Fuzible - General Documentation

Author	Guillaume Tristant
CREATION	20171002
REVIEW	20221227
Resources	www.fuzible-app.com
SUPPORT MAIL	guillaume@fuzible-app.com
Forum	www.fuzible-app.com/wordpress/fuzibleForum/phpBB3/
Language	FR / EN

Table of Content

Preamble

First, I would like to thank you warmly for using (or trying) Fuzible. This software is the culmination of several years of work, and the observation that many tasks related to data manipulation are too often time-consuming and redundant.

There are many alternatives, and the idea is not to create competition, but another way of designing data exchange and manipulation.

The program is aimed at developers with minimal knowledge of the SQL language.

The philosophy of the program is simple:

Any Data Source is a database and can be queried as such!

Some use cases:

- Data replication (copy)
- Data synchronization (smart comparison of 2 sources)
- Interfaces (ex: retrieve data from one software's webservice in SaaS and send it to another software in the form of XML files)
- Migration of data from one BDD to another (regardless of the driver)
- Fast data extraction (ex: SQL to CSV)
- Fast data importation (ex: EXCEL to BDD)
- Data comparison (ex: to control the integrity of 2 BDDs)
- Cross-join from different sources in real time
- Filling a data warehouse (ex:, integrating a file into a BDD is fully automated: from data analysis to table creation)
- Sync a pre-production environment with a production environment

Information about documentation

The documentation was written in French and translated in English using an automatic translator. I checked a few key things but I am sorry if there are still weird phrases!

Principle

Fuzible is a tool that allows you to import, export, mix, synchronize, replicate, compare data. The general principle is based on the definition of a Data Source (which will be queried), and a Target (in which data will be copied).

The tool can work with several SGBDs (see compatibility table), both for export and for data import. It offers an opportunity to analyze all the data needed to create intelligent import fields (consistent types and lengths). But beyond that, he can read and write in files, webservices, mailboxes, and in the Active Directory!

Finally, it is designed as modular, from data processing to LOG management and Jobs orchestration. A Swiss knife, in short. It can meet the most basic needs (copying a CSV file in a database) to the most complex (synchronizing 2 database environments).

Installation

The program can be installed anywhere on the hard drive. You can choose to use the portable version or the installer version. In both cases, no data will be written anywhere other than in the installation directory.

Recommended configuration

This assessment is based on my tests. Note that multithreading exponentially increases the needs of the machine running the program. Setup indicated for 4 threads.

- Microsoft Windows OS (from Windows 7, or Server 2008r2 and more)
- 4th generation Intel I5 CPU with at least 2 execution threads
- 8GB of RAM (large minimum)
- HDD 7200tr/min with10GBavailable.
- Internet access
- Microsoft NetCore Framework v3.1 (https://dotnet.microsoft.com/download)

Language

The program is available in 2 languages (French, English). By default, it is set to use the system language, but you can switch from one language to another in the "File" menu.

Enterprise installation

In a network environment, Fuzible is multi-user (and delivers all its possibilities when used like this), it is advisable to install it on a server accessible in RDP. In addition, a server will often have much more access rights to SQL instances, network paths, FTP... than a local computer.

Similarly, if you replicate from one BDD to another, and Fuzible is installed on a local computer, processing times could be horribly slow because the data will have to be retrieved first from the Source server to the local computer and then from it to the Target server. If you are behind a VPN, or if your connection to the network is slow, performance will be extremely degraded.

Installation for an Individual

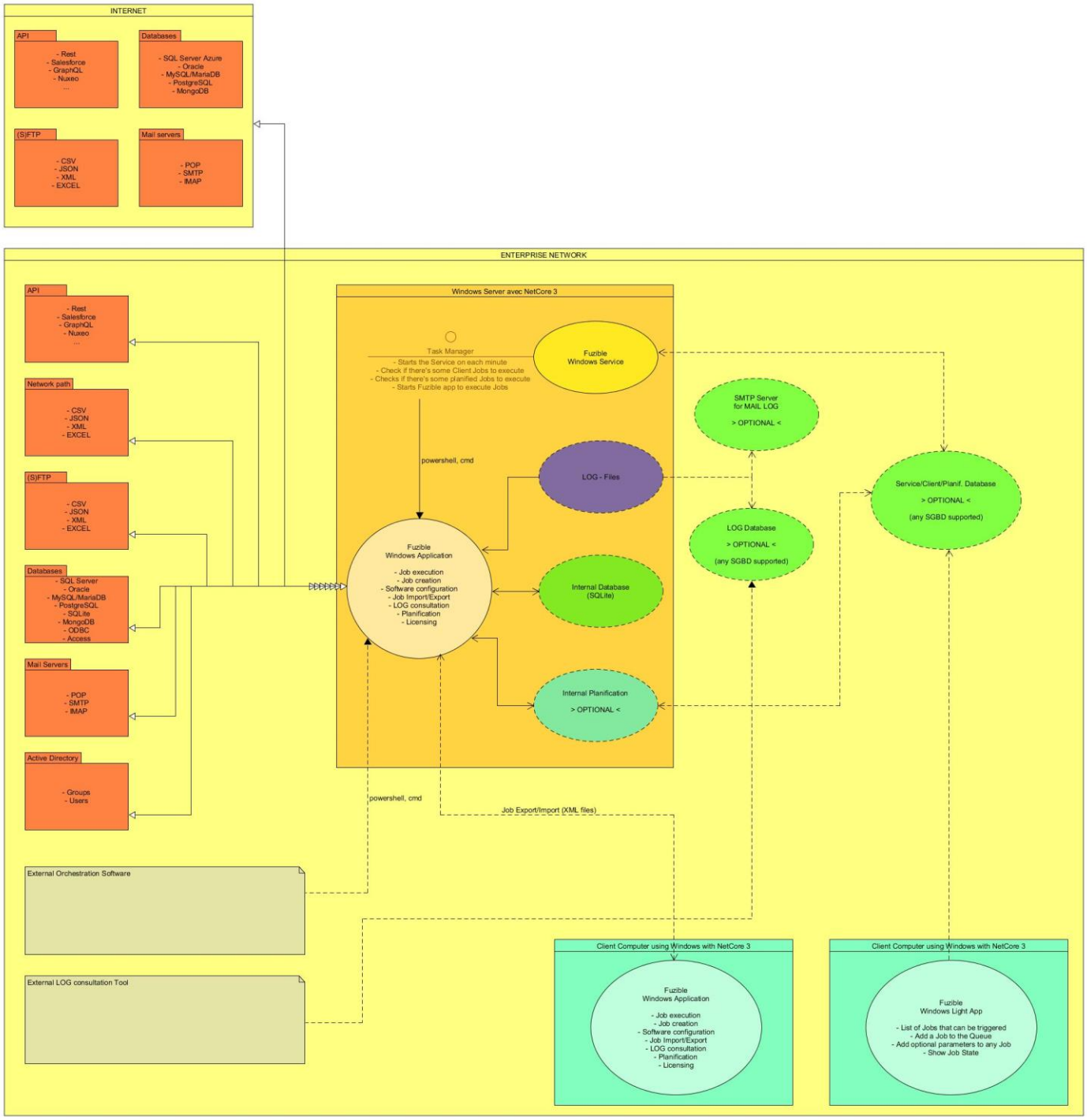
Just make sure that the computer on which you install the app accesses all the data sources you will handle (SQL instances, local network, Active Directory domain, Internet connection...) and that the user account is "administrator" of the computer.

Software Architecture

The diagram shows the different bricks of the application. Some of them are optional and do not need to be configured other than the application itself.

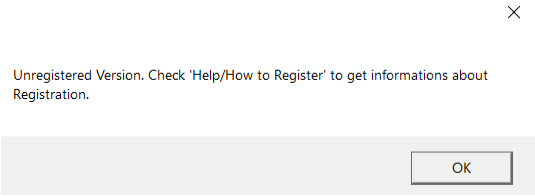
The Core application is in the Orange rectangle.

All dotted lines represent optional bricks. Green ones are included with Fuzible, and gray ones are external.



Software Registration

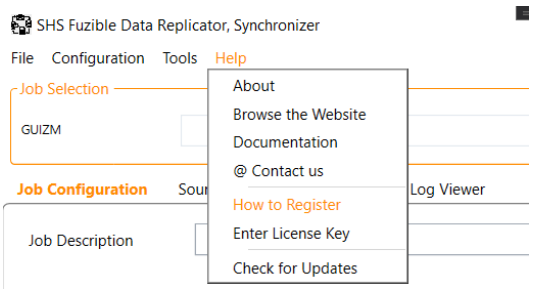
When starting the software, whether it was purchased or not, it runs in demonstration mode. It is limited in its use and a message will inform you at each start.



