedStatus](#)

[DataFormats](#)

[DataSet](#)

[DatasetKeys](#)

[Mappings](#)

[Mode](#)

[Options](#)

[RowFirst](#)

[RowLast](#)

[Statistic](#)

[TitleStatus](#)

## Events

[OnAfterExecute](#)

[OnAfterRecordEvent](#)

[OnBeforeExecute](#)

[OnBeforeRecordEvent](#)

[OnErrorEvent](#)

[OnGetCellParams](#)

# SMI2DS unit

[See also](#)

---

In this unit was declared TSMImportFromDataset component which allow to transfer a data from one dataset to other.

## Components

[TSMImportFromDataSet](#)

## See also

[Dataset](#)

[SourceFileName](#)

## Properties

- ▶ Run-time only
- Key properties
  - [SheetIndex](#)
  - [Version](#)



## Methods

- Key methods

Create{linkDelphi=Create\_Method}

Destroy{linkDelphi=Destroy\_Method}

LoadFromFile{linkDelphi=LoadFromFile\_Method}

# SMXLS unit

[See also](#)

---

In this unit is declared a component for direct reading of MS Excel spreadsheets.

## Components

[TMSExcel](#)

## Types

[TExcelVersion](#)

**See also**

[TExcelVersion](#)

## Properties

- ▶ Run-time only
- Key properties
- [Version](#)

## Methods

- Key methods

~~LoadFromFile~~{linkDelphi=LoadFromFile\_Method}

# SMWQ unit

[See also](#)

---

In this unit is declared a component for direct reading of Corel QuattroPro spreadsheets.

## Components

[TQuattroPro](#)

## Types

[TWQVersion](#)

**See also**

[TWQVersion](#)

## Properties

- ▶ Run-time only
- Key properties
- [Version](#)



## Methods

- Key methods

~~LoadFromFile~~{linkDelphi=LoadFromFile\_Method}

# SMWKS unit

[See also](#)

---

In this unit is declared a component for direct reading of Lotus 1-2-3 spreadsheets.

## Components

[TLotus123](#)

## Types

[TWKSVersion](#)

## See also

[TWKSVersion](#)

## Properties

- Run-time only
- Key properties

~~RowFirst~~{linkDelphi=RowFirst\_Property}

~~RowLast~~{linkDelphi=RowLast\_Property}

~~SourceFileName~~{linkDelphi=SourceFileName\_Property}

[About](#)

[AnimatedStatus](#)

[DataFormats](#)

[DataSet](#)

[DatasetKeys](#)

[Mappings](#)

[Mode](#)

[Options](#)

[Statistic](#)

[TitleStatus](#)

## Methods

- Key methods

~~Destroy~~{linkDelphi=Destroy\_Method}

[LoadSpecification](#)

~~Create~~{linkDelphi=Create\_Method}

[Extension](#)

[FillFileFilters](#)

[AboutSMI](#)

[Execute](#)

[SaveSpecification](#)

## Events

[OnAfterExecute](#)

[OnAfterRecordEvent](#)

[OnBeforeExecute](#)

[OnBeforeRecordEvent](#)

[OnErrorEvent](#)

[OnGetCellParams](#)

# SMI2Cell unit

[See also](#)

---

In this unit is declared a basic component for loading spreadsheets (MS Excel, QuattroPro, Lotus 1-2-3 etc) into dataset.

## Components

[TSMImportFromCell](#)

## **See also**

[TSMImportBaseComponent](#)

[TSMIWizardDlg](#)

[TSMImportFromText](#)

[TSMImportFromXLS](#)

[TSMImportFromWKS](#)

[TSMImportFromQuattro](#)

[TSMImportFromXML](#)

[TSMImportFromBDE](#)

[TSMImportFromDataset](#)



## Properties

- Run-time only
- Key properties

~~RowFirst~~{linkDelphi=RowFirst\_Property}

~~RowLast~~{linkDelphi=RowLast\_Property}

~~SourceFileName~~{linkDelphi=SourceFileName\_Property}

[About](#)

[AnimatedStatus](#)

[DataFormats](#)

[DataSet](#)

[DatasetKeys](#)

[Mappings](#)

[Mode](#)

[Options](#)

[Statistic](#)

[TitleStatus](#)

## Methods

### ■ Key methods

~~Extension~~{linkDelphi=Extension\_Method}

~~FillFileFilters~~{linkDelphi=FillFileFilters\_Method}

[LoadSpecification](#)

~~Create~~{linkDelphi=Create\_Method}

~~Destroy~~{linkDelphi=Destroy\_Method}

[AboutSMI](#)

[Execute](#)

[SaveSpecification](#)

## Events

[OnAfterExecute](#)

[OnAfterRecordEvent](#)

[OnBeforeExecute](#)

[OnBeforeRecordEvent](#)

[OnErrorEvent](#)

[OnGetCellParams](#)

# SMI2HTML unit

[See also](#)

---

In this unit was declared a TSMImportFromHTML component for direct HTML loading.

