New Features in ibaAnalyzer v8.0.0

Authors: M. Verschaeve, P. van Oosten, T. Seitz

Date: 2022-08-26

iba AG • Königswarterstr. 44 • D-90762 Fürth • Germany • www.iba-ag.com

Table of contents

1 Important remarks

1.1 PDO compatibility

If an update to ibaAnalyzer v8 is done, **PDO's containing a database connection configuration have to be adapted and are changed after saving**. Please note that only the DB connections are affected by this incompatibility.

Exception

If a database connection to Oracle or Microsoft SQL Server is configured in the pdo, it is likely that nothing needs to be changed and the extract works without any adaptions.

V7-pdo opened in ibaAnalyzer-v8

The database connection will not work (fail) in most cases. The connection settings need to be updated and the pdo must be saved again.

V8-pdo opened in ibaAnalyzer-v7

The database connection will not work in many cases. The connection settings need to be reconfigured and the pdo must be saved again with version 7.

See chapter 3.5 for details on adapting pdo files from ibaAnalyzer v7 to v8

2 Support for WIBU licensing

Up to now ibaAnalyzer licenses were stored in dongles from the company MARX. Starting from ibaAnalyzer v8 the licenses can additionally be stored in license containers from the company WIBU. There are 2 types of WIBU license containers supported:

- CmStick: This is a USB dongle.
- CmActLicense: This is a so-called "soft container". It is a file on disk that is coupled to some system IDs.

In order to use WIBU licenses the CodeMeter runtime needs to be installed which is automatically done during the installation of ibaAnalyzer. Information about the available license containers is available in the about dialog.

2.1 Dependencies

If you want to use any licenses provided on a WIBU license container, you are required to use ibaAnalyzer v8.0.0 or higher.

If you are using licensed features of ibaDatCoordinator v3.0.0 or higher, you are required to use ibaAnalyzer v8.0.0 or higher.

2.2 Free interactive usage of the data extractor

Starting from version 8 of ibaAnalyzer, the data extractor dialog, which was previously only available if a license was found, can be used for configuration and testing without license.

It is possible to:

- Open the dialog and configure File- and DB-Extract
- Test the extract with a single file / dataset

The option to extract multiple files from the filegroup and using the data extractor via command line or automated by ibaDatCoordinator is still subject to licensing.

2.3 New and discontinued products

2.3.1 ibaAnalyzer-DB and ibaAnalyzer-File-Extract are discontinued

The products ibaAnalyzer-DB and ibaAnalyzer-File-Extract are no longer available for ibaAnalyzer (see chapter 1.1). Existing licenses are further supported and no updates are required, even on MARX license dongles.

Instead, new products ibaDatCoordinator-DB, ibaDatCoordinator-File-Extract and ibaDatCoordinator-Publish are available, taking over the functionality of ibaAnalyzer-DB and ibaAnalyzer-File-Extract. See the New Feature document for ibaDatCoordinator v3.0.0 for details.

2.3.2 License behavior change for ibaAnalyzer-InSpectra

Starting from version 8.0.0 of ibaAnalyzer, the ibaAnalyzer-InSpectra views can be used without license. In order to get the result of the view as signals, the license ibaAnalyzer-InSpectra+ is required.

The concept is now the same as for ibaAnalyzer-InCycle+

3 New components for database connections (ibaAnalyzer-DB)

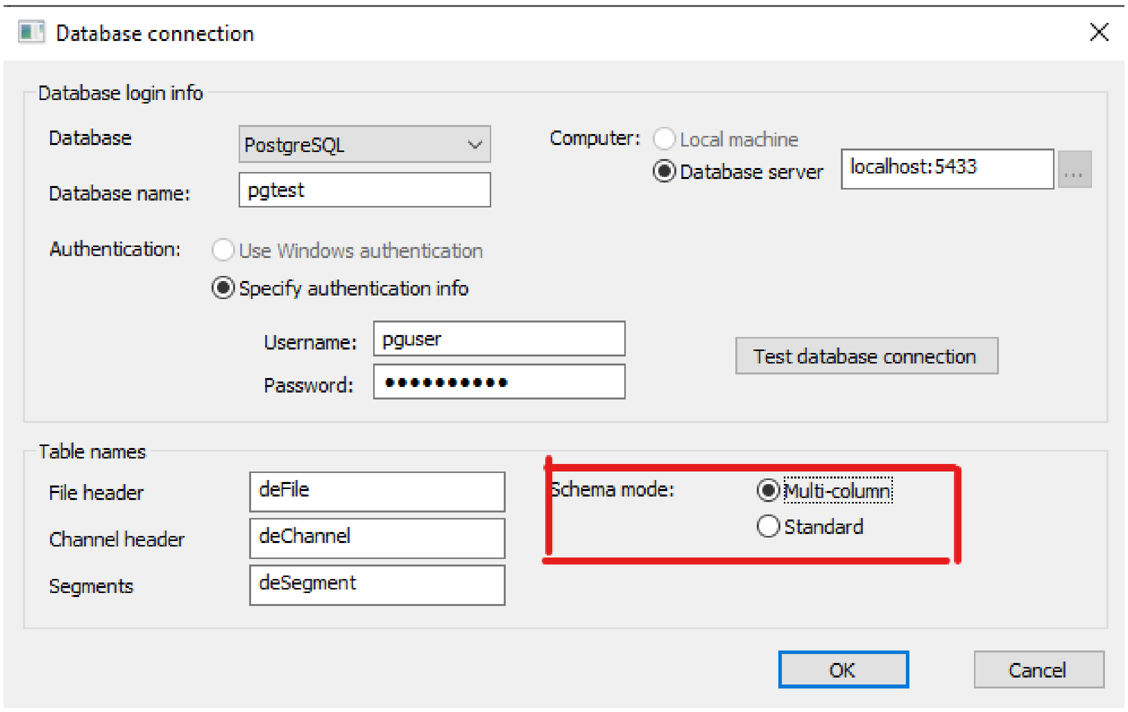
The SQL database connectivity in ibaAnalyzer was completely revised. From a user perspective the current state offers the same functionality as the previous version, however many components were changed behind the curtains. See chapter 3.6 for further technical details.

Relevant changes for the usage of the database components in ibaAnalyzer are described in the chapters below. Most notable changes are:

- ODBC connections are no longer supported (see chapter 3.5)
- The option “use transactions” has no effect in the current implementation (see chapter 3.6)

3.1 Schema selection

In earlier versions of ibaAnalyzer (<v8.0.0), DB-Extract was a separate component that could be registered independently of ibaAnalzyer. During installation, there was a choice between using a Standard or a Multi-Column library. The DB Extract is now considered as an integral part of ibaAnalzyer and switching between Standard and MC mode is no longer done during installation or by execution of a batch file. Now you can simply select the schema mode in the database connection dialog.

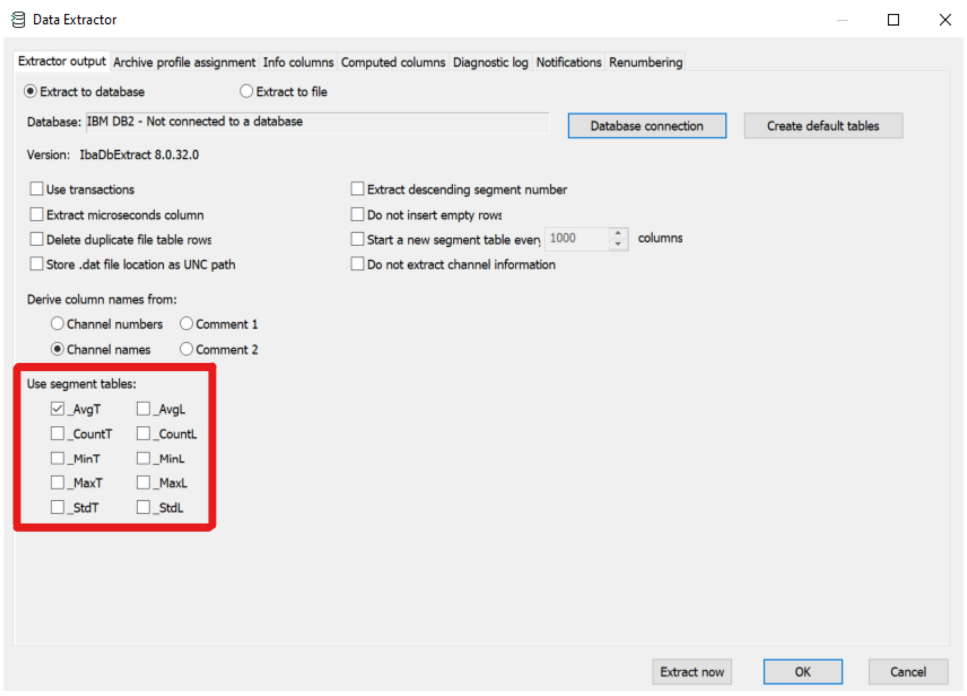

Database connection

Database login info
Database: PostgreSQL
Database name: pgtest
Computer: Local machine [✓ Database server] localhost:5433
Authentication: Use Windows authentication [✓ Specify authentication info]
Username: pguser
Password: ••••••••••

Table names
File header: deFile
Channel header: deChannel
Segments: deSegment
Schema mode: [✓ Multi-column] Standard

3.2 Segment tables

In earlier versions of ibaAnalyzer (<v8.0.0), it was possible to extract data to the channel table or leave this part out completely. Now it’s additionally possible to decide which segment tables will be created. The segment tables to be created must still be manually selected.


Data Extractor

Extractor output | Archive profile assignment | Info columns | Computed columns | Diagnostic log | Notifications | Renumbering

[✓ Extract to database] Extract to file

Database: IBM DB2 - Not connected to a database

Version: IbaDbExtract 8.0.32.0

Use transactions
Extract microseconds column
Delete duplicate file table rows
Store .dat file location as UNC path
Extract descending segment number
Do not insert empty rows
Start a new segment table every 1000 columns
Do not extract channel information

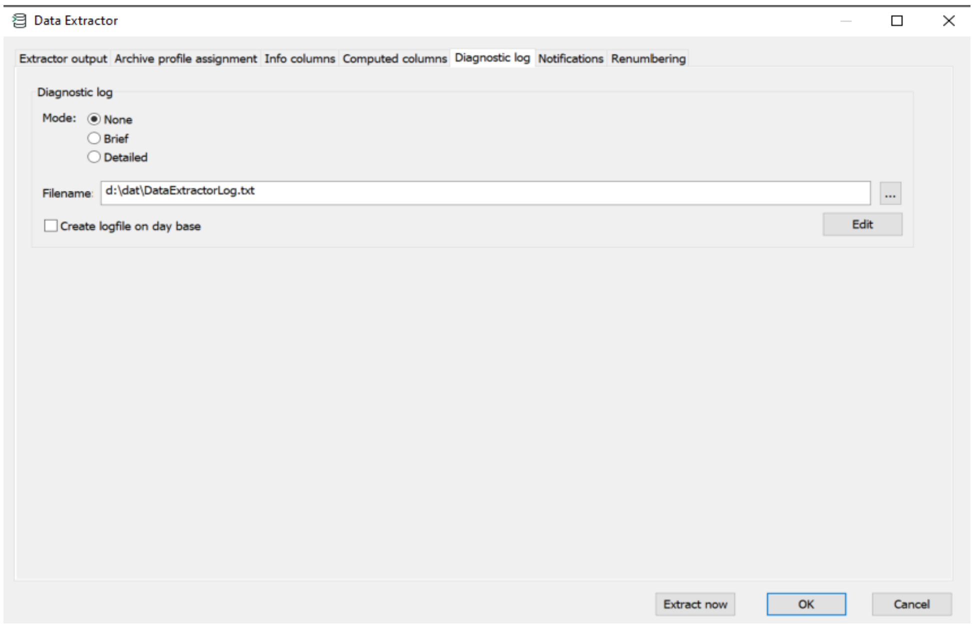
Derive column names from:
Channel numbers | Comment 1
[✓ Channel names] | Comment 2

Use segment tables:
[✓ _AvgT] _AvgL
_CountT _CountL
_MinT _MinL
_MaxT _MaxL
_StdT _StdL

When the standard schema mode is selected, the choice for the segment tables is not shown, because only one segment table will be created then.

3.3 Logging

Logging can be configured in *Diagnostic log* in the *Data Extractor* window. With the new version, the log files can optionally contain much more detailed information compared to previous versions.