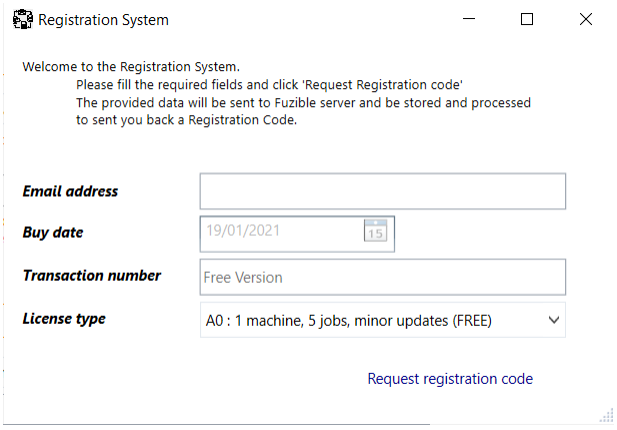
You will not be able to do the following:

- Orchestrating Jobs
- Importing and exporting Jobs
- Use the “Client” and “Service” app
- Create more than 3 Jobs

To Register Fuzible, just go to the Help menu:



A menu opens and asks you to enter your email address to register.



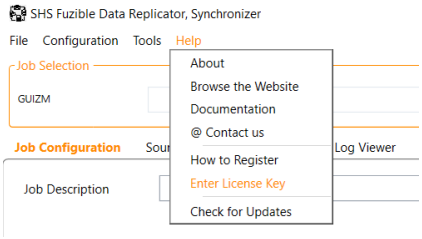
If you purchased the program, you also need to choose the type of License you purchased from the drop-down list, and then specify the transaction number that was provided to you at the time of purchase, as well as the date the License was purchased.

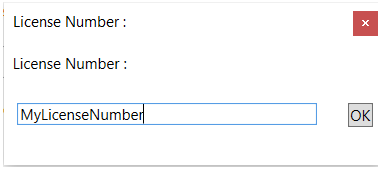
Note: Your information is only stored on Fuzible's server, the data is not transmitted under any circumstances to anyone. Registering the free version allows us to measure the number of active users on the app.

The data sent is: email address, CPU model, amount of RAM, Windows version, version of the NetCore Framework, unique PC identifiers (main hard drive, CPU), amount of Jobs in Fuzible, amount of connections, local IP address, public IP address, program installation path, program version.

Click "Request Registration Code" to send your data to the Fuzible Server. Your registration request will be processed manually within a few minutes, and a registration code will then be sent back to you by email.

This should be entered in the Help menu:





The code authenticity is checked locally, and a message will inform you of the validity of the code.

The free version gives you all the features of the app, but the number of Jobs that can be created is limited to 5. Paid versions are built around the number of Jobs that can be created, as well as the type of updates that will be possible to download in the future (corrective updates, minor, major developments).

License Perimeter

A Fuzible License is multi-user (several users on the same computer can use the app independently) but is limited to the computer on which the application is installed. The following changes will cause the program to return to demo mode:

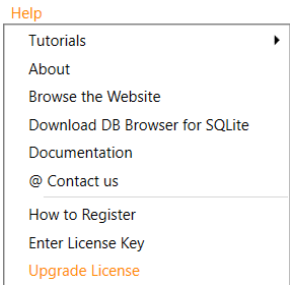
- Changing the installation directory
- Change of CPU
- Change of hard drive
- Manual changes to the program's internal database

If a change in circumstances proves to be legitimate, you can request assistance on the website:

www.fuzible-app.com

License Upgrade

If you already own a License, and you just acquired a better one from our website, you can go to this menu to update the software. The Registration process will be the same as the first time you did register the app.



Program compatibility

Dbms	Support
SQL Server	SOURCE: Full support
	CIBLE: Total support
Mysql	SOURCE: Full support
	TARGET: Total support
Postgres SQL	SOURCE: Full support
	TARGET: Total support
Odbc	SOURCE: Full support
	TARGET: Partial support: Need you to set up some system queries
Oracle	SOURCE: Full support
	TARGET: Total support
Sqlite	SOURCE: Full support
	TARGET: Total support
Access	SOURCE: Full support
	TARGET: Total support
Mongodb	SOURCE: Full support
	TARGET: Total support
File	SOURCE: Support XML, JSON, CSV, XLS, XLSX (FTP)
	TARGET: Support XML, JSON, CSV, XLS, XLSX (FTP)
Webservices	SOURCE: API REST
	TARGET: Partial support (REST, http)
Mailbox	SOURCE: POP, IMAP
	TARGET: SMTP
ACTIVE DIRECTORY	SOURCE: users, groups
	TARGET: users, groups

Supported Scenarios

Source	Target	REPLICATION	SYNCHRONIZATION
Database	Database	Yes	Yes
Database	File	Yes	Yes
Database	Webservice	Yes	No
Database	Mailbox	Yes	No
Database	Active Directory	Yes	Yes
File	Database	Yes	Yes
File	File	Yes	Yes
File	Webservice	Yes	No
File	Mailbox	Yes	No
File	Active Directory	Yes	Yes
Webservice	Database	Yes	Yes
Webservice	File	Yes	Yes
Webservice	Webservice	Yes	No
Webservice	Mailbox	Yes	No
Webservice	Active Directory	Yes	Yes
Mailbox	Database	Yes	Yes
Mailbox	File	Yes	Yes
Mailbox	Webservice	Yes	No
Mailbox	Mailbox	Yes	No
Mailbox	Active Directory	Yes	Yes
Active Directory	Database	Yes	Yes
Active Directory	File	Yes	Yes
Active Directory	Webservice	Yes	No
Active Directory	Mailbox	Yes	No
Active Directory	Active Directory	Yes	Yes

Security

The program uses an SQLite database to work. Several information is encrypted (AES), including login chains and passwords, to protect data privacy for each user session.

How it works, start-up

Start the program in "UI" mode

Run **Fuzible.exe**

Start the program in "CONSOLE" mode (silent execution of a Job)

Run **Fuzible.exe** with arguments

1 - Userspace (basically, the user who is connected)

2- Job ID to run at start-up (if needed)

3 - Password(encrypted)

4 - Dynamic Parameters (see "Script Language" section)

Example:

Fuzible .exe "GUILLAUME" "[10]" Apza-7824

- Will launch the no.10 job of the user "GUILLAUME"

Given the austerity of entering arguments, the program proposes, in its UI, to display you the "launcher" code of each Job so that you can copy and paste it directly (in a scheduling tool for example).

<u>In summary, the program has two operating modes:</u>

- In silent mode (name of job to be performed), it does not load the graphical interface and performs the Job in the background.
- In UI mode, it opens the graphical interface: it allows you to edit, add and launch Jobs

First launch

At the first launch of the program, it will create:

- Your default general configuration
- Two Demo Jobs containing about 20 test queries, to introduce you to the features.
- 3 local "file" connections, 1 SQLite connection, 1 connection to a demo webservice

You will also be notified that you are not registered, and because of that, you will not be able to create more than 3 Jobs. In addition, some features will be disabled.

Working Path

Fuzible stores its data into the « public » directory : **C:\Users\Public\Documents\Fuzible.**

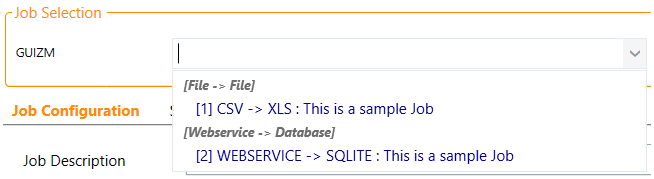
This is where you get the internal Fuzible database, LOG files, the Client App binaries, and also demo files that are used by the default Job File connections.