## Components

[TSMImportFromHTML](#)

## See also

[TSMImportBaseComponent](#)

[TSMIWizardDlg](#)

[TSMImportFromXLS](#)

[TSMImportFromWKS](#)

[TSMImportFromQuattro](#)

[TSMImportFromXML](#)

[TSMImportFromHTML](#)

[TSMImportFromBDE](#)

[TSMImportFromDataset](#)

## Properties

- Run-time only
- Key properties

FieldDelimiter{linkDelphi=FieldDelimiter\_Property}

FieldDelimiterCustom{linkDelphi=FieldDelimiterCustom\_Property}

Fixed{linkDelphi=Fixed\_Property}

RecordSeparator{linkDelphi=RecordSeparator\_Property}

RecordSeparatorCustom{linkDelphi=RecordSeparatorCustom\_Property}

RowFirst{linkDelphi=RowFirst\_Property}

RowLast{linkDelphi=RowLast\_Property}

SourceFileName{linkDelphi=SourceFileName\_Property}

TextQualifier{linkDelphi=TextQualifier\_Property}

TextQualifierCustom{linkDelphi=TextQualifierCustom\_Property}

[About](#)

[AnimatedStatus](#)

[DataFormats](#)

[DataSet](#)

[DatasetKeys](#)

[Mappings](#)

[Mode](#)

[Options](#)

[Statistic](#)

[TitleStatus](#)

## Methods

### ■ Key methods

~~Extension~~{linkDelphi=Extension\_Method}

~~FillFileFilters~~{linkDelphi=FillFileFilters\_Method}

[LoadSpecification](#)

~~Create~~{linkDelphi=Create\_Method}

~~Destroy~~{linkDelphi=Destroy\_Method}

[AboutSMI](#)

[Execute](#)

[SaveSpecification](#)

## Events

[OnAfterExecute](#)

[OnAfterRecordEvent](#)

[OnBeforeExecute](#)

[OnBeforeRecordEvent](#)

[OnErrorEvent](#)

[OnGetCellParams](#)



# SMI2TXT unit

[See also](#)

---

In this unit was declared a TSMImportFromText component which allow to load a data from fixed text files or comma-delimited files.

## Components

[TSMImportFromText](#)

## See also

[TSMImportBaseComponent](#)

[TSMIWizardDlg](#)

[TSMImportFromText](#)

[TSMImportFromXLS](#)

[TSMImportFromQuattro](#)

[TSMImportFromXML](#)

[TSMImportFromHTML](#)

[TSMImportFromBDE](#)

[TSMImportFromDataset](#)

## Methods

### ■ Key methods

~~Extension~~{linkDelphi=Extension\_Method}

~~FillFileFilters~~{linkDelphi=FillFileFilters\_Method}

[LoadSpecification](#)

~~Create~~{linkDelphi=Create\_Method}

~~Destroy~~{linkDelphi=Destroy\_Method}

[AboutSMI](#)

[Execute](#)

[SaveSpecification](#)

## Events

[OnAfterExecute](#)

[OnAfterRecordEvent](#)

[OnBeforeExecute](#)

[OnBeforeRecordEvent](#)

[OnErrorEvent](#)

[OnGetCellParams](#)

# SMI2WKS unit

[See also](#)

---

In this unit was declared a TSMImportFromWKS component for direct importing from Lotus 1-2-3 spreadsheets.

## Components

[TSMImportFromWKS](#)

## See also

[TSMImportBaseComponent](#)

[TSMIWizardDlg](#)

[TSMImportFromText](#)

[TSMImportFromXLS](#)

[TSMImportFromWKS](#)

[TSMImportFromXML](#)

[TSMImportFromHTML](#)

[TSMImportFromBDE](#)

[TSMImportFromDataset](#)

## Methods

### ■ Key methods

~~Extension~~{linkDelphi=Extension\_Method}

~~FillFileFilters~~{linkDelphi=FillFileFilters\_Method}

[LoadSpecification](#)

~~Create~~{linkDelphi=Create\_Method}

~~Destroy~~{linkDelphi=Destroy\_Method}

[AboutSMI](#)

[Execute](#)

[SaveSpecification](#)

## Events

[OnAfterExecute](#)

[OnAfterRecordEvent](#)

[OnBeforeExecute](#)

[OnBeforeRecordEvent](#)

[OnErrorEvent](#)

[OnGetCellParams](#)



# SMI2WQ unit

[See also](#)

---

In this unit was declared a TSMImportFromQuattro component for direct importing from QuattroPro spreadsheets.