Diagnostic log

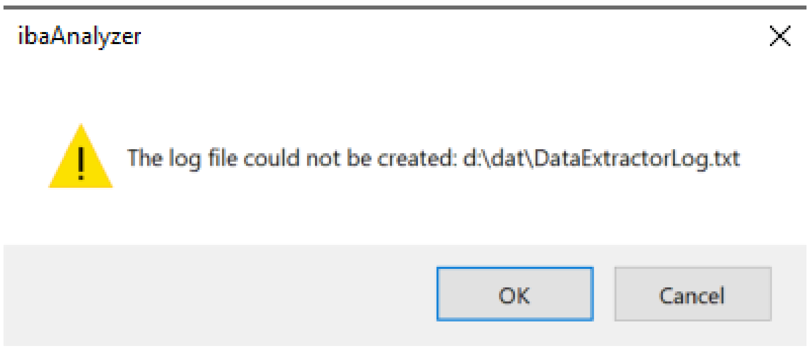
Mode: [✓ None] Brief Detailed

Filename: d:\dat\DataExtractorLog.txt

☐ Create logfile on day base

If the *Mode* is *None*, then no files will be created. *Brief* means that limited information will be written to the file and *Detailed* is for the maximum amount of information, including most SQL statements.

If the log file can't be created, you will be notified of that with a message box, like in the image below.



The log file could not be created: d:\dat\DataExtractorLog.txt

3.4 Metadata for the file table

A new table was introduced: the **file metadata table**. The file metadata table contains additional information (metadata) about values from computed columns in the **file table**. The name of the file metadata table is the name of the file table, to which “meta” is appended. For example, if the file table name is *deFile*, the file metadata table will automatically be *deFileMeta*.

The file metadata table is created when creating the default tables. It’s also created or updated when checking the computed fields. Currently, the file metadata table is created and filled along with the extraction to database, but no further actions are performed with the contained data.

In the file table, an additional field _MetaIdVersion is added, to refer to the version of the metadata that is used for each file. The file metadata table has the following fields:

- _MetaIdVersion: The version that records in the file table refer to. The current version is always the greatest _MetaIdVersion in the whole file metadata table.
- _ColumnName: The column name in the file table that this metadata record refers to. It should be possible to use the _ColumnName directly in an SQL statement.
- _FieldName: a friendly version of the column name, one that you would prefer to show to a user instead of the column name.
- _LogicalName: The name of the logical that is used to compute the value of the column in the referred column in the file table. _LogicalName can be NULL if no logical, but a plain expression was used for the calculation.
- _Expression: The expression that was used to calculate the value in the referred file table column
- _Unit: The unit that was added to the logical used to calculate the value of the referred column value in the file table.
- _Comment1 and _Comment2: comments that were added to the logical.

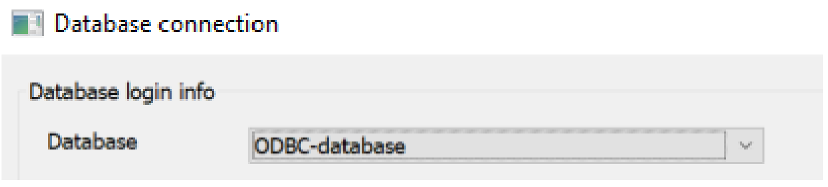
3.5 Changes for the connection configuration

The most important change for the ibaAnalyzer database components is with the connection settings.

ODBC configuration no longer supported

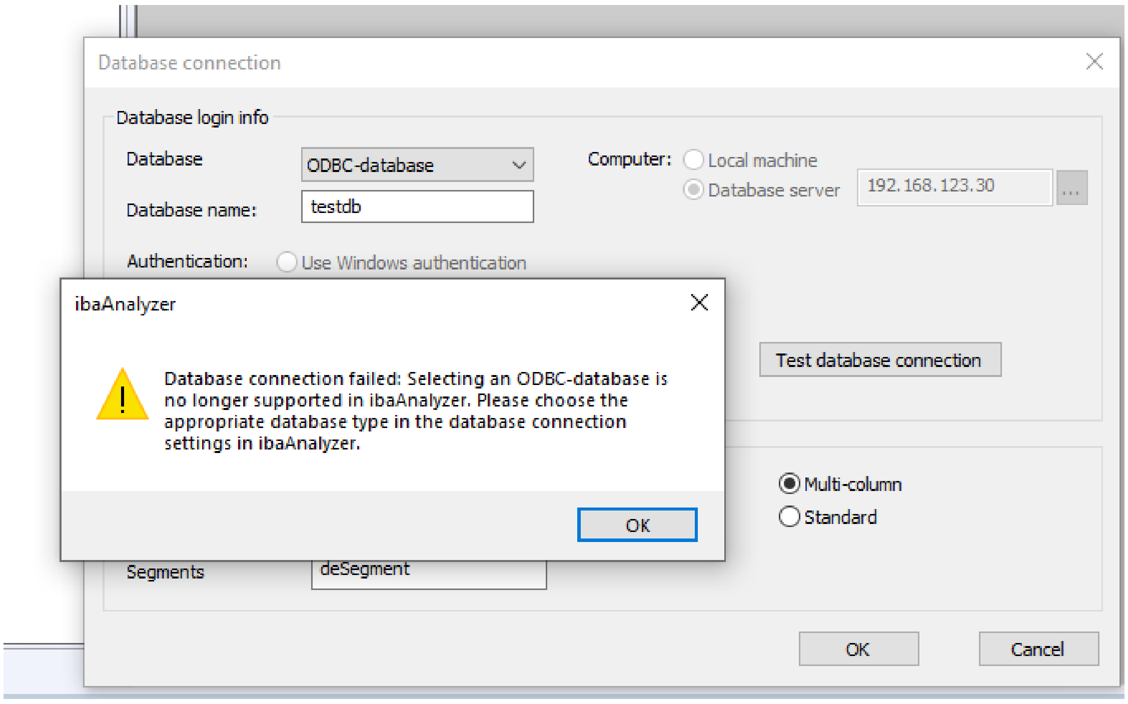
Since connections to ODBC data sources are no longer supported, any pdo containing an ODBC connection must be adapted to work with the new connectors.

Note that ibaAnalyzer doesn’t automatically convert the ODBC configuration to the new connection settings. It is still possible to select the database type *ODBC-database* to allow pdo files created with earlier versions of ibaAnalyzer to be loaded.



Database: ODBC-database

If you try to open a database connection configured with ODBC anyway, the following error message will be shown:



Database connection failed: Selecting an ODBC-database is no longer supported in ibaAnalyzer. Please choose the appropriate database type in the database connection settings in ibaAnalyzer.

Required steps and changes for the supported database types are summarized in the next chapters.

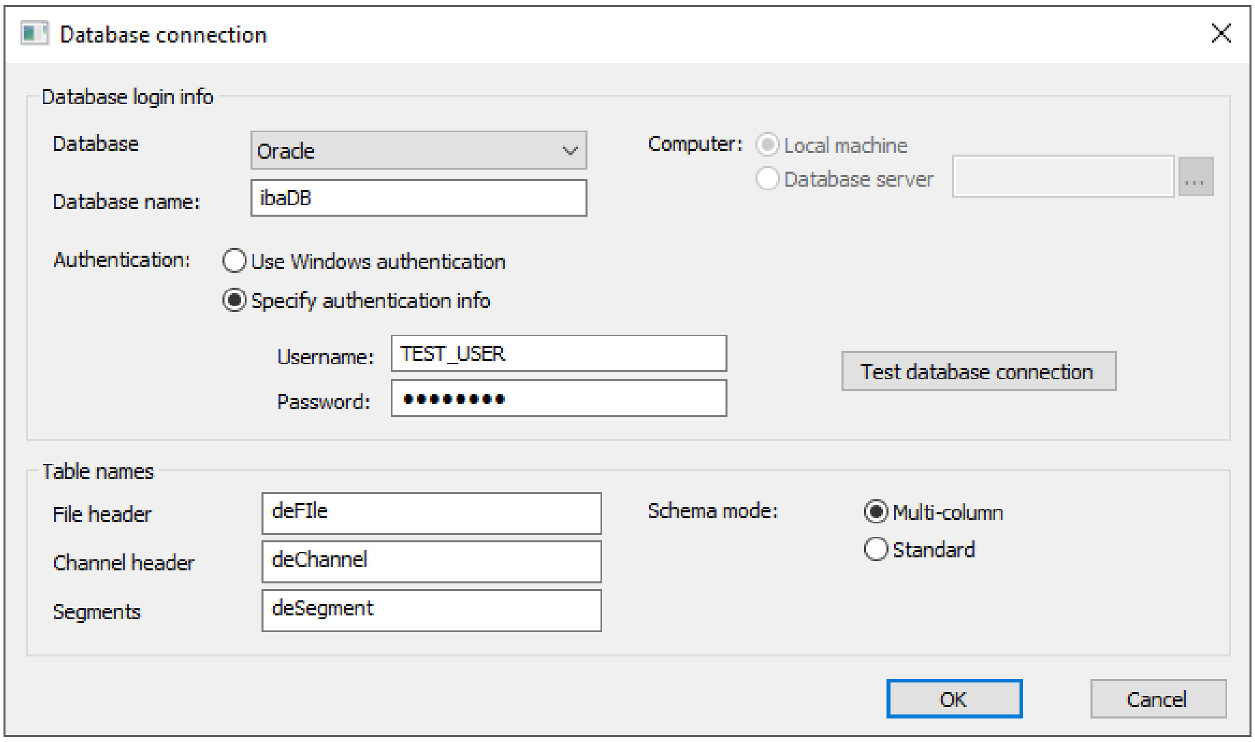
3.5.1 Oracle

The .NET data provider for Oracle is distributed with ibaAnalyzer. It's no longer required to install an Oracle client if you want to connect to Oracle with ibaAnalyzer. In earlier versions of ibaAnalyzer, an Oracle client installation with support for OleDb was required. That is no longer the case.

Connection settings

To connect to Oracle, *DB Extract* must be able to find a tnsnames.ora file. That file contains information about how to connect to an Oracle database by TNS name. The TNS name is what you fill in in the *Database* field.

Connecting with an ODBC DSN is no longer supported. All pdo files that use the *ODBC-connection* to connect to Oracle must be adapted, so that they directly use *Oracle* as the database type.


Database connection

Database login info

Database: Oracle

Database name: ibaDB

Computer: Local machine / Database server

Authentication: Use Windows authentication [✓ Specify authentication info]