Integrated software support

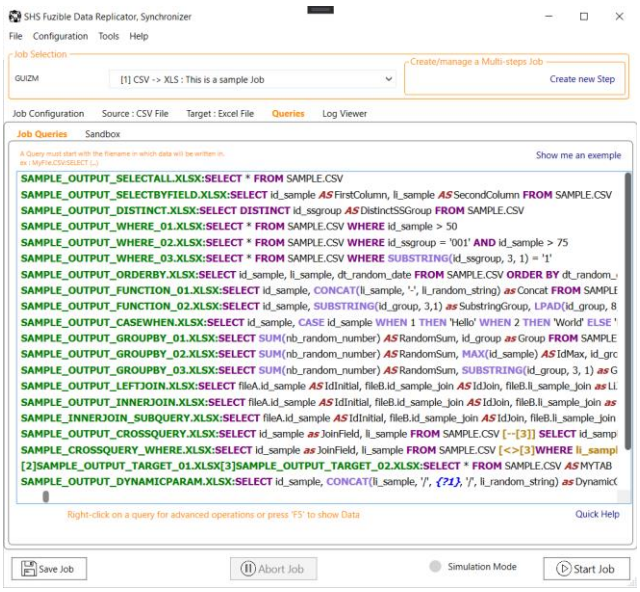
Like any program, the first use can sometimes be off-putting. Fuzible is no exception to the rule, but before closing the program permanently, let me show you how you can get some assistance.

Demonstration jobs

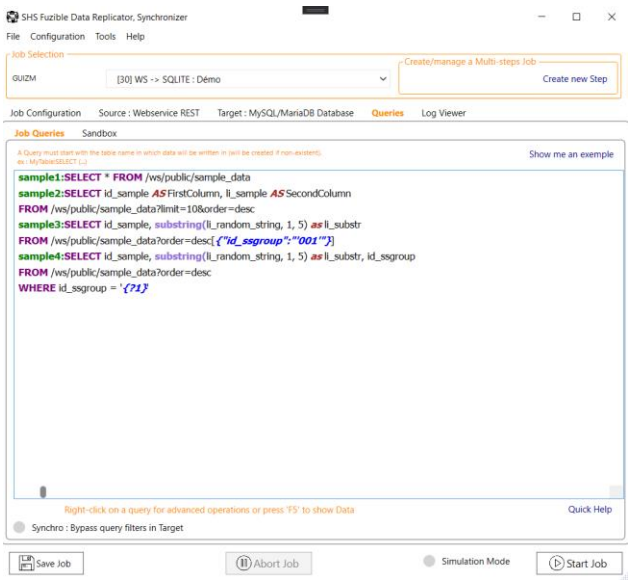
During the first launch, Fuzible will create 2 Demo Jobs for you that will help you understand how to query a Source other than a database using the SQL language. Those are accessed through the selection menu:



- A Job that copies data from CSV files into EXCEL (XLSX) files:

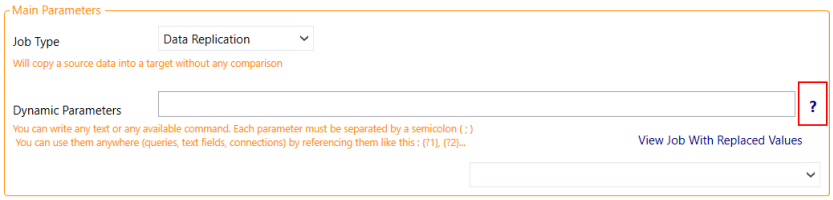


- A Job that retrieves data from the demonstration webservice and integrates it into Fuzible's local database:



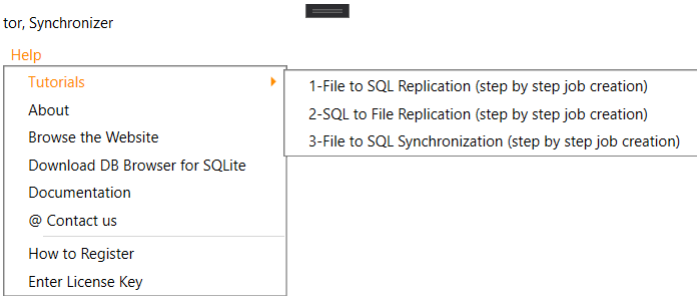
Quick Help Button

Some features do have a small "?" button to give you some quick explanations without having to look at the full documentation:



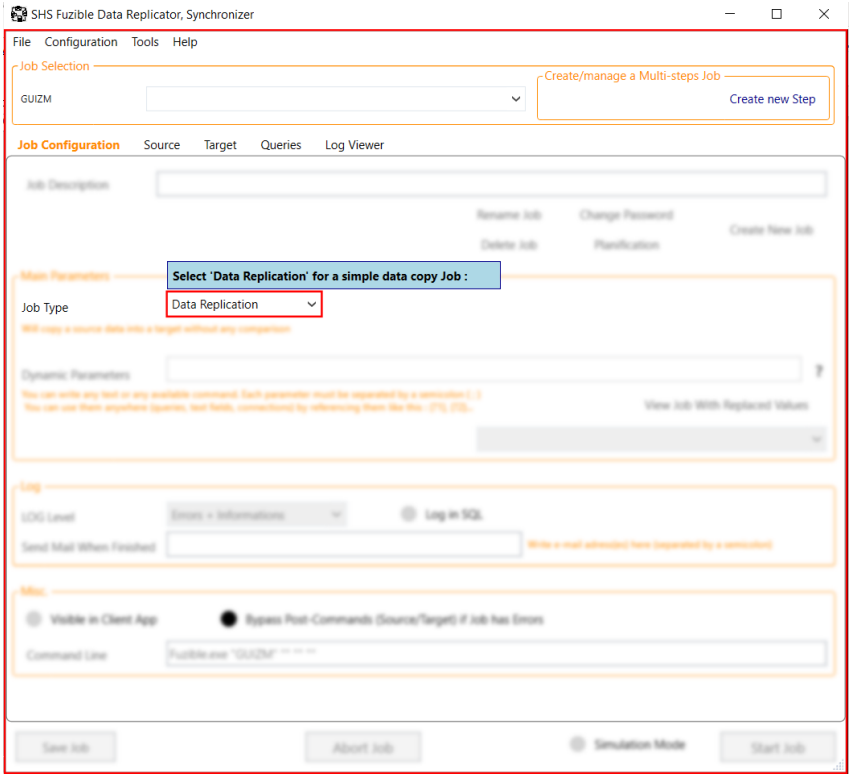
Tutorials

Three tutorials have been programmed to make it easier for you to accommodate with the software. You can find them here:



A script then triggers, and you will be fully guided.

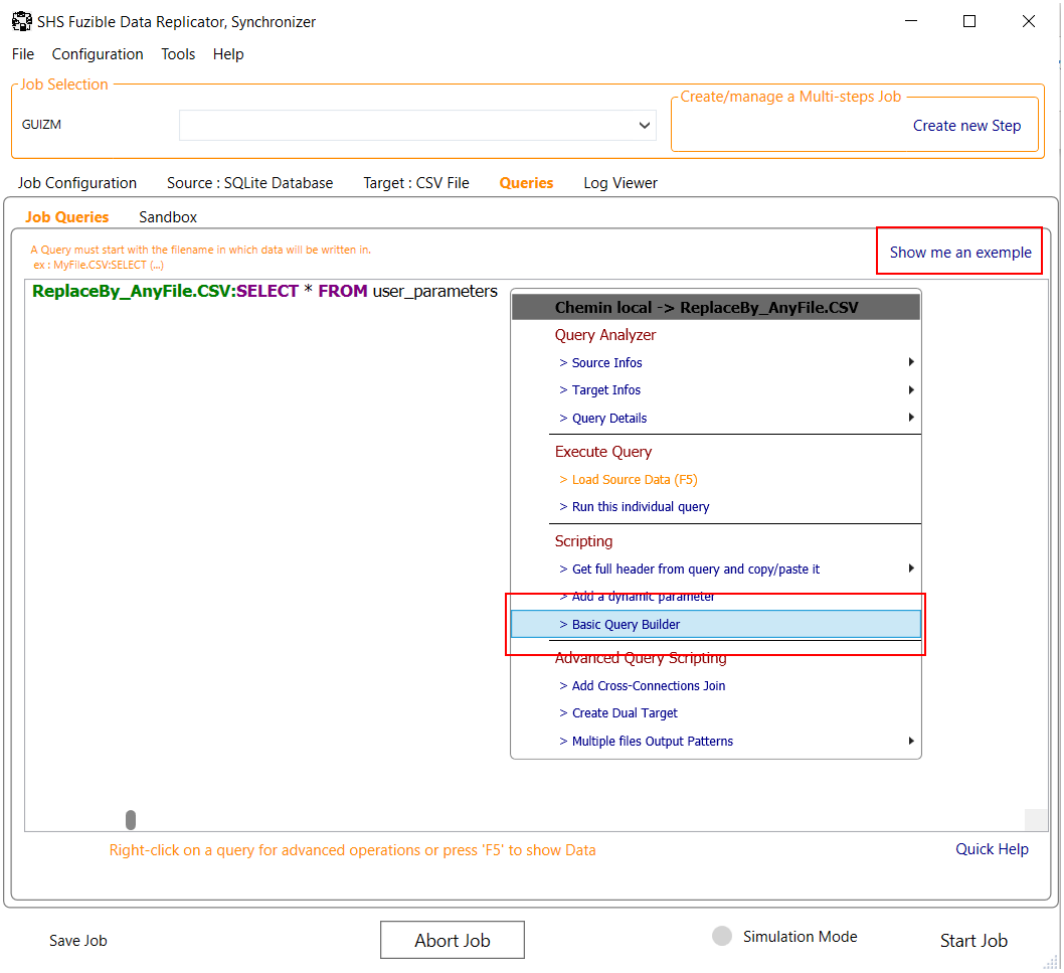
- "Escape" key: Leaves Tutorial mode
- "F1" key: Move forward at the next step.
- "F2" key: Go back to the previous step.