## Components

[TSMImportFromQuattro](#)

## See also

[TSMImportBaseComponent](#)

[TSMIWizardDlg](#)

[TSMImportFromText](#)

[TSMImportFromWKS](#)

[TSMImportFromQuattro](#)

[TSMImportFromXML](#)

[TSMImportFromHTML](#)

[TSMImportFromBDE](#)

[TSMImportFromDataset](#)

## Methods

### ■ Key methods

~~Extension~~{linkDelphi=Extension\_Method}

~~FillFileFilters~~{linkDelphi=FillFileFilters\_Method}

[LoadSpecification](#)

~~Create~~{linkDelphi=Create\_Method}

~~Destroy~~{linkDelphi=Destroy\_Method}

[AboutSMI](#)

[Execute](#)

[SaveSpecification](#)

## Events

[OnAfterExecute](#)

[OnAfterRecordEvent](#)

[OnBeforeExecute](#)

[OnBeforeRecordEvent](#)

[OnErrorEvent](#)

[OnGetCellParams](#)

# SMI2XLS unit

[See also](#)

---

In this unit was declared a TSMImportFromXLS component for direct importing from MS Excel spreadsheets.

## Components

[TSMImportFromXLS](#)

## See also

[TSMImportBaseComponent](#)

[TSMIWizardDlg](#)

[TSMImportFromText](#)

[TSMImportFromXLS](#)

[TSMImportFromWKS](#)

[TSMImportFromQuattro](#)

[TSMImportFromHTML](#)

[TSMImportFromBDE](#)

[TSMImportFromDataset](#)

## Properties

- Run-time only
- Key properties

~~RowFirst~~{linkDelphi=RowFirst\_Property}

~~RowLast~~{linkDelphi=RowLast\_Property}

~~SourceFileName~~{linkDelphi=SourceFileName\_Property}

[About](#)

[AnimatedStatus](#)

[DataFormats](#)

[DataSet](#)

[DatasetKeys](#)

[Mappings](#)

[Mode](#)

[Options](#)

[Statistic](#)

[TitleStatus](#)

## Methods

### ■ Key methods

~~Extension~~{linkDelphi=Extension\_Method}

~~FillFileFilters~~{linkDelphi=FillFileFilters\_Method}

[LoadSpecification](#)

~~Create~~{linkDelphi=Create\_Method}

~~Destroy~~{linkDelphi=Destroy\_Method}

[AboutSMI](#)

[Execute](#)

[SaveSpecification](#)



## Events

[OnAfterExecute](#)

[OnAfterRecordEvent](#)

[OnBeforeExecute](#)

[OnBeforeRecordEvent](#)

[OnErrorEvent](#)

[OnGetCellParams](#)

# SMI2XML unit

[See also](#)

---

In this unit was declared a TSMImportFromXML component for direct XML loading.

## Components

[TSMImportFromXML](#)

**See also**

[UserAccess](#)

[TSMIRestriction](#)

[TSMIRestrictions](#)

## Properties

- ▶ Run-time only    ■ Key properties
  - [FieldAdjustment](#)
  - [FieldDelimiter](#)
  - [FieldMapping](#)
  - [FirstRow](#)
  - [ImportMode](#)
  - [LastRow](#)
  - [PreviewData](#)
  - [RecordSeparator](#)
  - [SourceFileName](#)
  - [Specification](#)
  - [TableType](#)
  - [TextQualifier](#)
  - [TextType](#)

## Methods

- Key methods
- [Create](#)

# SMIWiz unit

[See also](#)

---

In this unit was declared TSMIWizardDlg component which implement compound import component with end-user wizard dialog for step-by-step import instructions.

Also in this unit was declared types for end-user restrictions.

## Components

[TSMIWizardDlg](#)

## Objects

[TSMIUserAccess](#)

## Types

[TSMIGetSpecificationsEvent](#)

[TSMIRestriction](#)

[TSMIRestrictions](#)

**See also**

[TSMIRestriction](#)

[UserAccess](#)

[TSMIuserAccess](#)

**See also**

[TSMIRestriction](#)

[UserAccess](#)

[TSMIuserAccess](#)



**See also**

[TSMIRestriction](#)

[UserAccess](#)

[TSMIuserAccess](#)

**See also**

[TSMIRestriction](#)

[UserAccess](#)

[TSMIuserAccess](#)

**See also**

[TSMIRestriction](#)

[UserAccess](#)

[TSMIuserAccess](#)

**See also**

[TSMIRestriction](#)

[UserAccess](#)

[TSMIuserAccess](#)

**See also**

[TSMIRestriction](#)

[UserAccess](#)

[TSMIuserAccess](#)

**See also**

[TSMIRestriction](#)

[UserAccess](#)

[TSMIuserAccess](#)