Username: TEST_USER

Password: ••••••••

Test database connection

Table names

File header: deFIle

Channel header: deChannel

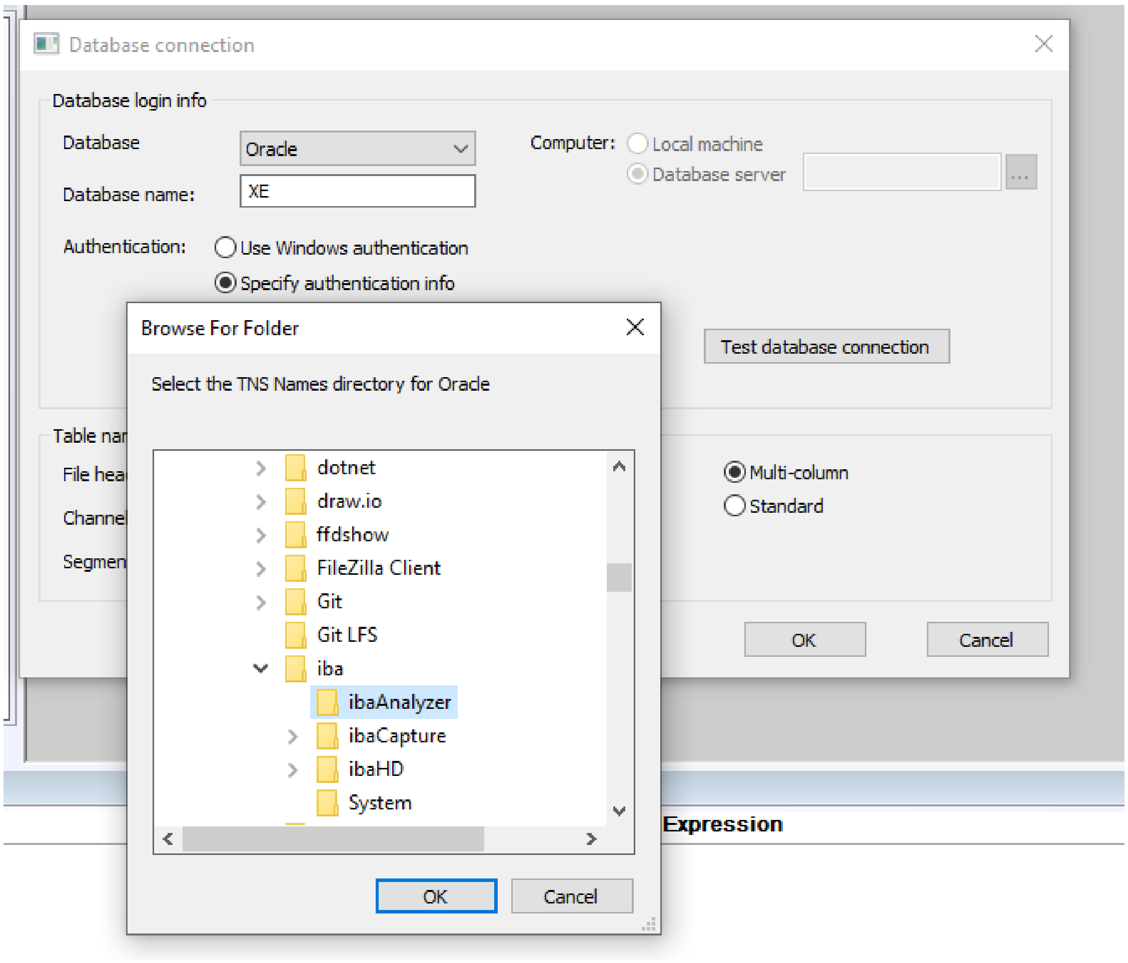
Segments: deSegment

Schema mode: [✓ Multi-column] Standard

OK Cancel

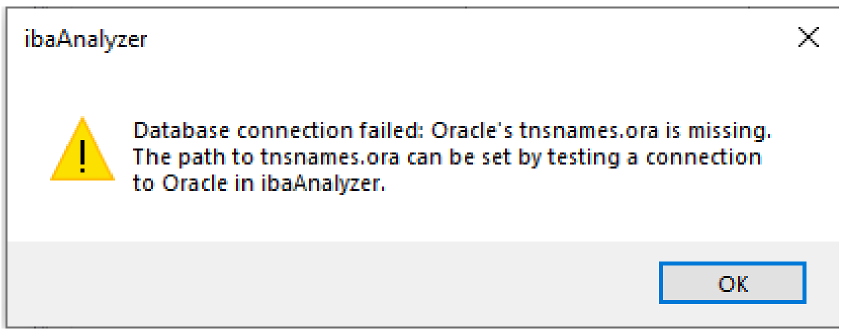
Tnsnames.ora

ibaAnalyzer automatically tries to find a file named tnsnames.ora. That file is required before a connection to Oracle can be made. If tnsnames.ora was not found, and ibaAnalyzer was started with the graphical user interface, a popup is shown. In the popup window, you can select the location of tnsnames.ora.



Your IT staff or DBA can provide you with a tnsnames.ora file that is valid for your installation within your organization. If you select the location of tnsnames.ora and DB Extract manages to use it, the location is saved in the registry. The registry is only written when ibaAnalyzer is used interactively.

On the other hand, if tnsnames.ora was not found, an error message will be produced. In the GUI, the error message is shown to the user. When invoked with ibaDatCoordinator, the error message is forwarded to ibaDatCoordinator.



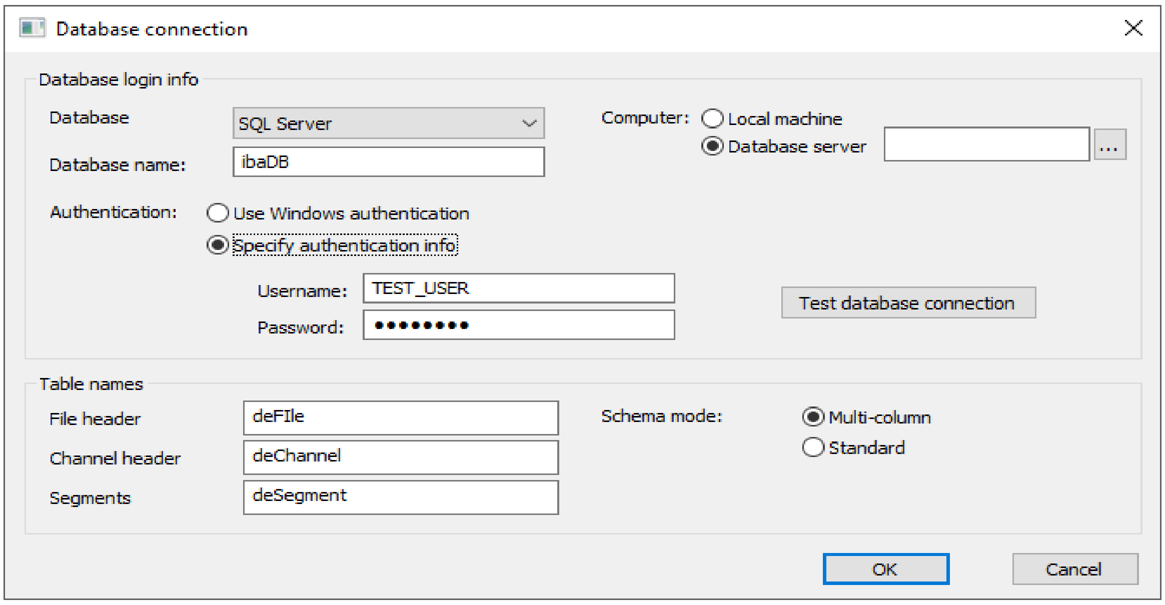
Database connection failed: Oracle's tnsnames.ora is missing. The path to tnsnames.ora can be set by testing a connection to Oracle in ibaAnalyzer.

3.5.2 Microsoft SQL Server

The SQL Server client is integrated in ibaAnalyzer

Connection settings

To connect to SQL Server, select the *SQL Server* database type. Connecting with an ODBC DSN is no longer supported. All pdo files that use the *ODBC-connection* to connect to SQL Server must be adapted, so that they directly use *SQL Server* as the database type.


Database connection

Database login info

Database	SQL Server
Database name:	ibaDB
Computer:	( ) Local machine [✓ Database server]
Authentication:	( ) Use Windows authentication [✓ Specify authentication info]
Username:	TEST_USER
Password:	••••••••

Table names

File header	deFIle
Channel header	deChannel
Segments	deSegment
Schema mode:	[✓ Multi-column] ( ) Standard

The (...) button can be used to search for SQL Server instances on the network.

Computer

The database server consists of parts that can be *computer name*, *DNS name*, *instance name*, *ip address* and *port number*. Multiple SQL Server instances may exist on the same computer. The default port for SQL Server is 1433. If the SQL Server port differs from that, it must be given explicitly in the *Database server* field.

Example:

192.168.111.222,1435 connects to a SQL Server instance on the computer with the given IP address, on TCP port 1435. Note that the host and the port are separated with a comma, which is unusual to address TCP sockets.

Database

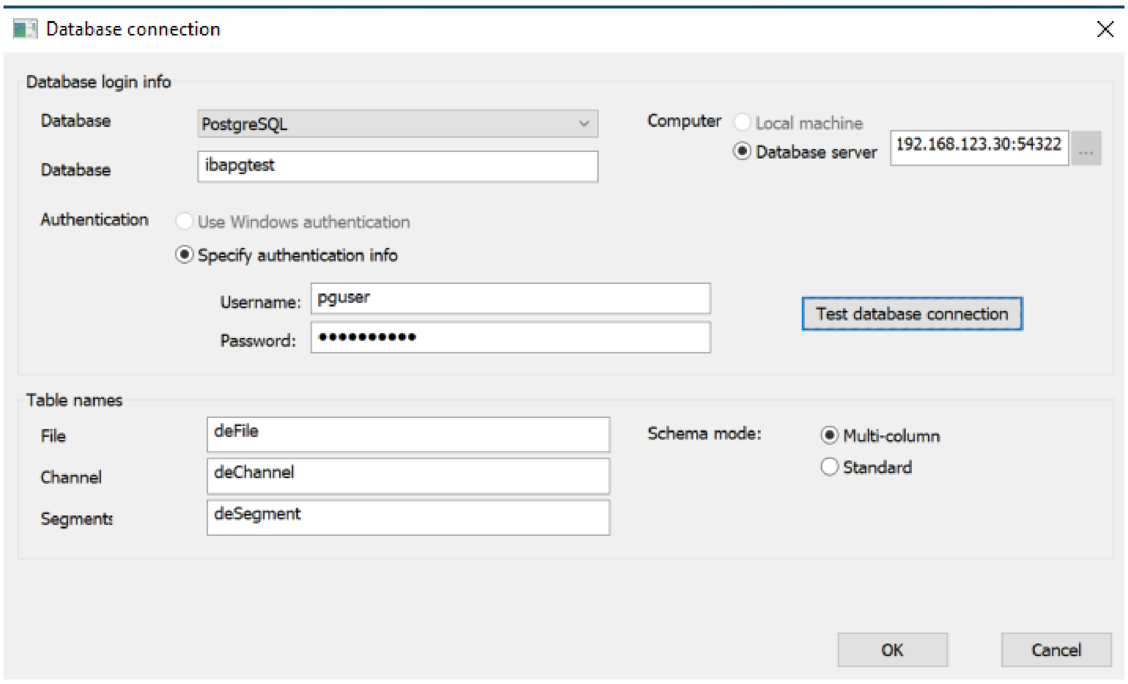
A SQL Server instance can contain multiple databases. Select the required database.

Authentication

Use Windows authentication or SQL Server authentication, depending on how you can get access to the SQL Server database.

3.5.3 PostgreSQL

To connect to PostreSQL Server, select the *PostgreSQL* database type. Connecting with an ODBC DSN is no longer supported. All pdo files that use the *ODBC-connection* to connect to PostgreSQL must be adapted, so that they directly use *PostgreSQL* as the database type.


Database connection

Database login info

Database: PostgreSQL

Database: ibapgtest

Computer: ( ) Local machine [✓ Database server] 192.168.123.30:54322

Authentication: ( ) Use Windows authentication [✓ Specify authentication info]

Username: pguser

Password: ●●●●●●●●●●●

Table names

File: deFile

Channel: deChannel

Segments: deSegment

Schema mode: [✓ Multi-column] ( ) Standard

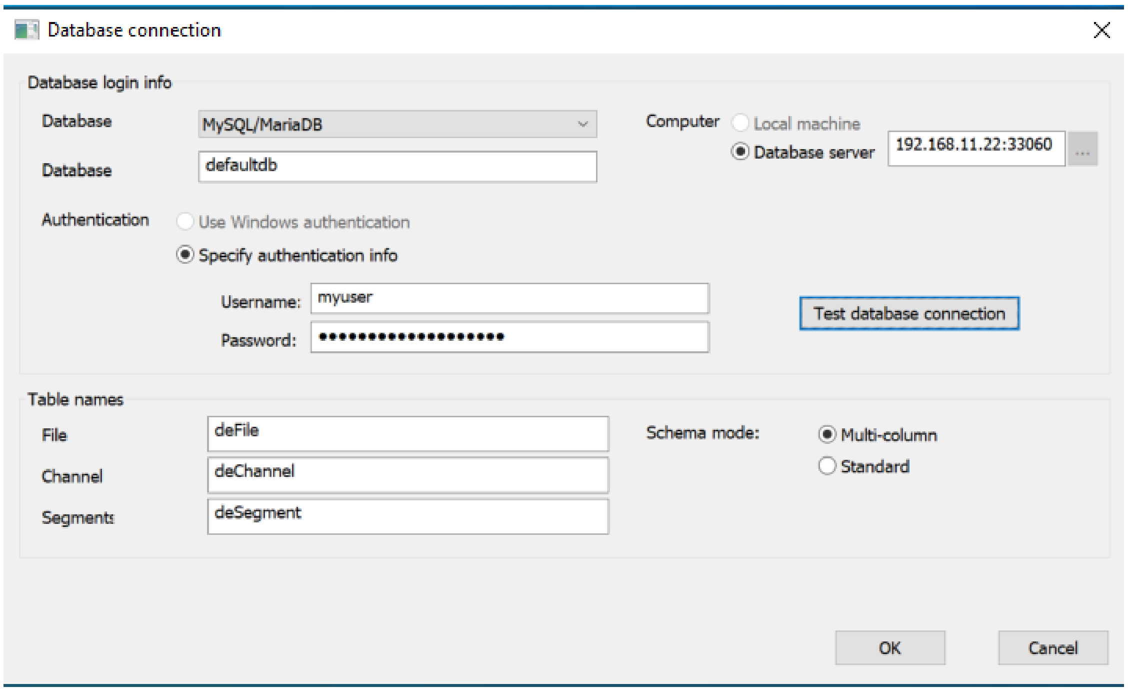
The IP address or host name of the PostgreSQL server must be filled in in the *Database server* field. For database servers on the local machine, you can use *localhost* or *127.0.0.1*. The default port number for PostgreSQL is 5432. If you want to connect to a PostgreSQL server that listens on a different port, append a colon (:) and then the port number to the host name or IP address. Examples:

- *localhost* for a local server on port 5432
- *192.168.111.222:54545* for a PostgreSQL server on IP address 192.168.111.222 on port 54545.
- *server.name* for a server named *server.name* on port 5432

A user name and password should be supplied. It’s not recommended to allow passwordless authentication on the PostgreSQL server.