Query Assistant

When you are creating your Job, you'll be writing queries, 2 options are available to help you understand Fuzible's philosophy.

- "Show Me an example" will show you a "generic" query adapted to the scenario you set.
- "Basic Query Builder" will accompany you in creating a simple query.



However, once you understand the logic of the program, these 2 options will be of no use to you, they are simply here to help you understand how Fuzible works, but I'm sure that Tutorials and Demonstration Jobs will be much more useful!

Online help

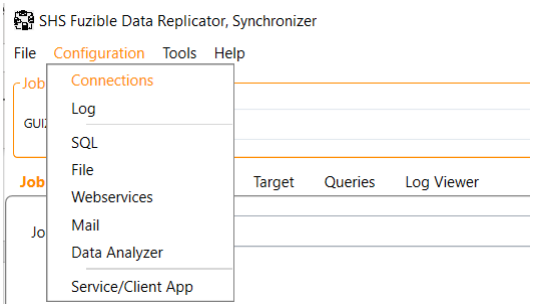
If you need additional help, feel free to go to the forum (link at the top of this documentation), contact me, or download Demo Job.

Indeed, Fuzible allows you to create complex scenarios. Its highly modular design allows it to meet many needs, but it can be helpful to ask for help if you are not sure the best way to achieve your goals.

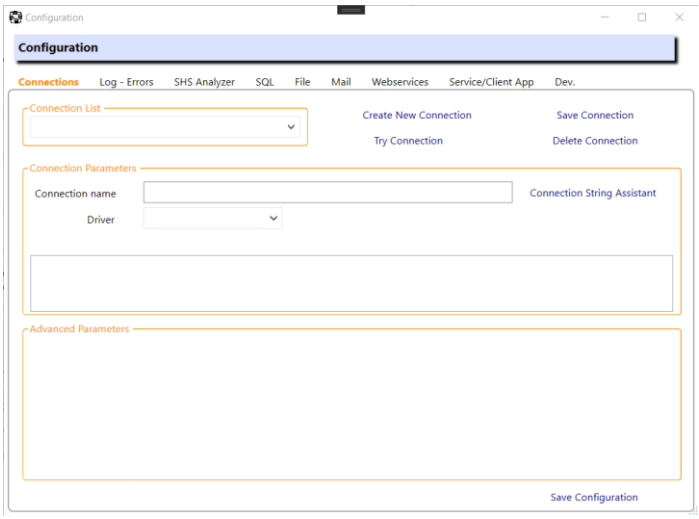
Software Settings

Connections

Go to the Configuration / Connections menu to set up the connection strings.

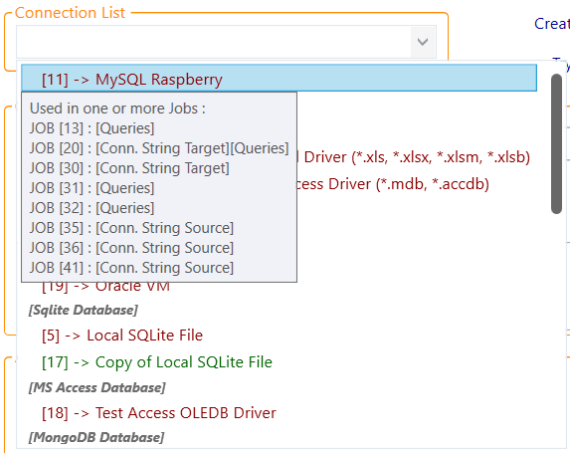


A menu then opens:

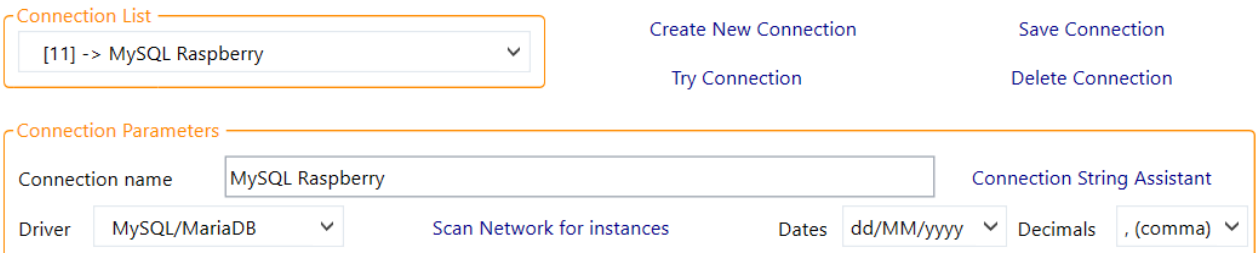


You can either choose to change or delete an existing connection.

Beware of impacts, the red connections in the list are used in at least 1 job! You can see this impact by flying over the name connection with the mouse:



Each connection is identified by an ID (ex: [10]) which allows it to be used in queries (see chapter related to queries)



- The common settings allow you to create a new connection:
  1/ From an existing connection if already selected.
  2/ From the data entered by the user.

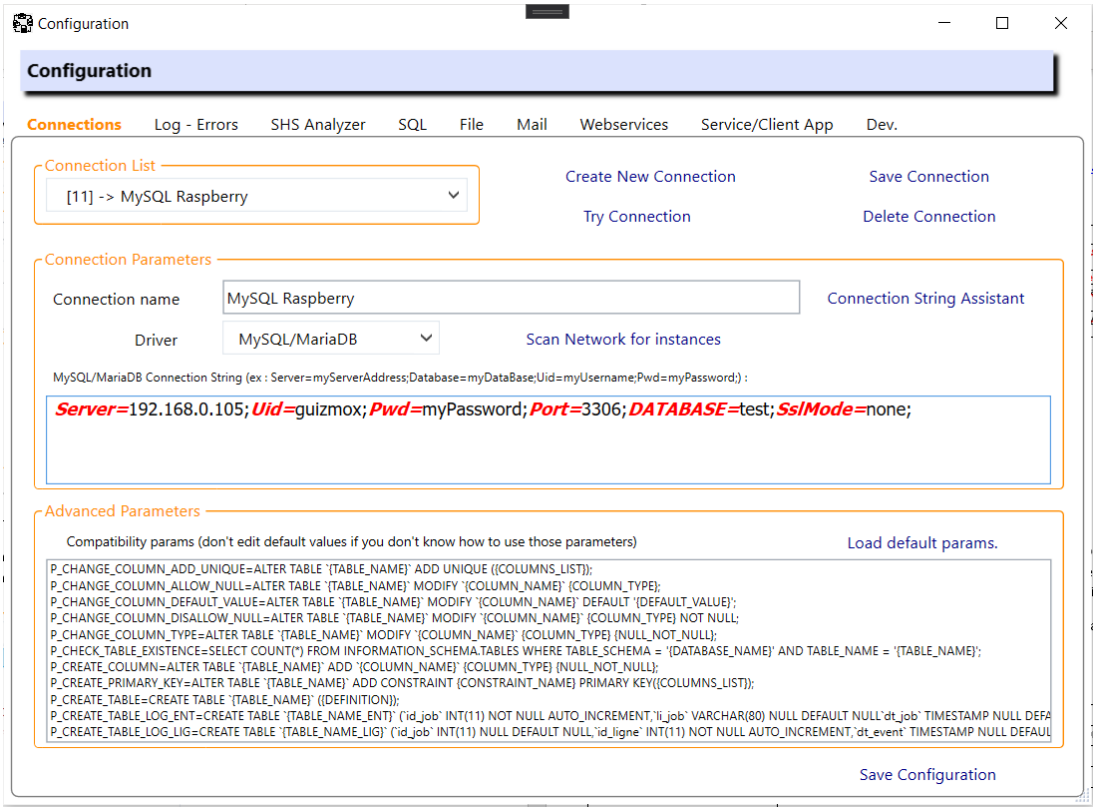
- You can also test the current connection to check if it is working.