**See also**

[TSMIRestriction](#)

[UserAccess](#)

[TSMIuserAccess](#)

**See also**

[TSMIRestriction](#)

[UserAccess](#)

[TSMIuserAccess](#)



## **See also**

[TSMIRestriction](#)  
[UserAccess](#)  
[TSMIuserAccess](#)

[Extension](#)

**See also**

[TSMIRestriction](#)

[UserAccess](#)

[TSMIuserAccess](#)

**See also**

[TSMIRestriction](#)

[UserAccess](#)

[TSMIuserAccess](#)

## **See also**

[TSMImportBaseComponent](#)

[TSMImportFromText](#)

[TSMImportFromXLS](#)

[TSMImportFromWKS](#)

[TSMImportFromQuattro](#)

[TSMImportFromXML](#)

[TSMImportFromHTML](#)

[TSMImportFromBDE](#)

[TSMImportFromDataset](#)

## Properties

- Run-time only
- Key properties

FieldDelimiter{linkDelphi=FieldDelimiter\_Property}

FieldDelimiterCustom{linkDelphi=FieldDelimiterCustom\_Property}

Fixed{linkDelphi=Fixed\_Property}

- [Formats](#)

- [Picture](#)

RecordSeparator{linkDelphi=RecordSeparator\_Property}

RecordSeparatorCustom{linkDelphi=RecordSeparatorCustom\_Property}

RowFirst{linkDelphi=RowFirst\_Property}

RowLast{linkDelphi=RowLast\_Property}

SourceFileName{linkDelphi=SourceFileName\_Property}

- [TableType](#)

TextQualifier{linkDelphi=TextQualifier\_Property}

TextQualifierCustom{linkDelphi=TextQualifierCustom\_Property}

- [Title](#)

- [UseDisplayNames](#)

- [UserAccess](#)

[About](#)

[AnimatedStatus](#)

[DataFormats](#)

[DataSet](#)

[DatasetKeys](#)

[Mappings](#)

[Mode](#)

[Options](#)

[Statistic](#)

[TitleStatus](#)

## Methods

- Key methods

Create{linkDelphi=Create\_Method}

Destroy{linkDelphi=Destroy\_Method}

- [Execute](#)

- [ExecuteWithoutDialog](#)

[LoadSpecification](#)

[Extension](#)

[FillFileFilters](#)

[AboutSMI](#)

[SaveSpecification](#)

## Events

- Key events
- [OnGetSpecifications](#)

[OnAfterExecute](#)

[OnAfterRecordEvent](#)

[OnBeforeExecute](#)

[OnBeforeRecordEvent](#)

[OnErrorEvent](#)

[OnGetCellParams](#)

## See also

[TitleStatus](#)  
[Title](#)



### Picture property example

{load a logo from external file}

```
Picture.LoadFromFile('c:\My Documents\company.bmp');
```

{load a logo from linked resource file}

```
Picture.Bitmap.Handle := LoadBitmap(hInstance, 'LOGO');
```

**See also**

[TTableTypeImport](#)

## See also

[Picture](#)

[TitleStatus](#)

### Title property example

```
smi.Title := 'Step-by-step data loading';
```

**See also**

[TSMIUserAccess](#)

[TSMIRestriction](#)

[TSMIRestrictions](#)

**See also**

[ExecuteWithoutDialog](#)

### Execute method example

{1. load a specification with pre-defined settings}  
smi.LoadSpecification('C:\ForLoad\remote\_office.smi');

{2. show a wizard with step-by-step instructions}  
smi.Execute;

**See also**

[Execute](#)



### **ExecuteWithoutDialog method example**

{1. load a specification}

```
smi.LoadSpecification('C:\ForLoad\remote_office.smi');
```

{2. "silence" data loading by specification}

```
smi.ExecuteWithoutDialog;
```

**See also**

[LoadSpecification](#)

[SaveSpecification](#)

[TSMIGetSpecificationsEvent](#)

### **OnGetSpecifications event example**

```
IstFiles.AddObject('Country specification (fixed list)',  
TObject(LongInt(NewStr('d:\spec\country1.smi'))));
```

# ThousandSeparator property

[See also](#)

---

## Applies to

[TSMIDataFormats](#) object

## Declaration

**property** ThousandSeparator: Char;

## Description

Using this property you can customize the ThousandSeparator character for any numeric and currency values.

By default is a current value from Regional settings of Windows.

# TAfterRecordEvent type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TAfterRecordEvent = procedure(Sender: TObject; var Abort:  
Boolean); of object;
```

## Description

This type is defined for OnAfterRecordEvent that allow to control an import process after each loaded row.

# TBeforeRecordEvent type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TBeforeRecordEvent = procedure(Sender: TObject; const Fields:  
string; Values: Variant; var Accept: Boolean); of object;
```

## Description

This type is defined for OnBeforeRecordEvent that allow to control an import process before loading of any row.