3.5.4 MySQL/MariaDB

To connect to MySQL, select the *MySQL/MariaDB* database type. Connecting with an ODBC DSN is no longer supported. All pdo files that use the *ODBC-connection* to connect to MySQL or a compatible database must be adapted, so that they directly use *MySQL/MariaDB* as the database type.


Database connection

Database login info

Database	MySQL/MariaDB
Database	defaultdb
Computer	( ) Local machine [✓ Database server] 192.168.11.22:33060
Authentication	( ) Use Windows authentication [✓ Specify authentication info]
Username:	myuser
Password:	•••••••••••••••••••

Table names

File	deFile
Channel	deChannel
Segments	deSegment
Schema mode:	[✓ Multi-column] ( ) Standard

The IP address or host name of the MySQL/MariaDB server must be filled in in the *Database server* field. For database servers on the local machine, you can use *localhost* or *127.0.0.1*. The default port number for MySQL or compatible database types is 3306. If you want to connect to a MySQL server that listens on a different port, append a colon (:) and then the port number to the host name or IP address. Examples:

- *localhost* for a local server on port 3306
- *192.168.111.222:33036* for a PostgreSQL server on IP address 192.168.111.222 on port 33036.
- *server.name* for a server named *server.name* on port 3306

A user name and password should be supplied. It's not recommended to allow passwordless authentication on the MySQL server.

3.5.5 IBM DB2

To use IBM DB2, the *IBM Data Server Driver* must be installed.

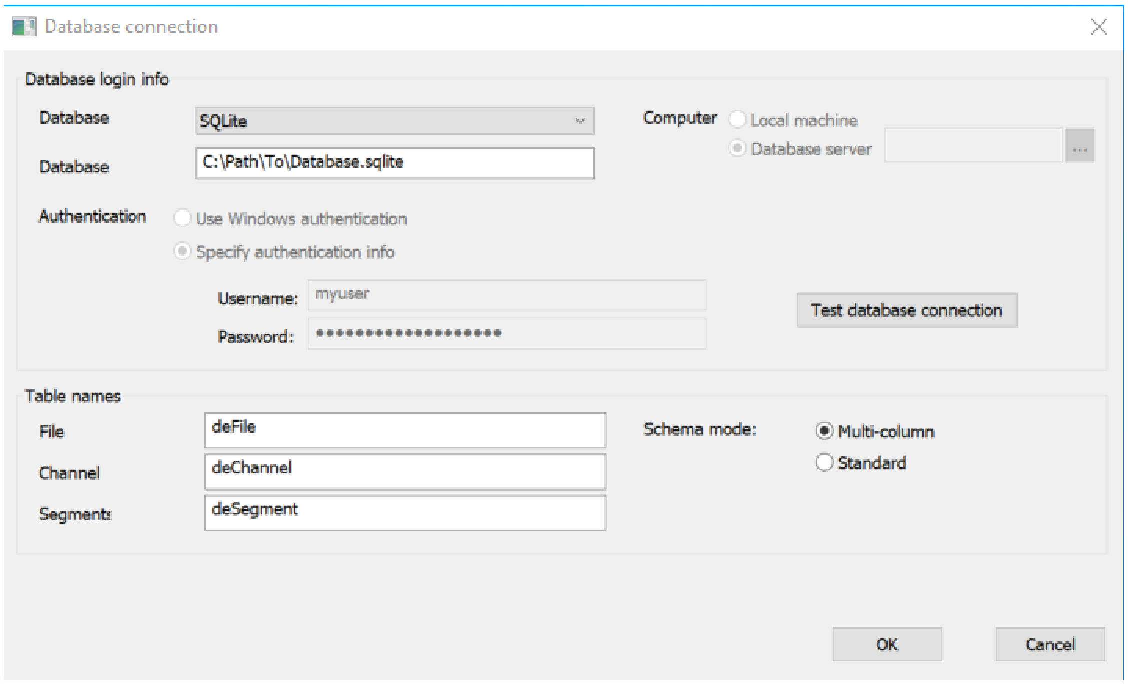
Version 11.5.7 is tested and works with ibaAnalyzer.

Username/password authentication is supported. Kerberos (Active Directory) authentication is untested and currently not supported.

Connecting with an ODBC DSN is no longer supported. All pdo files that use the *ODBC-connection* to connect to DB2 must be adapted, so that they directly use *DB2-UDB* as the database type.

3.5.6 SQLite

In-memory SQLite databases are not supported. ibaAnalzyer also doesn't create new SQLite databases. Connecting with an ODBC DSN is no longer supported. All pdo files that use the *ODBC-connection* to connect to SQLite must be adapted, so that they directly use *SQLite* as the database type.


Database connection

Database login info

Database: SQLite
Database: C:\Path\To\Database.sqlite
Computer: Local machine [✓ Database server]
Authentication: Use Windows authentication [✓ Specify authentication info]
Username: myuser
Password: •••••••••••••••••••

Test database connection

Table names

File: deFile
Channel: deChannel
Segments: deSegment
Schema mode: [✓ Multi-column] Standard

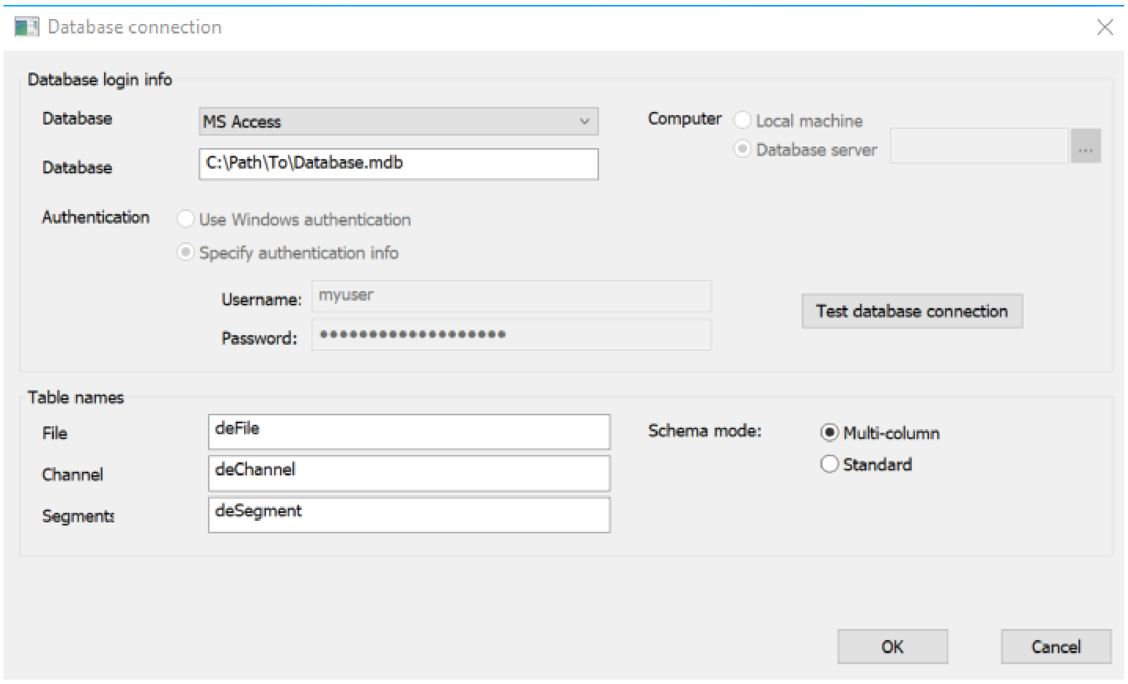
OK Cancel

In the *Database* text field, enter the full path to the SQLite database file. The database file must already exist.

3.5.7 MS Access

There is support for *mdb* files. *accdb* files are not supported currently.

Connecting with an ODBC DSN is no longer supported. All pdo files that use the *ODBC-connection* to connect to MS Access databases must be adapted, so that they directly use *MS Access* as the database type.



Database connection

Database login info

Database: MS Access
Database: C:\Path\To\Database.mdb
Computer: Local machine [✓ Database server]
Authentication: Use Windows authentication [✓ Specify authentication info]
Username: myuser
Password: •••••••••••••••••••

Table names

File: deFile
Channel: deChannel
Segments: deSegment
Schema mode: [✓ Multi-column] Standard

In the *Database* text field, enter the full path to the MS Access mdb file. The database file must already exist. IbaAnalzyer doesn’t create new MS Access (*.mdb) databases. MS Access database files can be created with the ODBC Data Sources tool (but the DSN can’t be used in ibaAnalyzer) that comes with Windows. Files with the *mdb* extension created with MS Access can also be used.

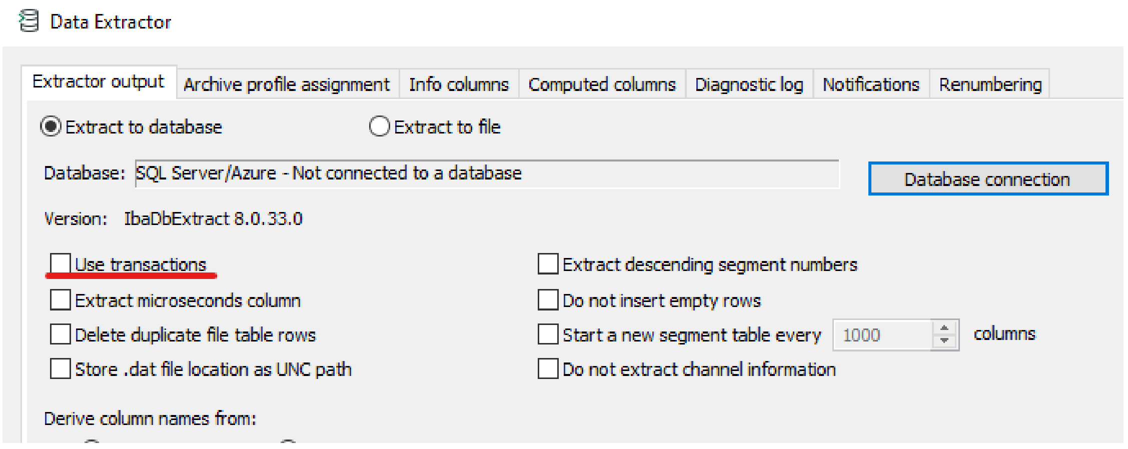
3.6 Further technical details

Single .NET library for Standard and MC mode

The implementation now contains a set of .NET dll files in the ibaAnalyzer installation. It runs in .NET Framework 4.8. This also means that error messages if a connection fails, on syntax errors, etc. may be different in v8 than they were in v7. Some error messages are produced by the underlying data providers, which used to be OleDB and are now ADO.NET. Behind the curtains, loading and saving an analysis in a database is now also done via *DB Extract*.

The registration of *DB Extract* is entirely handled by the installer and uninstaller. The new *DB Extract* doesn't interfere with any versions of DB-Extract that were supplied with earlier versions of ibaAnalyzer. It's no longer necessary to be able to register or unregister DB Extract.

DB Extract uses connection pooling, parallel processing and statement maximization like the DB/Cloud data stores do in ibaPDA. Especially due to parallel processing, transactional processing is not feasible in the current development status. Therefore, you may notice that the option *Use transactions* in the *Extractor output* tab of the *Data Extractor* dialog deliberately has no effect.



(● Extract to database) ○ Extract to file

Database: SQL Server/Azure - Not connected to a database

Version: IbaDbExtract 8.0.33.0

☐ Use transactions
☐ Extract microseconds column
☐ Delete duplicate file table rows
☐ Store .dat file location as UNC path
☐ Extract descending segment numbers
☐ Do not insert empty rows
☐ Start a new segment table every 1000 columns
☐ Do not extract channel information

Derive column names from:

Common code base: ibaSqlDialect

Multiple applications in the iba System use SQL functionality which was migrated to a common code base. The result is *ibaSqlDialect*, which is used in multiple applications:

- ibaPDA v8 and later:
  - SQL interface (input and output modules)
  - DB/Cloud data store
  - Eventing and Alarming (EVA)
- ibaAnalyzer v8 and later:
  - DB Extract
  - Save/Load PDO in DB

Logging for *DB Extract*

In .NET, logging is done by the ibaLogger component. If Extract to database is selected in Extractor output, then the .NET DB Extract component takes care of the logging. Otherwise, if Extract to file is active, the logging functionality is the same as it was before.