- The connection string assistant will accompany you during the creation of the new connection.

- You can also change dates & decimals localization connection properties: you may need for example to extract US data (MM/dd/YYYY dates, dotted decimals) and import them into a European database (dd/MM/yyyy, comma decimals)
  It's not changing your OS locales – you just change the way data is handled by Fuzible for a specific connection.
  Note : by default, the localization settings are set according to your OS localization.

- Finally, in the case of a database, an option allows you to scan the local network to find possible available instances (the tool scans the network for open ports: those are different depending on the SGBD you choose).

Database



A complete listing of connection strings can easily be found online: https://www.connectionstrings.com/

However, here is a basic example for the different SGBDs

DRIVER TYPE	EXPECTED CONNECTION STRING
BDD SQL Server	server=MyServer;DATABASE=MyDatabase;User ID=user;Password=password;Trusted_Connection=True;Connection Timeout=60;Integrated Security=false;
BDD MySQL	server= MyServer;uid= user;pwd= password;DATABASE= MyDatabase;Convert Zero Datetime=True;SslMode=none;
BDD Postgres	Server= MyServer;Port=5432;DATABASE=MyDatabase;Userid= user;Password= password;Ssl Mode=Require;Trust Server Certificate=true;
BDD ODBC	DRIVER={HyperfileSQL};Server Name=MyServer;Server Port=4900;DATABASE=MyDatabase;UID=user;PWD=password;
BDD SQLite	Data Source=C:\Tools\Fuzible\Fuzible.db;Version=3;Foreign Keys=true;
BDD Access	Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Tools\Fuzible\accessDB.accdb;Persist Security Info=False;
BDD Oracle	Data Source=WIN81VIRTUEL;User Id=guizmox;Password=myPassword;

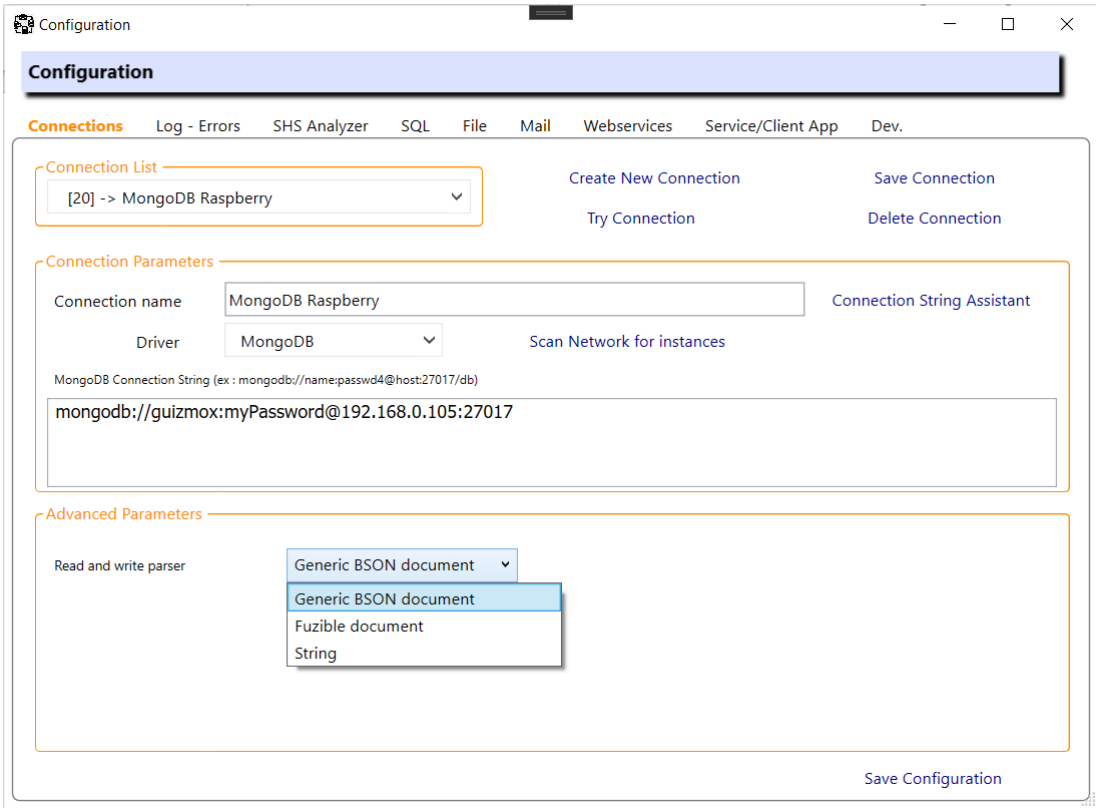
ODBC Driver Case: Fuzible uses specific queries and settings for each type of database for its proper operation. By definition, those are unique to each SGBD and you may want to enjoy all the benefits of Fuzible with an unsupported native driver.

The "Connection Params" zone is there to meet this need. It sets up each of Fuzible's internal queries to make it compatible with any SGBD.

Parameter	**Value**
P_ESCAPE_CHAR	Escape character to delineate table and field names (often quotation marks)
P_DATE_FORMAT	Native date format (DATE, TIMESTAMP, DATETIME)
P_STRING_FORMAT	Native character format (VARCHAR, VARCHAR2, NVARCHAR)
P_CREATE_PRIMARY_KEY	SQL code for creating a primary key in a table
P_CREATE_TABLE	SQL code to create a table
P_CHANGE_COLUMN_ALLOW_NULL	SQL code to allow NULL values in a specific column
P_CHANGE_COLUMN_DISALLOW_NULL	SQL code to ban NULL values in a specific column
P_CREATE_COLUMN	SQL code to create a column in a table
P_CHANGE_COLUMN_TYPE	SQL code to alter the size of a column in a table
P_CHANGE_COLUMN_DEFAULT_VALUE	SQL code to change or put a default value in a column in a table
P_CHANGE_COLUMN_ADD_UNIQUE	SQL code to add a single key to a column in a table
P_SHRINK_TABLE	SQL code to clean a table
P_GET_COLUMNS_LIST	SQL code to retrieve the list of columns of a table (expected: name, type, max size, nullable,identity, type, numerical accuracy, default, single, key)
P_GET_COLUMN	SQL code to retrieve information from a column of a table (expected: name, type, max size, nullable,identity, type, numerical accuracy, default, single, key)
P_DISABLE_TABLE_CONSTRAINTS	SQL code to disable table constraints (for example, allow insertion without key control)
P_ENABLE_TABLE_CONSTRAINTS	SQL code to activate table constraints (key control)
P_GET_TABLES	SQL code to retrieve a BDD's table list
P_GET_TABLES_FILTERED	SQL code to retrieve the filtered list of tables of a BDD
P_GET_PRIMARY_KEY	SQL code to retrieve the primary key from a table
P_GET_FOREIGN_KEYS	SQL code to retrieve foreign keys from a table
P_CHECK_TABLE_EXISTENCE	SQL code to control the existence of a table on a BDD
P_DELETE_LOG_EVENTS	SQL code to remove a log line from the program
P_INSERT_LOG_EVENT	SQL code to insert a log line of the program
P_UPDATE_LOG_EVENT	SQL code to update a log line of the program
P_CREATE_TABLE_LOG_ENT	SQL code to create the program's LOG table (header)
P_CREATE_TABLE_LOG_LIG	SQL code to create the program LOG table (lines)
P_CREATE_FROM_SELECT	SQL code that creates a table from another (ex: SELECT * INTO [TABLE] FROM [ORIGINAL_TABLE] WHERE 0 = 1)

Note: You can use dynamic parameters in connection strings by using {?1}, {?2}... Those parameters will be replaced by those mentioned in the Job (explained later)