Also using this event you can skip some row from loading (just by your custom condition).

# TErrorEvent type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TErrorEvent = procedure (Sender: TObject; Error: Exception; var
Abort: Boolean); of object;
```

## Description

This type declared for events which will be called after any error during importing.

The Sender is a current import component where was raised the exception.  
The Error is standard exception type.

If you want to stop a process, you must return True in Abort parameter.  
But if you want to ignore this error and continue the import, just return a False.

In this event you can handle the any errors and, for example, to save the "bad" values in some buffer for next modification and re-loading.

# TGetCellParamsEvent type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TGetCellParamsEvent = procedure (Sender: TObject; Field: TField; var Value: Variant); of object;
```

## Description

This type declared for events where you can control import process in own hands. The event will be called for each imported value (each field processing for any records).

If you want to change a result value which will be imported in destination dataset, you must change a Value parameter.

The Field parameter will show a field of destination dataset where will be placed a value.

The Sender is an import component which was used for data loading and parsing.



# TImportFormatTypes type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TImportFormatTypes = set of TTableTypeImport;
```

## Description

This type is a set of supported formats of external files.

# TImportMode type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TImportMode = (imAppend, imUpdate, imAppendUpdate, imDelete, imCopy);
```

## Description

This type is enumeration for available import modes:

1. imAppend - the each record from source file will be inserted in destination dataset (appends all records). The destination must not have any records with the key of the any of the records in the source.
2. imUpdate - the records in destination dataset will be updated with "similar" (by key fields) records in source file. Each record in the source must have a record in the destination with the same key the source.
3. imAppendUpdate - the records in destination dataset will be updated with "similar" records in source file and rest records will be inserted there (appends any records which do not already exist and replaces those which do)
4. imDelete - in destination dataset will be deleted the records which have "similar" (by key fields) records in source file. Each source record must have a key which is also found in the destination.
5. imCopy - makes an exact duplicate of the source table. Before importing the all records in destination will be deleted and after that inserted the records from source file.

The imAppend is the default mode.

# TSMIAbout type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TSMIAbout = string;
```

## Description

This type is design-time only and is declared for About property that allow to receive a short information about SMIImport suite in Delphi/C++Builder IDE from Object Inspector window.

# TSMIDateOrder type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TSMIDateOrder = (doMDY, doDMY, doYMD, doYDM, doDYM, doMYD);
```

## Description

This type provides a possibility to customize the date order for any date/time values.

doMDY is a 'm/d/yy' format of date

doDMY is a 'd/m/yy' format of date

doYMD is a 'yy/m/d' format of date

doYDM is a 'yy/d/m' format of date

doDYM is a 'd/yy/m' format of date

doMYD is a 'm/yy/d' format of date

# TSMIFieldDelimiter type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TSMIFieldDelimiter = (fdNone, fdCustom, fdTab, fdSemicolon,  
fdComma, fdSpace);
```

## Description

This type declared for easy delimiter setting using list of pre-defined field delimiters:

1. fdNone - your external text file haven't any delimiter between fields
2. fdCustom - you have a custom delimiter which defined in FieldDelimiterCustom
3. fdTab - the delimiter is tabular character
4. fdSemicolon - the delimiter is semicolon (;) character
5. fdComma - the delimiter is comma (,) character
6. fdSpace - the delimiter is space (#32) character

# TSMIOption type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TSMIOption = (soShowMessage, soExtendedStatistic,  
soSkipEmptyRow, soUseAnimatedControl, soWaitCursor);
```

## Description

This type enumerates the supported options during importing:

**soShowMessage:** if you'll exclude this flag from Options property, then import will be "silence" - your user will not receive any interactive dialogs. Else user can receive a dialog with warning, error etc

This option is very useful for web-applications, server-application or other multi-tier environments.

**soExtendedStatistic:** if you'll include this flag, during import process will be displayed an extended information about data loading - total processed rows, number of errors, number of added records, number of updated records and number of deleted records

**soSkipEmptyRow:** allow to skip all empty lines in text file.

**soUseAnimatedControl:** if flag is not included in options, then animated picture is not created in status dialog. Enabled animated will increase a time and reduce a speed of import process.

**soWaitCursor:** please exclude this option if you develop server-side tool. If flag is included, when import process is started, default cursor will be changed to crHourGlass and restored to default when process is finished.