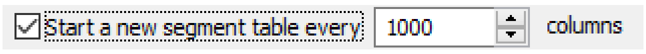
In fact, because both ibaAnalyzer itself and *DB Extract* use the configured log file location, two log files will be created. In the example in the image above that would be

- d:\dat\DataExtractorLog.txt for ibaAnalyzer
- d:\dat\DataExtractorLog-dotnet.txt for *DB Extract*

The log files are cleaned up on a regular basis. If *Extract to database* is active, also the log file for ibaAnalyzer will be cleaned up. A limited history of log files is kept in the same folder as the log files themselves. Log files that are cleaned up have a timestamp appended to their file name. Older log files are zipped. Eventually old zip files are deleted.

Existing tables during multi-column extract

During multi-column extract, an MC segment table can already exist in the database and have a number of columns. If one or more of those columns will not be used in the next extract, the columns are skipped, but they still count for the maximum number of columns per table. Previously, only the columns which were actually written were taken into account.


☑ Start a new segment table every 1000 columns

That means that the extract can start a new segment table sooner if there are existing unused columns in the table. Therefore, the same PDO can have different outcomes, a channel can be in a different table depending on the already existing data in the database tables.

Column limit is handled during multi-column extract

Each database type has a maximum number of columns that can be added to a table.

It’s possible to configure segment tables that are wider than the column limit. In earlier versions of ibaAnalyzer, that would result in an error and the extract would be aborted. Since ibaAnalyzer v8, the column limit is taken into account when creating or altering segment tables. The tables never have more columns than the database system allows. If the column limit is exceeded, a new segment table is created.

For example, in DB2 the maximum number of columns per table is 500. In earlier versions of ibaAnalyzer, setting the number of columns in the MC segment table to a higher number resulted in an error during extract. That error is avoided in ibaAnalyzer v8.

Sensitive data in registry

Some sensitive data were stored in the registry after ibaAnalyzer was closed. That is now resolved and sensitive data is encrypted.

Connect to database only when needed

Database connections are only made as soon as they are necessary. In earlier versions, it could take a while to open the *Data extractor* window if the database connection was unavailable.

Date/time precision

Most databases support a timestamp or a datetime type with microsecond precision. Most of the .NET data providers that are used by ibaSqlDialect also support these data types. The additional microseconds column is therefore in many cases no longer necessary.

MS Access supports only up to 1ms precision for mdb files. The most recent version of MS Access also supports up to nanosecond precision, but that is not tested and not supported in ibaSqlDialect.

4 Support for additional external file formats (ibaAnalyzer-E-Dat)

The interface ibaAnalyzer-E-Dat for reading non dat-file formats has been extended by several file formats. These are very specific file formats and were supported by the deprecated product ibaRotate.

The following formats can be read now:

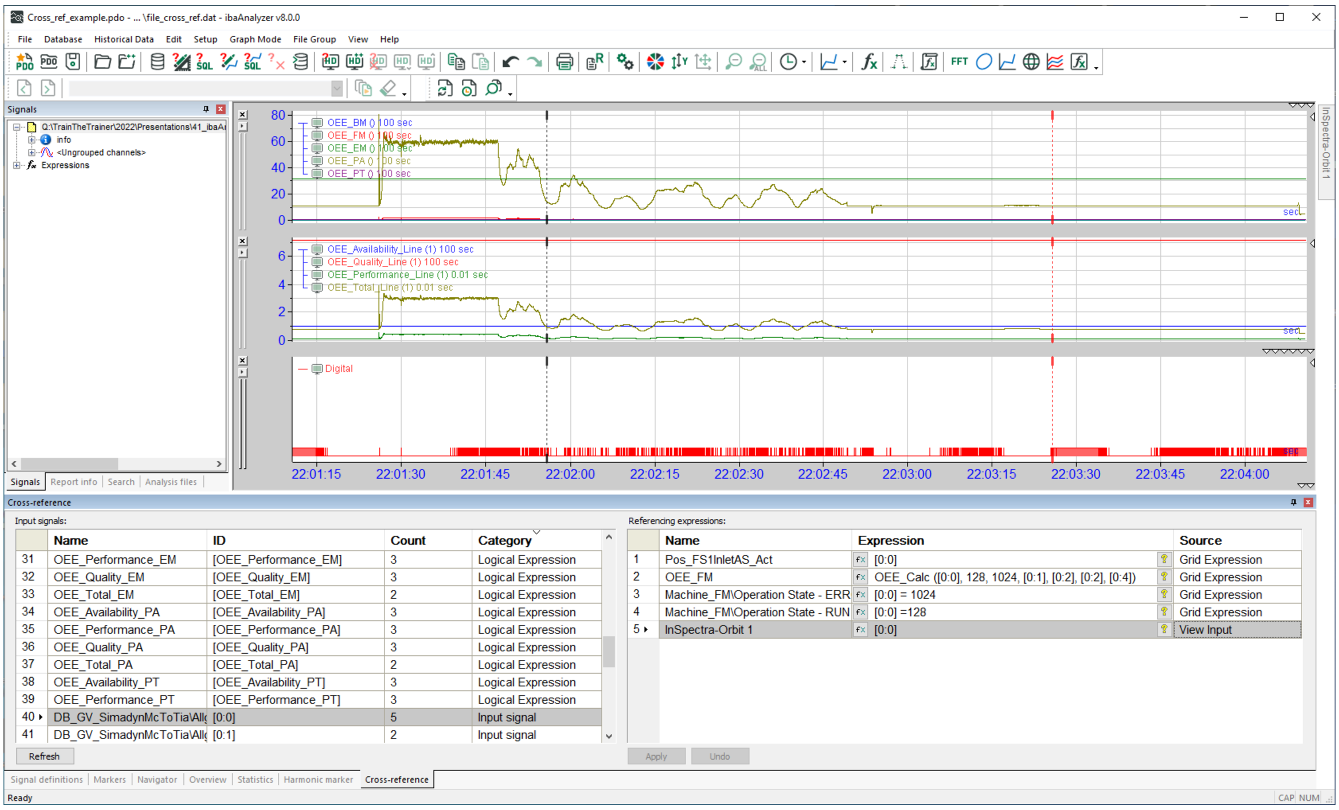
- Standard data format files (*.dat)
- Universal 58 binary (*.bunv, *.uff) and text (*.unv) files
- Vold Trace (*.tra), Vold Torsion (*.tor), and Vold Spline (*.spl) files

5 Cross-reference pane

In the current version of ibaAnalyzer, a new function is added to resolve cross references of signals and expressions. A new cross-reference pane is available for this. It presents the user with a way to view which inputs are in use by the currently opened analysis. The pane is by default docked in the bottom tab window as the last tab, after the “Overview” pane and is labeled “Cross-reference”.

The pane consists out of the following elements:

- On the left, a grid which lists the input signals in use, labeled “Input Signals”. This grid is initially empty.
- A button labeled “Refresh” below the input signals grid. Clicking this button will list the currently used inputs in the input signals grid.
- On the right, a grid which lists the expressions that contain a reference to the selected input signal in the left grid, labeled “Referencing Expressions”. It will list all expressions that were referencing the input signals at the moment the “Refresh” button was clicked. The expressions in this grid are editable.
- An “Apply” button below the expressions grid which will apply any changes you made to the referencing expressions. With this, it is not required to go to the actual dialogs or panes where the expressions are defined.
- An “Undo” button which will undo any changes you made to the referencing expressions.


Cross_ref_example.pdo - ...\file_cross_ref.dat - ibaAnalyzer v8.0.0

Cross-reference

Input signals:

	Name	ID	Count	Category
31	OEE_Performance_EM	[OEE_Performance_EM]	3	Logical Expression
32	OEE_Quality_EM	[OEE_Quality_EM]	3	Logical Expression
33	OEE_Total_EM	[OEE_Total_EM]	2	Logical Expression
34	OEE_Availability_PA	[OEE_Availability_PA]	3	Logical Expression
35	OEE_Performance_PA	[OEE_Performance_PA]	3	Logical Expression
36	OEE_Quality_PA	[OEE_Quality_PA]	3	Logical Expression
37	OEE_Total_PA	[OEE_Total_PA]	2	Logical Expression
38	OEE_Availability_PT	[OEE_Availability_PT]	3	Logical Expression
39	OEE_Performance_PT	[OEE_Performance_PT]	3	Logical Expression
40 ▸	DB_GV_SimadynMcToTia\All	[0:0]	5	Input signal
41	DB_GV_SimadynMcToTia\All	[0:1]	2	Input signal

Referencing expressions:

	Name	Expression	Source
1	Pos_FS1InletAS_Act	[0:0]	Grid Expression
2	OEE_FM	OEE_Calc ([0:0], 128, 1024, [0:1], [0:2], [0:2], [0:4])	Grid Expression
3	Machine_FM\Operation State - ERR	[0:0] = 1024	Grid Expression
4	Machine_FM\Operation State - RUN	[0:0] =128	Grid Expression
5 ▸	InSpectra-Orbit 1	[0:0]	View Input

5.1 Input signals grid

The input signal grid has the following columns

Name

The name of the input signal

ID

The ID that is used in expressions when referencing this input.

Count

The number of expressions that contain a reference to this input. This corresponds to the number of rows in the referencing expression grids when the row for the input is selected in the input signal grid.

Category

The type of input. The currently supported categories are:

- *Input signal:* This is an input signal from one of the currently loaded input files, usually a .dat file. This can also be a pseudo-dat file as generated from a database or HD query. Only the input signals which are actually referenced are listed; unreferenced input signals are omitted.

- *Logical Expression:* This is a logical expression that is defined in the logical expression dialog. By default, all logical expressions are reported. Logical expressions have a Count of 0 when they are unreferenced. When they are unreferenced, the row will be displayed in orange. Indicating that the analysis can be optimized by removing the unused logical.

  The row will be depicted in red when the logical is a “duplicate”, which means there is an input signal or logical with the same ID as the logical expression. Since the input signal with the same ID cannot be referenced this way, this is likely a mistake in the analysis that needs to be corrected. Likewise, if there is a logical with the same name, only one of the logical expressions can actually be referenced and this is likely a mistake in the analysis that needs to be corrected.
- *Grid Expression:* This is an expression that is defined in the signal grid. A grid expression will only be present in the input signal grid if it is referenced in another expression (i.e. the Count column is greater than zero) or if it’s a duplicate (i.e. there is another logical or expression with the same name so that this expression is obscured and can’t be referenced). In the latter case the row will appear in red and will have a Count of zero in that column. This is likely a mistake in the analysis that needs to be corrected.

  A Grid Expression will not be listed if it’s a direct reference to an input signal, as this means it is likely just a visualized signal rather than a calculation.
- *Trend query:* channel returned from a database trend query (and placed in the signal tree rather than the overview).
- *View Result:* a result channel from either ibaInSpectra, ibaInCycle or the new Computation module. Only result channels that are actually referenced are listed; unreferenced input signals are omitted.

The grid can be sorted on any column alphabetically by clicking the column header, clicking the column header a second time will sort the rows in reverse order. A little arrow appears in the column on which the grid is sorted, indicating the sort direction.

When double clicking on a row in the input signals grid, ibaAnalyzer will try to navigate to the relevant pane where the input is defined:

- For the categories “Input Signal”, “Trend query” or “View Result”, the signal will be selected in the signal tree, the signal tree will be made visible if it was tabbed away or auto-hidden. If the input signal is directly referenced in the signal grid the row in the signal grid will be selected as well.
- For the category “Grid Expression”, the row will be selected in the signal grid, the signal grid will be made visible if it was tabbed away or auto-hidden.
- For the category “Logical”, the logical expressions dialog will be opened and the specific logical will be selected in this dialog.

5.2 Referencing expressions grid

The grid will list the expressions that reference the input specified in the selected row in the input signals grid. When double clicking on a row in the referencing expressions grid, ibaAnalyzer will try to navigate to the relevant pane where the input is defined like for the input signal grid.

The referencing expressions has the following columns

Name

The name the expression is identified with.

Expression

The expression itself, which can be edited and hence the usual expression cell controls (diagnose and expression builder buttons) are present.