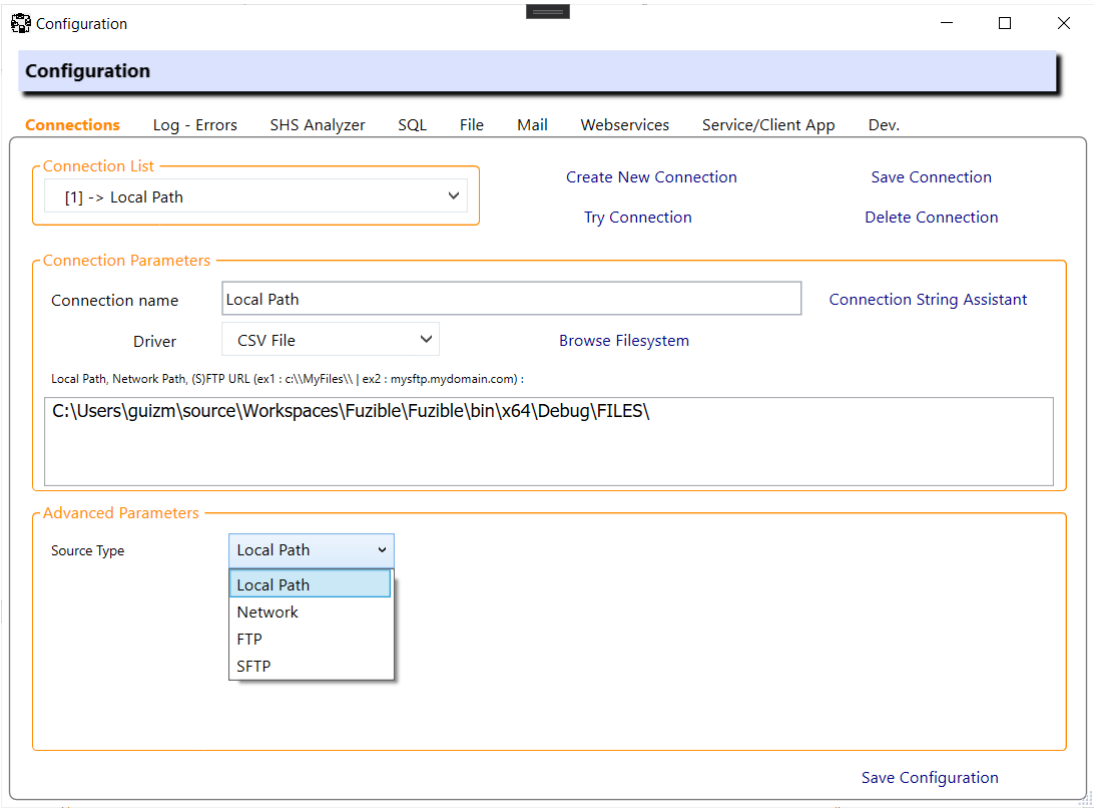
Mongodb



The syntax of a MongoDB connection string can easily be found online: https://www.connectionstrings.com/

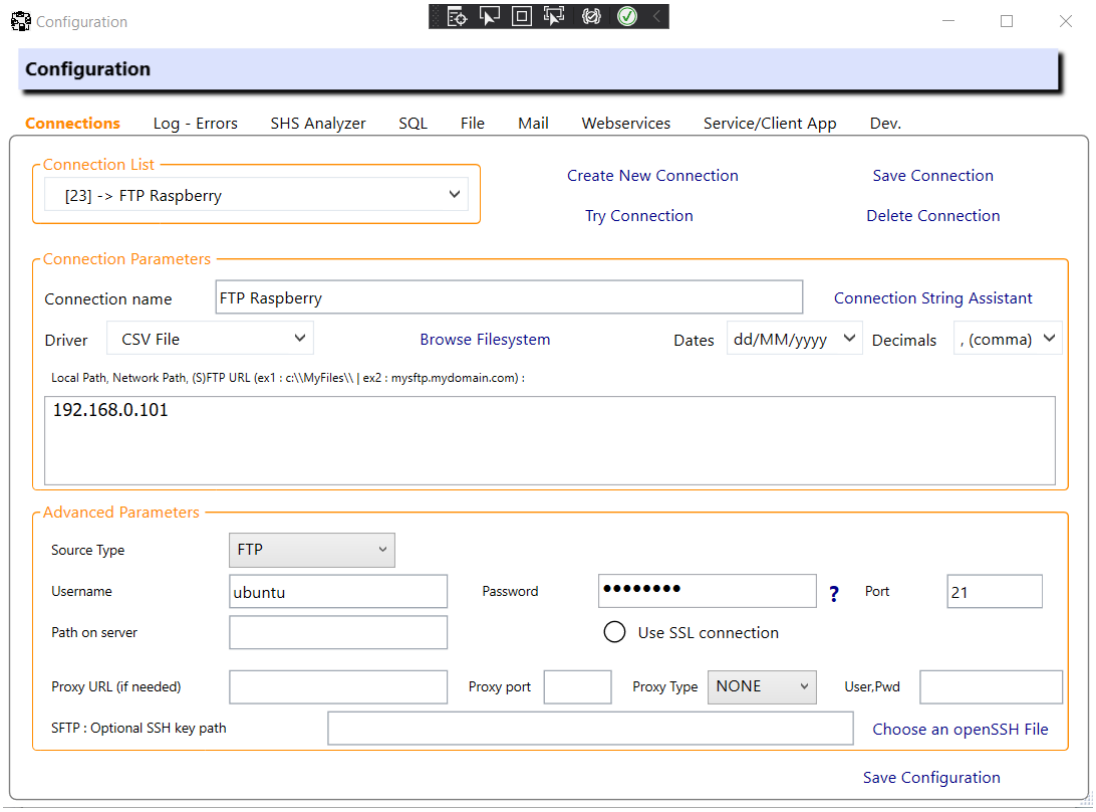
In addition, the MongoDB driver treats collection data as BSON documents by default. However, you can choose to use a Fuzible-specific type (containing additional METADATA) or simply as strings. It all depends on the use.

Files

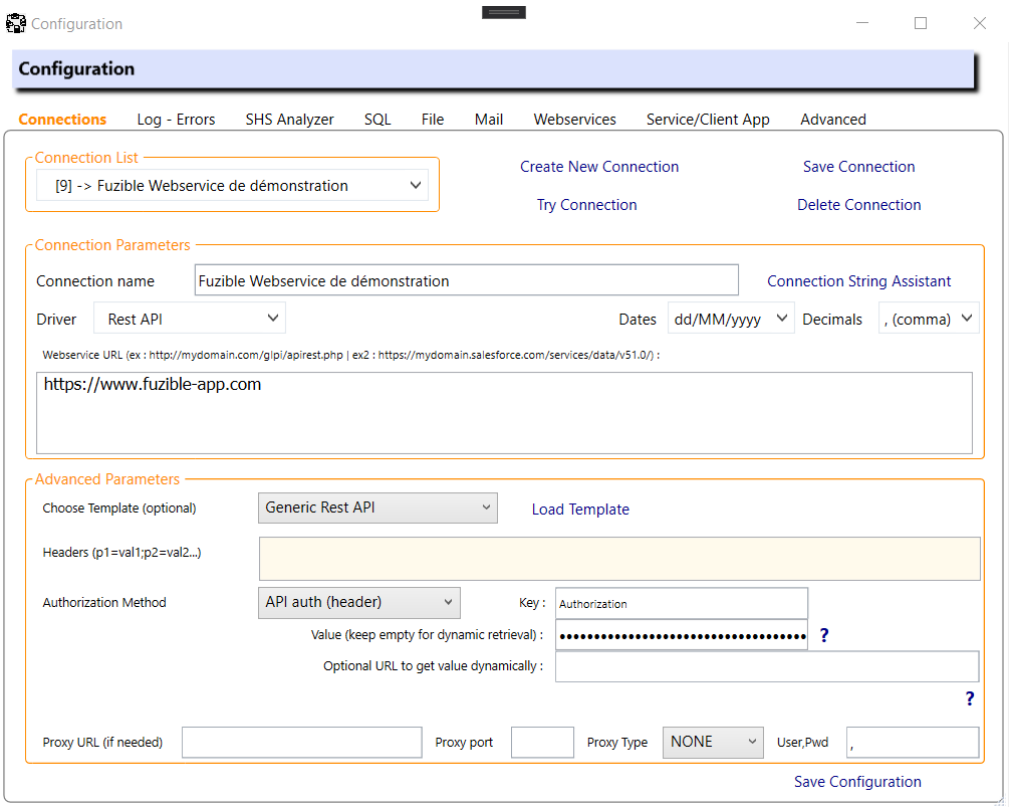


You must choose here the local or network path that leads to the files we want to process in a Job.