# TSMIOptions type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TSMIOptions = set of TSMIOption;
```

## Description

This type is a list of supported options during import process.

# TSMIRecordSeparator type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TSMIRecordSeparator = (rsCustom, rsCRLF, rsCR, rsLF);
```

## Description

This type enumerates a supported pre-defined separators between records (lines) in text file which will be imported:

rsCustom: used a custom separator which was assigned to RecordSeparatorCustom property

rsCRLF: used a "standard" Windows/DOS text separator (#13#10)

rsCR: used a character #13 as separator

rsLF: used a "standard" UNIX separator (#10 character)



# TSMISStatistic type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TSMISStatistic = class
TotalCount: LongInt; {count of records that must be loaded from
file}
TotalImported: LongInt; {number of processed/loaded records}
TotalAdded: LongInt; {number of new added records}
TotalDeleted: LongInt; {number of deleted records}
TotalUpdated: LongInt; {number of modified records}
TotalErrors: LongInt; {number of errors}

UpdateStep: LongInt; {step for visual progress bar updating}
end;
```

## Description

This type provides a possibility to read an extended statistic about import process.

# TSMITextQualifier type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TSMITextQualifier = (tqNone, tqCustom, tqQuot, tqApos);
```

## Description

This type is list of pre-defined text qualifier:

tqNone: the text qualifier is not used

tqQuot: the text is quoted and must be extracted between "

tqApos: the text is quoted and must be extracted between '

tqCustom: used a custom qualifier which was defined in TextQualifierCustom property

# TTableTypeImport type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TTableTypeImport = (teParadox, teDBase, teText, teHTML, teXLS,  
teWKS, teQuattro, teXML, teAccess, teWord);
```

## Description

This type is a list of supported file formats which you can use for importing.

# AboutSMImport procedure

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
procedure AboutSMImport;
```

## Description

This global procedure allow to activate a dialog with short information about SMImport suite and author without import component creation.

# Statistic property

[See also](#)

---

## Applies to

[TSMImportBaseComponent](#) component

## Declaration

```
property Statistic: TSMISStatistic;
```

## Description

You can read an extended statistic information about import process.

This information is available during import process and after end of data loading.

After each starting of data loading the statistic information will be empty so if you need some global information that combine a few import processes, you must summarize it after import finishing.

# TSMIColumn type

[See also Example](#)

---

## Unit

[SMIBase](#)

## Description

This type is an item in collection of parsed columns from external file that could be used for creation of dataset before import will be started.

Next published attributes are available:

property **Alignment**: TAlignment

property **FieldName**: string

property **Caption**: string

property **DataType**: TSMIDataType

property **Size**: Integer

property **Precision**: Integer

# TSMIColumns type

[See also](#)

---

## Unit

[SMIBase](#)

## Description

This type is a collection of parsed columns that could be used for dataset creation by data from external file.

# TSMIOnCreateStructure type

---

## Unit

[SMIBase](#)

## Declaration

```
type TSMIOnCreateStructure = procedure (Sender: TObject; Columns: TSMIColumns); of object;
```

## Description

This type is declared for [OnCreateStructure event](#) that allow to create a dataset with parsed structure from external file.



# PSpreadSheetRow type

[See also](#)

---

## Unit

[SMCells](#)

## Declaration

```
type PSpreadSheetRow = ^TSpreadSheetRow;
```

## Description

This is an internal type for spreadsheet reading and interaction with virtual array.

# TSpreadSheetCols type

[See also](#)

---

## Unit

[SMCells](#)

## Declaration

```
type TSpreadSheetCols = array[0..MAXCol] of Variant;
```

## Description

This type is declaration of column array in each row of virtual array.

**See also**

[SMI2Cell unit](#)

[SMXLS unit](#)

[SMWQ unit](#)

[SMWKS unit](#)

# TOnCellValue type

[See also](#)

---

## Unit

[SMCells](#)

## Declaration

```
type TOnCellValue = procedure (Sender: TObject; Row, Column: Integer; Value: Variant); of object;
```

## Description

This type is used for OnCellValue event declaration that allow to load a value to any cell of virtual array.

# TOnColumnWidth type

[See also](#)

---

## Unit

[SMCells](#)

## Declaration

```
type TOnColumnWidth = procedure(Sender: TObject; ColumnNo: Integer;  
Width: Integer); of object;
```

## Description

This type is used for OnColumnWidth event declaration that allow to load a width of any column in virtual array.

# TOnDimensions type

[See also](#)

---

## Unit

[SMCells](#)

## Declaration

```
type TOnDimensions = procedure(Sender: TObject; FirstRow, LastRow,  
FirstColumn, LastColumn: Integer); of object;
```

## Description

This type is used for OnDimensions event declaration that allow to define a number of total rows and columns of virtual array.

# TOnRowHeight type

[See also](#)

---

## Unit

[SMCells](#)

## Declaration

```
type TOnRowHeight = procedure (Sender: TObject; RowNo: Integer;  
Height: Integer); of object;
```

## Description

This type is used for OnRowHeight event declaration that allow to define a row height in the virtual array.