Source

This is an indicator where the expression is defined. the following sources are currently available:

- *Grid Expression*: This source indicates an expression defined in the signal grid. When navigating to this source, the row defining the expression will be selected in the signal grid, the signal grid will be made visible if it was tabbed away or auto-hidden.
- *Logical Component:* This source indicates a logical or one of its components if it is a multidimensional logical. Navigating to this source will open the logical expressions dialog and select the logical and the row corresponding with the component in case of a multidimensional logical.
- *X-axis Marker*: This source indicates a computed static marker. Navigating to this source will open the markers dialog and select the row containing the marker. The visibility settings of the dialog will be for the first graph that has this marker visible.
- *X-axis Interval*: This source indicates an interval. Navigating to this source will open the intervals dialog and select the row containing the interval.
- *Report Computed Value*: This source indicates a computed column in the report generator. Navigating to this source will open the report dialog, select the computed columns tab and select the field in the grid.
- *Extract Computed Value:* This source indicates a computed column in the extractor. Navigating to this source will open the extractor dialog, select the computed columns tab and select the field in the grid.
- *X-Axis Manual Scale*: This source indicates one of the expressions for an X-Axis manual scale. Navigating to this source will open the settings dialog for the X-Axis in question.
- *Y-Axis Manual Scale:* This source indicates one of the expressions for a Y-Axis manual scale. Navigating to this source will open the settings dialog for the Y-Axis in question.
- *View input:* This source indicates that the input selected in the input signal grid is used as a signal or value by one of the Views (Maps, ibaInSpectra, ibaInCycle, Computation

Module). Navigating to this source will focus the view and make it visible if it was tabbed away or auto-hidden.

Note that the “Source” field might be blank in some cases as not all possible expression sources of ibaAnalyzer are currently implemented. Additional sources will be implemented step by step or on request.

6 Group structure and renaming for logical expressions

In the current version of ibaAnalyzer, it is possible to structure the logical expressions into groups within the logical expression dialog. A hierarchical group structure (groups containing groups) is supported. Each logical expression can belong to one and only one group but can have several parent groups.

This feature is entirely forwards and backwards compatible with other versions of ibaAnalyzer, as the groups are implicitly defined in the full name of the logical expression. The groups and subgroups are prepended to the logical’s short name and separated by a backslash (‘\’).

Consequently, any logical expression in an existing analysis containing a backslash will be shown in an according group. When opening a pdo containing grouped logical expressions in an older version of ibaAnalyzer, the expressions will be shown in a flat list containing the group names prepended with backslashes.

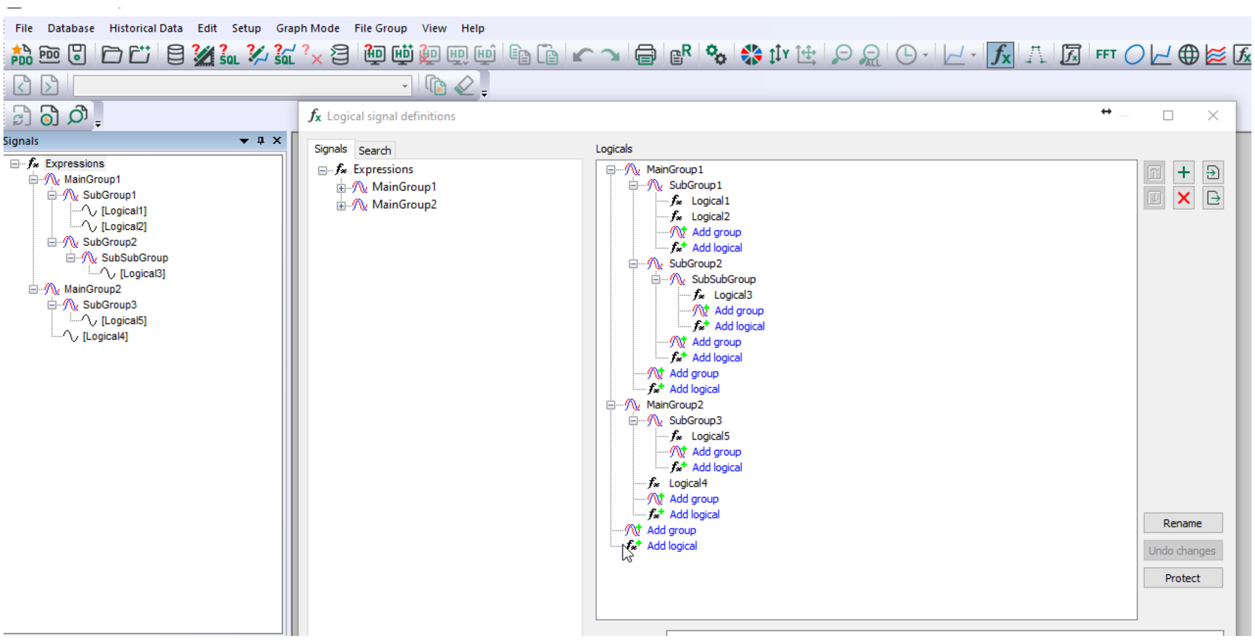
Example

The set of logical expressions with the following names:

- [MainGroup1\SubGroup1\Logical1]
- [MainGroup1\SubGroup1\Logical2]
- [MainGroup1\SubGroup2\SubSubGroup\Logical3]
- [MainGroup2\Logical4]
- [MainGroup2\SubGroup3\Logical5]

Corresponds with the following group/logical structure:

- MainGroup1
    - SubGroup1
        - Logical1
        - Logical2
    - SubGroup2
        - SubSubGroup
            - Logical3
- MainGroup2
    - SubGroup3
        - Logical5
    - Logical4



There are 2 ways to define groups:

- Create a logical and rename it with the parent groups prepending its new name, separated by backslashes. Note that if the logical itself is already in a group, the parent groups do not need to be repeated, i.e. the new name should be specified relative to the existing group.
- Click on the blue link labeled "Add group" after which you have an option to name the group, you can also specify subgroups immediately here by typing backslashes in the name of the group.

Logical expressions can be created directly in an existing group by clicking the blue link labeled "Add logical" as well by clicking the green "+" button while a group is selected in the tree structure. Logical expressions and also complete groups can be reassigned to other groups by drag&drop. Multi-select is possible.

Groups and logical expressions can be deleted by clicking the red "X" button. If a group is selected in the tree, this will mark the entire group for deletion recursively, i.e., all subgroups and logical expressions contained in the group or any subgroup will be marked for deletion.

The group structure will be respected when the logical expressions are depicted in the signal tree. In referencing expressions, the logical expressions need to be referred by their full name, i.e. with the parent groups prepended and separated by backslashes.

6.1 Group structure for report chart fields

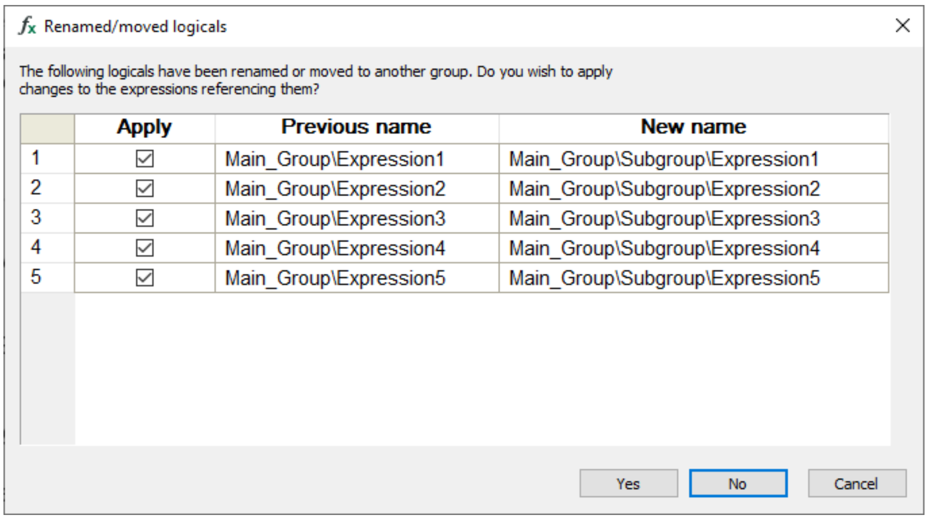
Grouping with basically the same functionality as above is also available for chart fields in the report generator. Since the report designer usually uses a dot character ('.') to build a group structure, ibaAnalyzer will convert the backslashes to dots before presenting them in the report designer

6.2 Renaming logical expressions

When renaming a logical expression in ibaAnalyzer, a new dialog will be shown with the possibility to adapt all referencing expressions. This is especially useful when moving expressions to another group or when renaming groups.

With this it is possible to automatically adapt all expressions that referred to the old name or grouping structure so that they refer to the new name/grouping and stay valid.

The dialog is automatically shown when you exit the logical expressions dialog while having renamed or regrouped logical expressions. In the dialog you can select the logical expressions for which you want all referring expressions adapted.



The following logicals have been renamed or moved to another group. Do you wish to apply changes to the expressions referencing them?

	Apply	Previous name	New name
1	☑	Main_Group\Expression1	Main_Group\Subgroup\Expression1
2	☑	Main_Group\Expression2	Main_Group\Subgroup\Expression2
3	☑	Main_Group\Expression3	Main_Group\Subgroup\Expression3
4	☑	Main_Group\Expression4	Main_Group\Subgroup\Expression4
5	☑	Main_Group\Expression5	Main_Group\Subgroup\Expression5

In a grid the previous name and new name are listed along with a check box. By default, all renamed or regrouped logical expressions are checked; you can uncheck the ones for which you do not wish the expressions to be adapted.

You can confirm the dialog with

- *Yes*: all referencing expressions of the selected logical expressions will be adapted
- *No*: no expressions will be adapted.
- *Cancel*: return to the logical expressions dialog without any changes having been applied.

7 Simplifying computed columns

To avoid redundancy and errors, it is good practice to specify all computations which are not for visualization only in the logical expressions dialog and to refer to these logical expressions in other places like the computed columns tabs in the report generator and data extractor dialogs.

To assist such a workflow and also to easily adapt existing pdo's not sticking to this practice, a button is added to the computed columns that will create the necessary logical expressions and place the references to these in the computed columns grid.

The created logical expressions will be added to a predefined group (“Extract computed column” or “Report computed column”) automatically. You can resort expressions from these groups or re-name them easily in the logical expressions dialog.

When using this function, several scenarios can happen for each computed column:

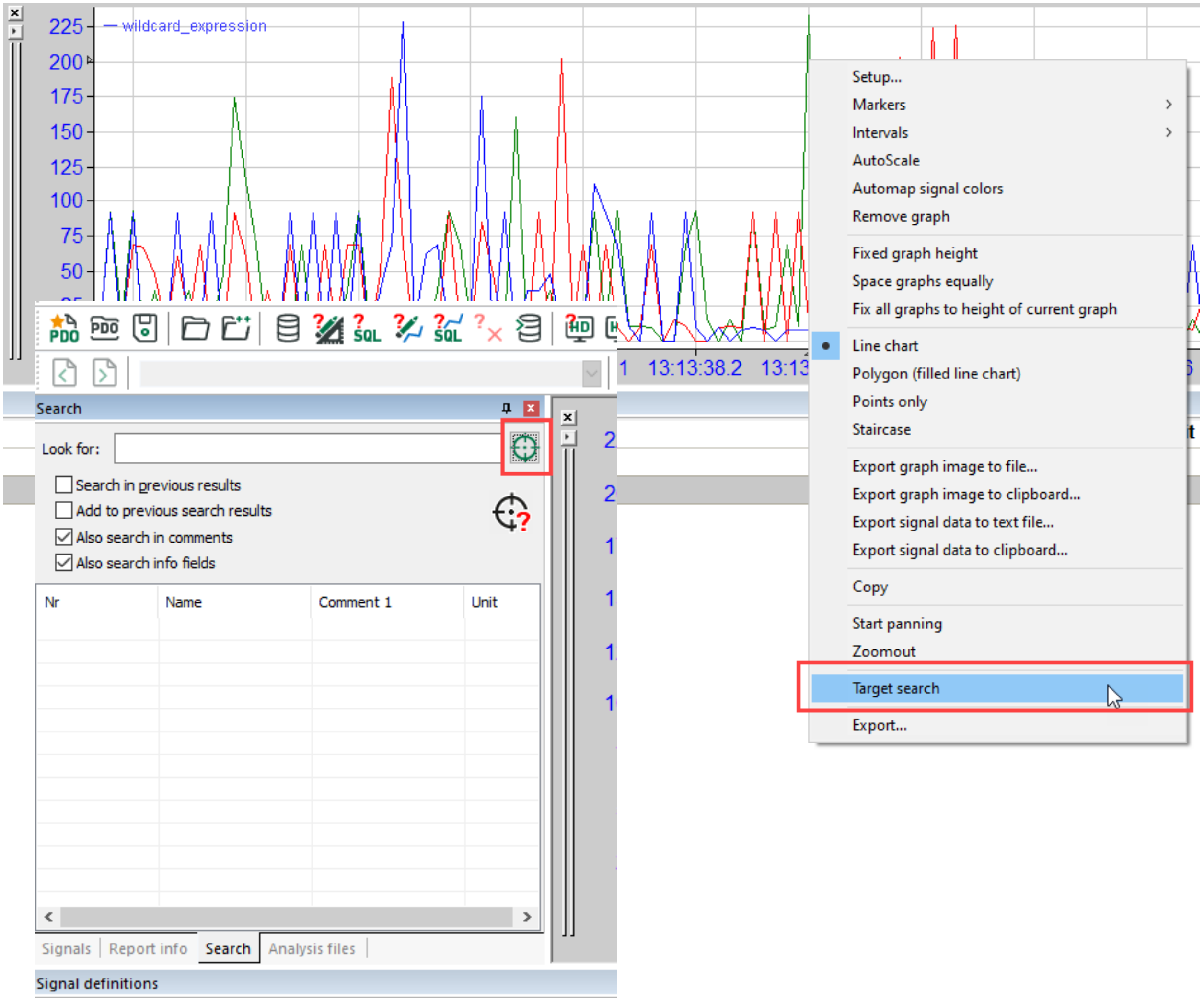
- If the computed column is a direct reference to a signal or logical expression, the computed column is left unaltered.
- If the computed column has a name and expression which both cannot be found in the logical expressions, the logical gets created and the computed field gets adapted to refer to the new logical without any user interaction.
- If the computed column has the **same name and expression** as an existing logical, the computed column gets adapted to refer to the logical without any user interaction.
- If the computed column has the **same name but different expression** than an existing logical, a dialog to choose between the following options is opened:
    - Ignore the existing logical and create a new logical with the computed column expression and adapt the computed column to refer to this logical.
    - Adapt the existing logical with the expression in the computed column and adapt the computed column to refer to the existing logical. (This is the default selected option and will be selected when the user dismisses the dialog with OK)
    - The computed column can also be left unaltered by dismissing the dialog with Cancel or closing it.
- If the computed column has a **different name but same expression** compared to an existing logical, a dialog to choose between the following options is opened:
    - Adapt the computed column to refer to this logical. (This is the default selected option and will be selected when the user dismisses the dialog with OK)
    - Ignore the existing logical and create a new logical with the computed column expression and adapt the computed column to refer to this logical.
    - Remove the existing logical and create a new logical with the computed column expression and adapt the computed column to refer to this logical.
    - The computed column can also be left unaltered by dismissing the dialog with Cancel or closing it.
- If the computed column is a direct reference to a signal grid expression, a dialog to choose between the following options is opened:
    - Create a logical and adapt both the computed column and grid expression to refer to this logical. (This is the default selected option and will be selected when the user dismisses the dialog with OK)
    - Create a logical but only adapt the computed column, the grid expression is left unaltered.
    - The computed column can also be left unaltered by dismissing the dialog with Cancel or closing it.

In each of the dialogs it is possible to check the option "Apply for all similar computed columns" which will suppress the dialog for all other computed columns belonging to this scenario. The selected choice is applied to all automatically.

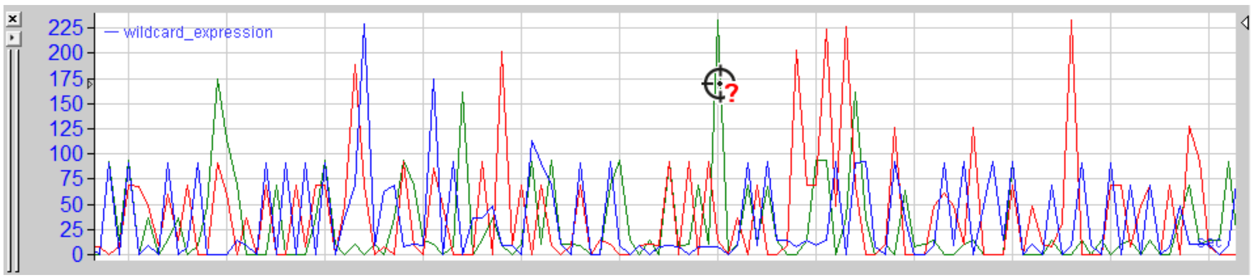
8 Search signals by targeting in the graph

When showing multiple signals in one graph it can be cumbersome (or even impossible for vectors or wildcard expressions) to find out which signal is which. Especially when searching for outliers, this can be very useful.

It is possible to find the signal ID or name by clicking on the signal in the graph. This function needs to be enabled either by a button in the search dialog or from the context menu of the graph.



Activating this option will change the cursor to a target icon. You can now click near any signal in a graph in the signal strip to identify the signal that has a sample depicted closest to the pixel clicked. The result will be reported in the search dialog.

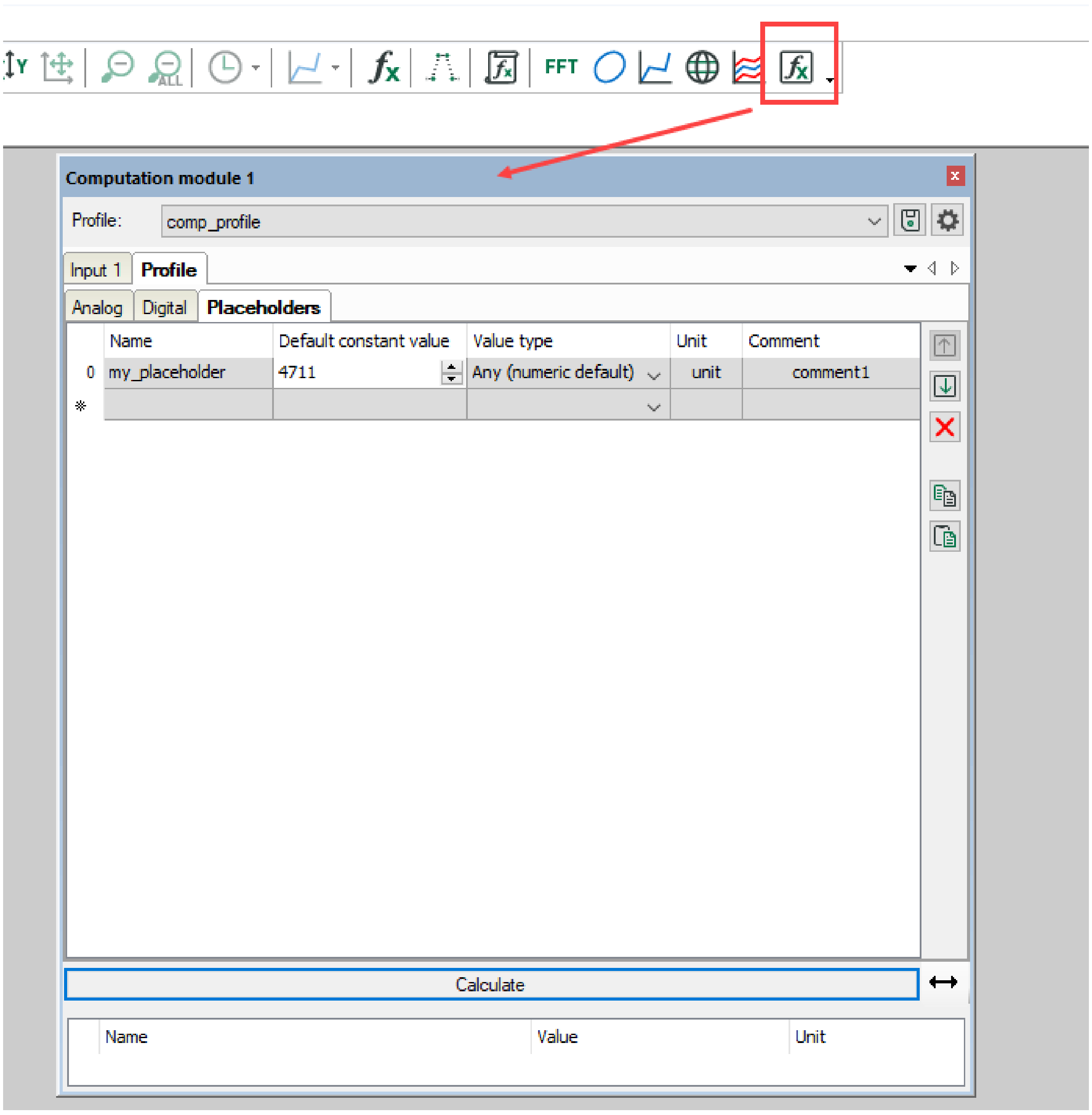


Next to the entry in the search grid, the signal is identified in other ways, if applicable.

- The corresponding expression is also selected in the signal grid.
- If the selected signal is a direct input channel reference, the corresponding measurement file is highlighted in the signal tree
  Note that this is only useful if the signal tree is made visible, either by selecting the tab or have the signal tree undocked or tabbed in another window than the search dialog.
- If the target signal is part of a multidimensional logical or vector, an appropriate expression (using the “GetRows” function) is generated and returned as the search result in the search dialog, allowing you to double click it and add the signal as separate expression to the expression grid.

9 Integration of the computation module from ibaPDA

The computation module known from ibaPDA is now available as additional view in ibaAnalyzer



Computation module 1

Profile: comp_profile

Input 1 | Profile

Analog | Digital | Placeholders

	Name	Default constant value	Value type	Unit	Comment
0	my_placeholder	4711	Any (numeric default)	unit	comment1
✻					

Calculate

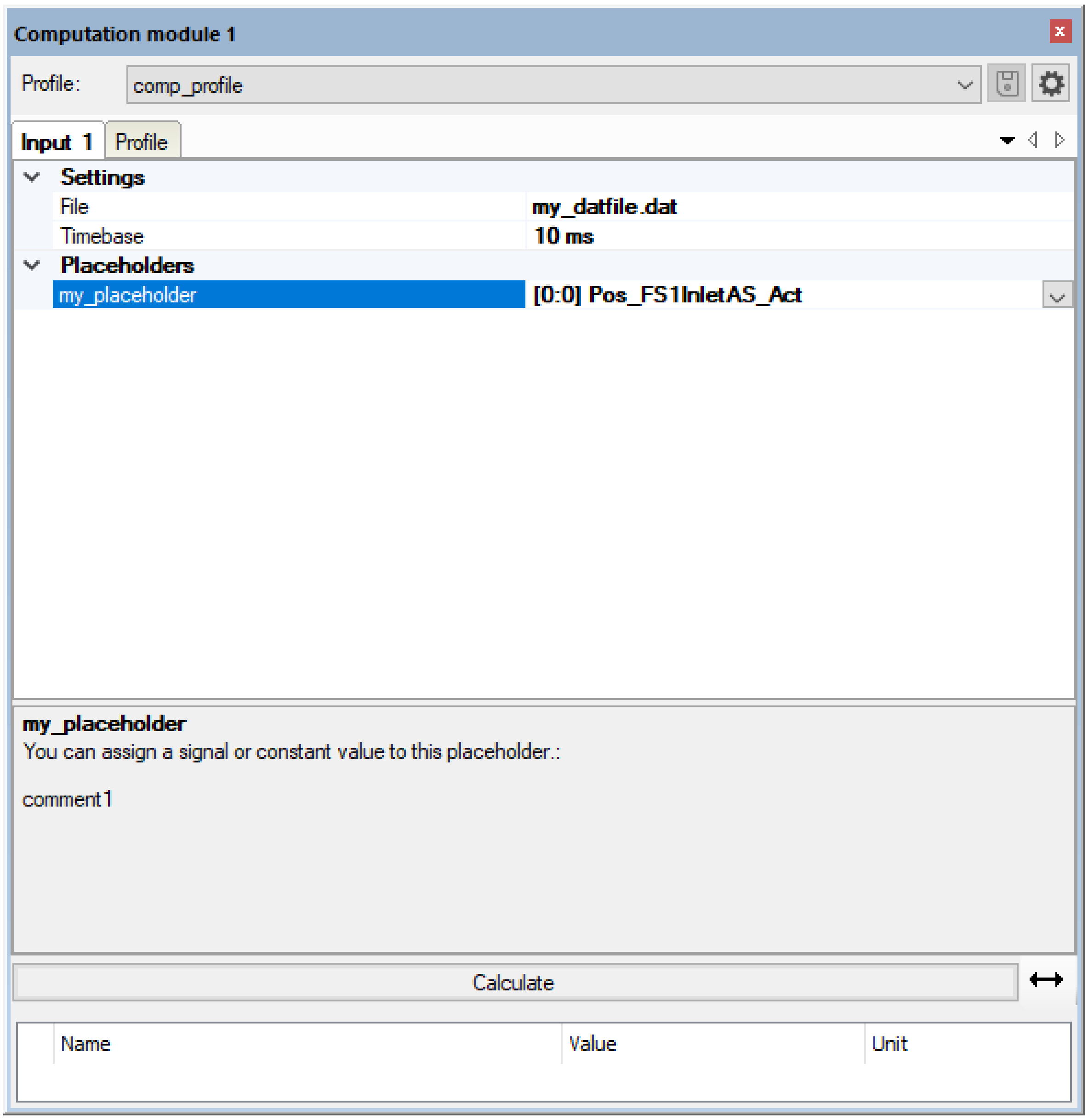
Name	Value	Unit

Similar to the integration of ibaInSpectra and ibaInCycle, profiles can be loaded, modified, and saved from this view. Within the view, it is possible to modify the profile settings.