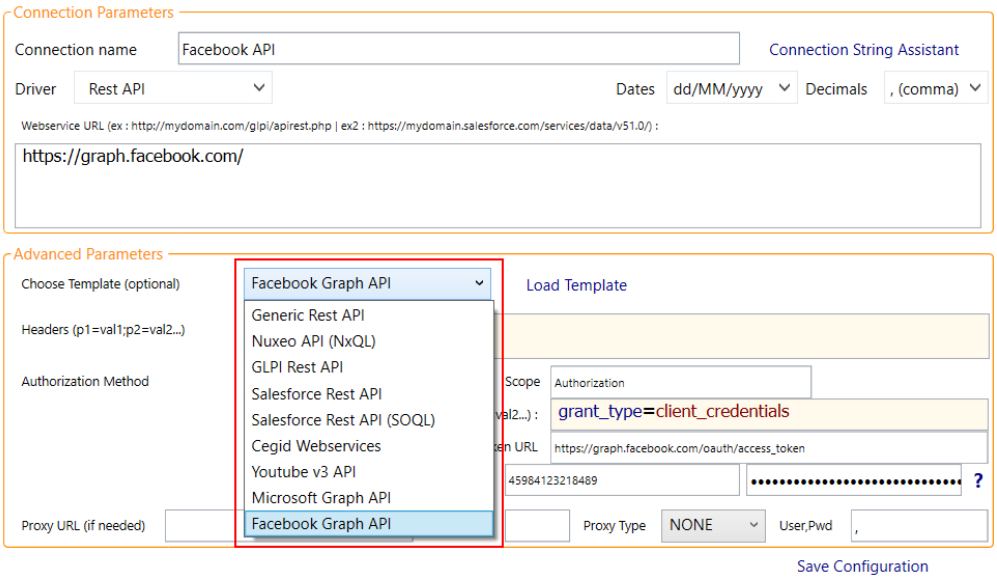
If you change "Source Type," you can also access the (S)FTP settings:



Webservice REST



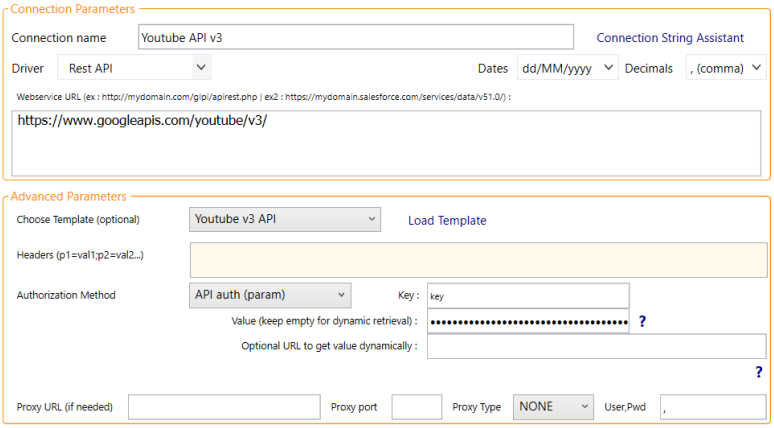
A webservice is a little more complicated to set up because there are several ways to interact with them, that's why a drop-down menu allows you to simplify the task by loading a "template" for some known APIs.



By choosing an example from the list and loading it, the fields will automatically be pre-filled to simplify the setup and understanding of the API connection manager.

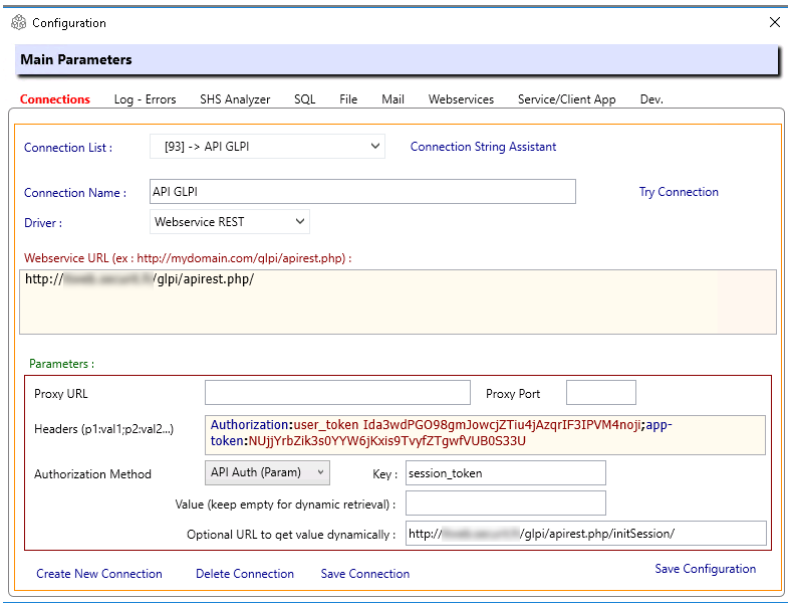
If you want to configure a connection to an unreferenced API by yourself, use "Generic Rest API", and manually configure the settings, as well as the authentication mode among the list of those proposed.

The example below shows the setting of youtube's API v3.

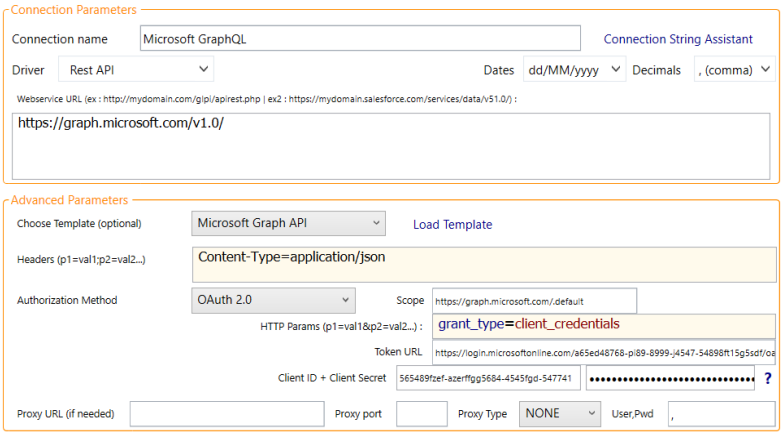


For other APIs, apart from entering the URL, one must also provide all authentication information and sometimes additional header information. For example, a call to the GLPI API requires a dynamic key… You have to add the URL from which you'll retrieve that token (Optional URL to get value dynamically) :

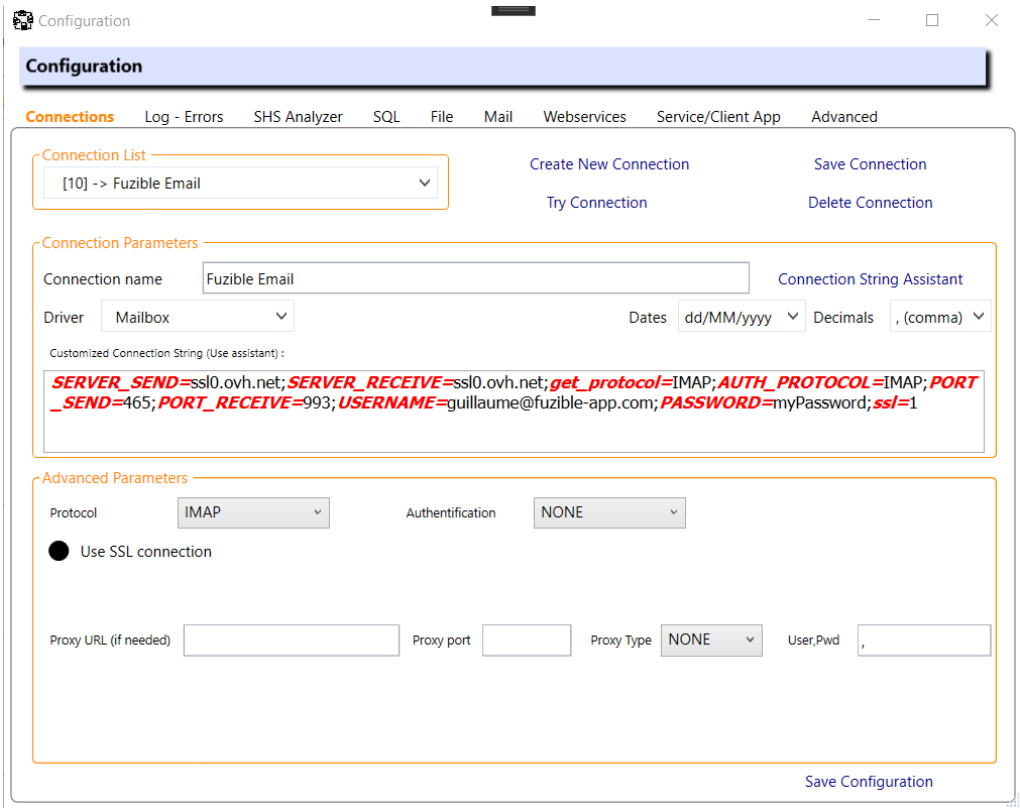
Each connection to the webservice will then automatically first invoke a call to get that token and will allow you to call any other API object.