# TSpreadSheetRow type

[See also](#)

---

## Unit

[SMCells](#)

## Declaration

```
type TSpreadSheetRow = record  
  Index: Integer;  
  Cols: TSpreadSheetCols;  
end;
```

## Description

This is an internal type for spreadsheet reading and interaction with virtual array.



## **See also**

[SMIBase unit](#)

[SMI2DS unit](#)

[SMI2Cell unit](#)

[SMI2HTML unit](#)

[SMI2TXT unit](#)

[SMI2WKS unit](#)

[SMI2WQ unit](#)

[SMI2XLS unit](#)

[SMI2XML unit](#)

[SMIWiz unit](#)

# GetAvailableImportFormats function

[See also](#)

---

## Unit

[SMI2BDE](#)

## Declaration

```
function GetAvailableImportFormats: TImportFormatTypes;
```

## Description

This function returns a list of available file formats. By default, if BDE is not installed on computer, then BDE's file formats (Paradox+DBase) will be excluded from set of available formats.

## **See also**

[SMIBase unit](#)  
[SMI2BDE unit](#)  
[SMI2Cell unit](#)  
[SMI2HTML unit](#)  
[SMI2TXT unit](#)  
[SMI2WKS unit](#)  
[SMI2WQ unit](#)  
[SMI2XLS unit](#)  
[SMI2XML unit](#)  
[SMIWiz unit](#)

**See also**

[SMCells unit](#)

[SMWQ unit](#)

[SMWKS unit](#)

# TExcelVersion type

[See also](#)

---

## Unit

[SMXLS](#)

## Declaration

```
type TExcelVersion = (evUnknown, evExcel2, evExcel3, evExcel4,  
evExcel5, evExcel7, evExcel8);
```

## Description

This type is enumerates the available spreadsheet versions which are supported by SMIimport suite for direct reading without any external libraries.

**See also**

[SMCells unit](#)

[SMXLS unit](#)

[SMWKS unit](#)

# TWQVersion type

[See also](#)

---

## Unit

[SMWQ](#)

## Declaration

```
type TWQVersion = (qvUnknown, qvWQ1, qvWQ2);
```

## Description

This type is enumerates the available spreadsheet versions which are supported by SMIimport suite for direct reading without any external libraries.

**See also**

[SMCells unit](#)

[SMXLS unit](#)

[SMWQ unit](#)



# TWKSVersion type

[See also](#)

---

## Unit

[SMWKS](#)

## Declaration

```
type TWKSVersion = (wvUnknown, wvWKS1, wvWKS2, wvWR1);
```

## Description

This type is enumerates the available spreadsheet versions which are supported by SMIImport suite for direct reading without any external libraries.

**See also**

[Version](#)

## **See also**

[SMICells unit](#)

[SMIBase unit](#)

[SMI2BDE unit](#)

[SMI2DS unit](#)

[SMI2HTML unit](#)

[SMI2TXT unit](#)

[SMI2WKS unit](#)

[SMI2WQ unit](#)

[SMI2XLS unit](#)

[SMI2XML unit](#)

[SMIWiz unit](#)

## **See also**

[SMIBase unit](#)  
[SMI2BDE unit](#)  
[SMI2DS unit](#)  
[SMI2Cell unit](#)  
[SMI2TXT unit](#)  
[SMI2WKS unit](#)  
[SMI2WQ unit](#)  
[SMI2XLS unit](#)  
[SMI2XML unit](#)  
[SMIWiz unit](#)

## **See also**

[SMIBase unit](#)

[SMI2BDE unit](#)

[SMI2DS unit](#)

[SMI2Cell unit](#)

[SMI2HTML unit](#)

[SMI2WKS unit](#)

[SMI2WQ unit](#)

[SMI2XLS unit](#)

[SMI2XML unit](#)

[SMIWiz unit](#)

## **See also**

[SMIBase unit](#)

[SMI2BDE unit](#)

[SMI2DS unit](#)

[SMI2Cell unit](#)

[SMI2HTML unit](#)

[SMI2TXT unit](#)

[SMI2WQ unit](#)

[SMI2XLS unit](#)

[SMI2XML unit](#)

[SMIWiz unit](#)

## **See also**

[SMIBase unit](#)

[SMI2BDE unit](#)

[SMI2DS unit](#)

[SMI2Cell unit](#)

[SMI2HTML unit](#)

[SMI2TXT unit](#)

[SMI2WKS unit](#)

[SMI2XLS unit](#)

[SMI2XML unit](#)

[SMIWiz unit](#)

## **See also**

[SMIBase unit](#)

[SMI2BDE unit](#)

[SMI2DS unit](#)

[SMI2Cell unit](#)

[SMI2HTML unit](#)

[SMI2TXT unit](#)

[SMI2WKS unit](#)