In the “Placeholders” tab you can specify placeholders. Input signals will then be mapped to these placeholders to do the calculations specified in the “Analog” and “Digital” tabs. For the calculations, you can use the complete ibaPDA expression builder where the placeholders serve as input.

Please refer to the ibaPDA manual part 4 for syntax and usage of the ibaPDA expression builder.

Depending on the profile, input signals can be specified in the input tab. The result of the view is available as additional signal and corresponds to the results in ibaPDA when using a computation module.


Computation module 1

Profile: comp_profile

Input 1 | Profile

Settings

File	my_datfile.dat
Timebase	10 ms

Placeholders

my_placeholder	[0:0] Pos_FS1InletAS_Act

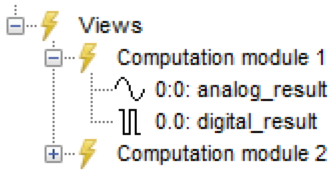
my_placeholder
You can assign a signal or constant value to this placeholder.:

comment1

Calculate

Name	Value	Unit

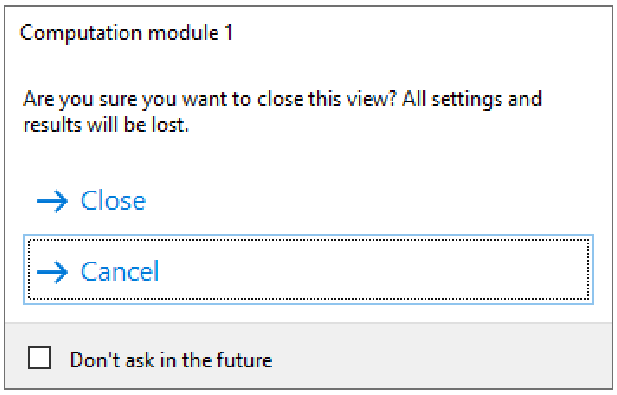
With the selection of input data, the results from the computation module are available as signals in the ibaAnalyzer file tree.


Views
Computation module 1
0:0: analog_result
0.0: digital_result
Computation module 2

For further details on how to use the computation module, please also refer to the New Feature document of ibaPDA v7.3.0.

10 Warning when closing Views

When closing a view in ibaAnalyzer, a warning is shown to not lose data from the view by accident.


Computation module 1

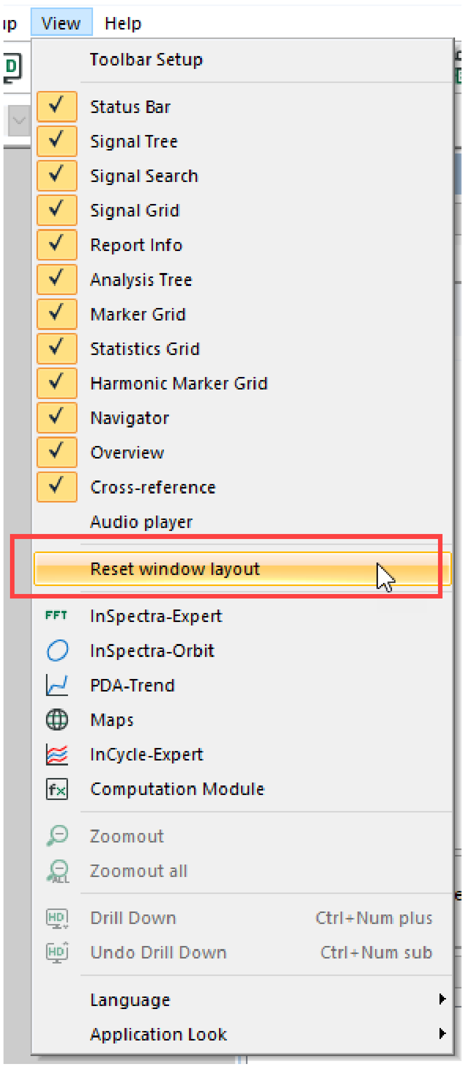
Are you sure you want to close this view? All settings and results will be lost.

→ Close

→ Cancel

☐ Don't ask in the future

If you select “don't show again” once, the warning will no longer be shown for any view. To return to the previous behavior use the “reset window layout” function



11 ISO 8601 timestamp support in text files

11.1 Importing text files

When importing text files including a “time” column in ibaAnalyzer, the timestamps need a specific format in order to be interpreted correctly by ibaAnalyzer. Next to the already supported format, the text file can now also contain time-stamps of the following form which is compatible with the ISO 8601 norm:

YYYY-MM-DD[T]hh:mm:ss[.ffffff][±hh:mm]

The parts denoted in brackets [ ] are optional. The letter “T” to separate date and time can be substituted by a space. The fraction of the second needs always six digits if it is present. The time-zone can be added including hours and minutes. The letter “Z” can be used instead to indicate UTC time.

Examples of valid time-stamps:

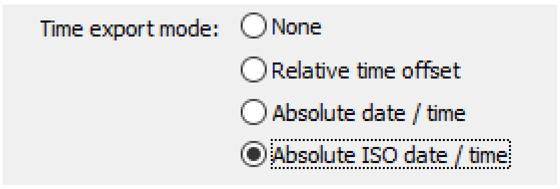
- 2022-09-01T07:30:05.123456+02:00
- 2022-09-01T07:30:05.123456-04:00
- 2022-09-01 07:30:05Z
- 2022-09-01 07:30:05.123456
- 2022-09-01 07:30:05+02:00

Examples of **INVALID** time-stamps:

- 2022-09-01T07:30:05.123
- 2022-09-01 07:30:05+Z
- 2022/09/01 07:30:05

11.2 Exporting or Extracting text files

When exporting or extracting data to a text-file in ibaAnalyzer, a time-stamp can be added. The time export mode has been extended with an additional option “Absolute ISO date/time”



Time export mode: None / Relative time offset / Absolute date / time / [✓ Absolute ISO date / time]

This option is now available in all places where an export of text-files can be configured in ibaAnalyzer. The time stamp is written to the text-file in the following form

YYYY-MM-DDThh:mm:ss.ffffff±hh:mm

With a letter “T” separating the date and the time. The fraction of the second is always denoted with six digits and the time-zone is added including hours and minutes. If the time-zone is +00:00 (UTC time) the letter “Z” is appended instead of “+00:00”.

The time-zone information is taken from the infofield “$PDA_UtcOffset”. If this infofield is not available in the source file, the time-zone information is skipped.

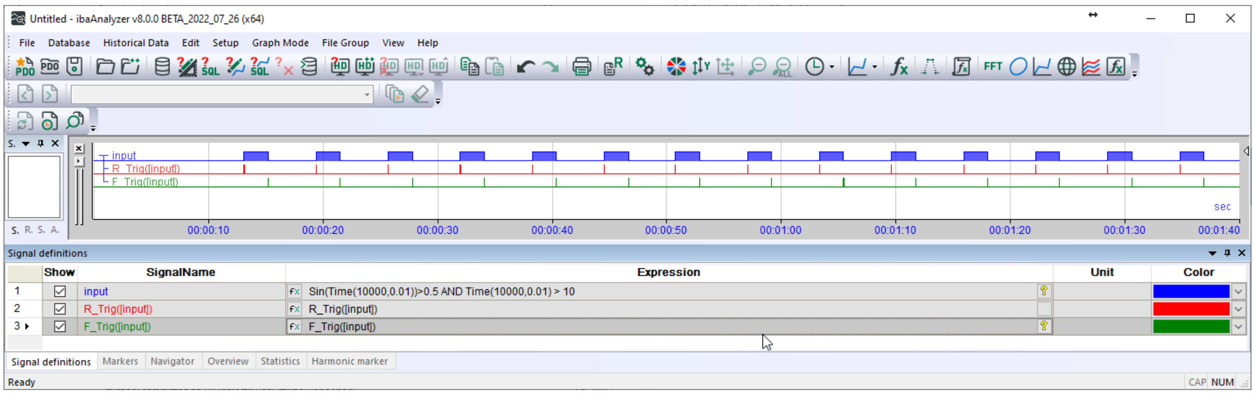
12 New functions and improvements

12.1 R_Trig, F_Trig

In previous versions of ibaAnalyzer, rising or falling edges in digital signals could be detected with the *Diff* function. Two new functions were added to ibaAnalyzer to simplify expressions and to be compatible with PDA expressions:

- *R_Trig*, returns TRUE for every rising edge, i.e. for every transition from FALSE to TRUE.
- *F_Trig*, returns TRUE for every falling edge, i.e. for every transition from TRUE to FALSE.

The functions return FALSE everywhere else. The functions expect only one argument, namely the input signal for which rising or falling edges needs to be detected.


Untitled - ibaAnalyzer v8.0.0 BETA_2022_07_26 (x64)
File Database Historical Data Edit Setup Graph Mode File Group View Help
input
R_Trig([input])
F_Trig([input])
00:00:10 00:00:20 00:00:30 00:00:40 00:00:50 00:01:00 00:01:10 00:01:20 00:01:30 00:01:40
sec
Signal definitions
Signal definitions Markers Navigator Overview Statistics Harmonic marker
Ready
CAP NUM

	Show	SignalName	Expression	Unit	Color
1	✓	input	Sin(Time(10000,0.01))>0.5 AND Time(10000,0.01) > 10		🟦
2	✓	R_Trig([input])	R_Trig([input])		🟥
3	✓	F_Trig([input])	F_Trig([input])		🟩

12.2 RelativeTime

The *RelativeTime* function will calculate the number of seconds since the *ibaAnalyzer start time* (usually the start time of the loaded file with the earliest start time) given an absolute time stamp specified by individual time components. This function is intended to be used in other functions that require time components in relative time or for expressions in the marker dialog.

The function takes the following arguments:

- *Year,* 4-digit year component e.g.: *2022*. This value is expected to be integral and if not will be rounded to the nearest integer.
- *Month*, the month component, 1 for January, 2 for February, …, 12 for December. This value is expected to be integral and if not will be rounded to the nearest integer. If this integral value is not between 1 and 12, an invalid sample (i.e., a gap) will be inserted.
- *Day*, the day component, Minimum 1, maximum 31. This value is expected to be integral and if not will be rounded to the nearest integer. If this integral value is not valid for the given month and year, an invalid sample (i.e., a gap) will be inserted.
- *Hour,* the hour component, from 0 inclusive up to 24 exclusive. If you have your time component in a 12 hour time period, PM times after 12h59 need 12 hours added. This value is expected to be integral and if not will be rounded to the nearest integer. If this value is negative or is equal to or above 24, an invalid sample (i.e., a gap) will be inserted.
- *Minute,* the minute component, from 0 inclusive up to 60 exclusive. This value is expected to be integral and if not will be rounded to the nearest integer. If this value is negative or is equal to or above 60, an invalid sample (i.e., a gap) will be inserted.
- *Second,* the second component, from 0 inclusive up to 60 exclusive. This value can have a decimal component, it does not need to be integral. If this value is negative or it equals or exceeds 60, an invalid sample (i.e., a gap) will be inserted.
- *Millisecond,* the millisecond component*.* This value is optional and can be omitted. If present, this value divided by 1000 is added to the seconds part (which then still needs to below 60 for a valid sample). The value does not need to be integral.

Example

If the start time of the file is 2022-09-01 07:30:00 the expression *RelativeTime(2022,9,1,7,31,5)* will return the value 65 (meaning that the time given as argument is 65 seconds after the start time)