[SMI2WQ unit](#)

[SMI2XML unit](#)

[SMIWiz unit](#)



## **See also**

[SMIBase unit](#)

[SMI2BDE unit](#)

[SMI2DS unit](#)

[SMI2Cell unit](#)

[SMI2HTML unit](#)

[SMI2TXT unit](#)

[SMI2WKS unit](#)

[SMI2WQ unit](#)

[SMI2XLS unit](#)

[SMIWiz unit](#)

# TSMIRestriction type

[See also](#)

---

## Unit

[SMIWiz](#)

## Declaration

```
type TSMIRestriction = (irDisabled, irReadOnly, irReadWrite);
```

## Description

This type is declared for support of restrictions in wizard component:

irDisabled: the access is disabled

irReadOnly: user can view a current setting but can't change it

irReadWrite: user have a full access to setting - he/she can view and change a current setting

# TSMIRestrictions type

[See also](#)

---

## Unit

[SMIWiz](#)

## Declaration

```
type TSMIRestrictions = set of TSMIRestriction;
```

## Description

This type is declared for support of restrictions in wizard component.

## **See also**

[SMIBase unit](#)

[SMI2BDE unit](#)

[SMI2DS unit](#)

[SMI2Cell unit](#)

[SMI2HTML unit](#)

[SMI2TXT unit](#)

[SMI2WKS unit](#)

[SMI2WQ unit](#)

[SMI2XLS unit](#)

[SMI2XML unit](#)

# TSMIGetSpecificationsEvent type

[See also](#)

---

## Unit

[SMIWiz](#)

## Declaration

```
type TSMIGetSpecificationsEvent = procedure (Sender: TObject;  
lstFiles: TStrings); of object;
```

## Description

This type is declared for OnGetSpecifications event that allow to customize a list of available specification for wizard dialog.

# Formats property

[See also Example](#)

---

## Applies to

[TSMIWizardDlg](#) component

## Declaration

**property** Formats: TImportFormatTypes;

## Description

This property allow to restrict an end-user access to some file formats.

Especially this is very useful when you want to deploy an application without BDE - in this case just exclude the teParadox and teDBase file formats.

## **See also**

[TSMIDataFormats](#)

[BooleanFalse](#)

[BooleanTrue](#)

[DateOrder](#)

[DecimalSeparator](#)

[ThousandSeparator](#)

[FourDigitYear](#)

[LeadingZeroInDate](#)

[TimeSeparator](#)

**See also**

[TBeforeRecordEvent](#)  
[OnAfterRecordEvent](#)



**See also**

[OnBeforeRecordEvent](#)  
[TOnAfterRecordEvent](#)

**See also**

[OnErrorEvent](#)

**See also**

[OnGetCellParams](#)

## See also

[TTableTypeImport](#)  
[Formats](#)

**See also**

[Mode](#)

## See also

[About](#)

[AboutSMI](#)

[AboutSMImport](#)

## See also

[DateOrder](#)

[TSMIDataFormats](#)

## See also

[FieldDelimiter](#)



## See also

[Options](#)

[TSMIOptions](#)

## See also

[Options](#)

[TSMIOption](#)

## See also

[RecordSeparator](#)

**See also**

[Statistic](#)

**See also**

[TextQualifier](#)

**See also**

[TableType](#)

## See also

[About](#)

[AboutSMI](#)

## Statistic property - See also

[TSMIStatistic](#)



# TSMIColumn type

[TSMIDataType](#)

[TSMIColumns](#)

[OnCreateStructure](#)

# TSMIColumns type - See also

[TSMIColumn](#)  
[OnCreateStructure](#)

## See also

[TSpreadSheetRow](#)

## See also

[TSpreadSheetRow](#)

**See also**

[OnCellValue](#)

**See also**

[OnColumnWidth](#)

**See also**

[OnDimensions](#)

**See also**

[OnRowHeight](#)



## See also

[PSpreadSheetRow](#)

[TSpreadSheetCols](#)

**See also**

[TableType](#)

**See also**

[Version](#)

**See also**

[Version](#)

**See also**

[TSMIRestrictions](#)  
[UserAccess](#)

**See also**

[TSMIRestriction](#)  
[UserAccess](#)

**See also**

[OnGetSpecifications](#)

## Formats property - See also

[TImportFormatTypes](#)



## Formats property - Example

To exclude BDE's desktop files:

Formats := Formats - [teParadox, teDBase];

To Include WEB's files:

Formats := Formats + [teHTML, teXML];

# TSMIDataType type

[See also](#)

---

## Unit

[SMIBase](#)

## Declaration

```
type TSMIDataType = (itString, itInteger, itFloat, itDateTime,  
itDate, itTime, itBoolean);
```

## Description

This type enumerate supported types for parsed columns in external files.