Last example, to configure the Microsoft Graph API, we use an OAuth 2.0 authentication with some specific settings which is described in the online API documentation.



Mail



The "Mail" connection string is unique to Fuzible. The "Connection String Assistant" can be used to help build it, but in short, the following parameters are required:

SERVER_SEND: SMTP URL

SERVER_RECEIVE: POP or IMAP URL

GET_PROTOCOL: The reception protocol: IMAP or POP

AUTH_PROTOCOL: The security protocol: TLS (10,11,12,13), NONE

PORT_SEND: SMTP Port

PORT_RECEIVE: IMAPT/POP Port

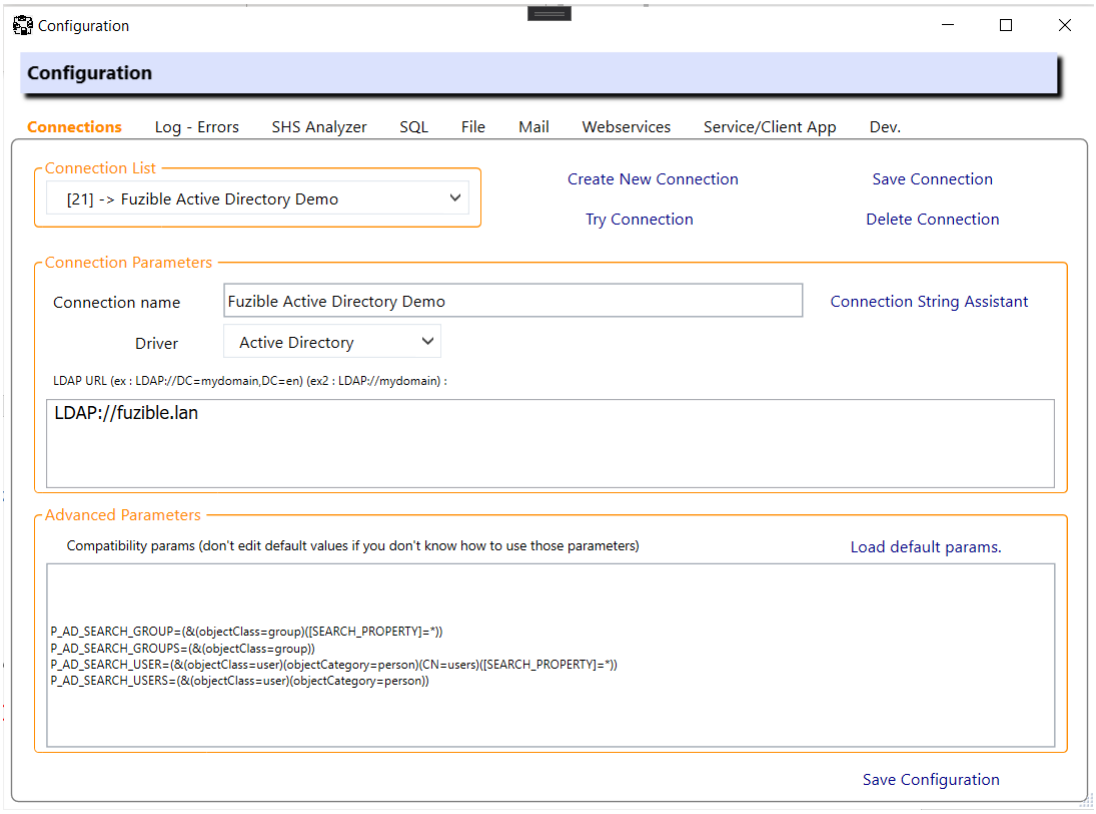
USERNAME: the associated email address

PASSWORD: the password associated with the email address

SSL: 1 or 0

You can also set up a proxy.

Active Directory



The connection string is rather simple, here are two examples:

LDAP://DC=fuzible,DC=fr

LDAP://fuzible.lan

In addition to the connection string, you can modify compatibility settings:

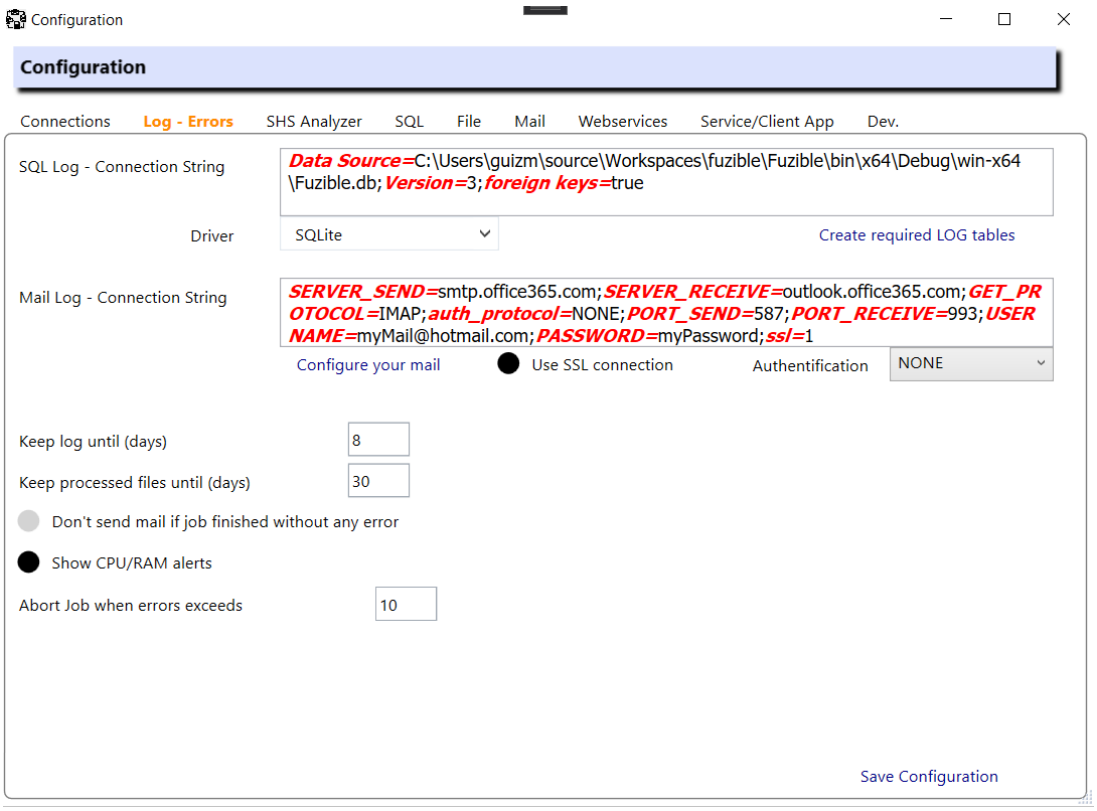
P_AD_SEARCH_GROUP	Search query for a specific group	SEARCH_PROPERTY will be replaced by the value set in the Job (see below). This is the "key" field available in the AD that defines the uniqueness of a group
P_AD_SEARCH_GROUPS	Group search query	
P_AD_SEARCH_USER	Query to search a specific user	SEARCH_PROPERTY will be replaced by the value set in the Job (see below). This is the "key" field available in the AD that defines a user's uniqueness
P_AD_SEARCH_USERS	User search query	

Log

This tab sets up all the software's LOG options. By default, ". TXT" LOG files are stored here :

C:\Users\Public\Documents\Fuzible\LOG

… but it can also be integrated into a database, or sent by email.



SQL Log

It is strongly advised to change SQL instance to store LOG, by default, it is the local Fuzible database that is used, but if you want to manipulate/get the LOG from elsewhere, you should use a network database (MySQL, SQL Server, Oracle, Postgres...)

In this case, the program will automatically create the required SQL tables in this new instance.

Mail Log

By default, this field is empty. This connection to an inbox allows you to send Job reports to one or more email addresses. The connection is configured like any connection (as seen above).

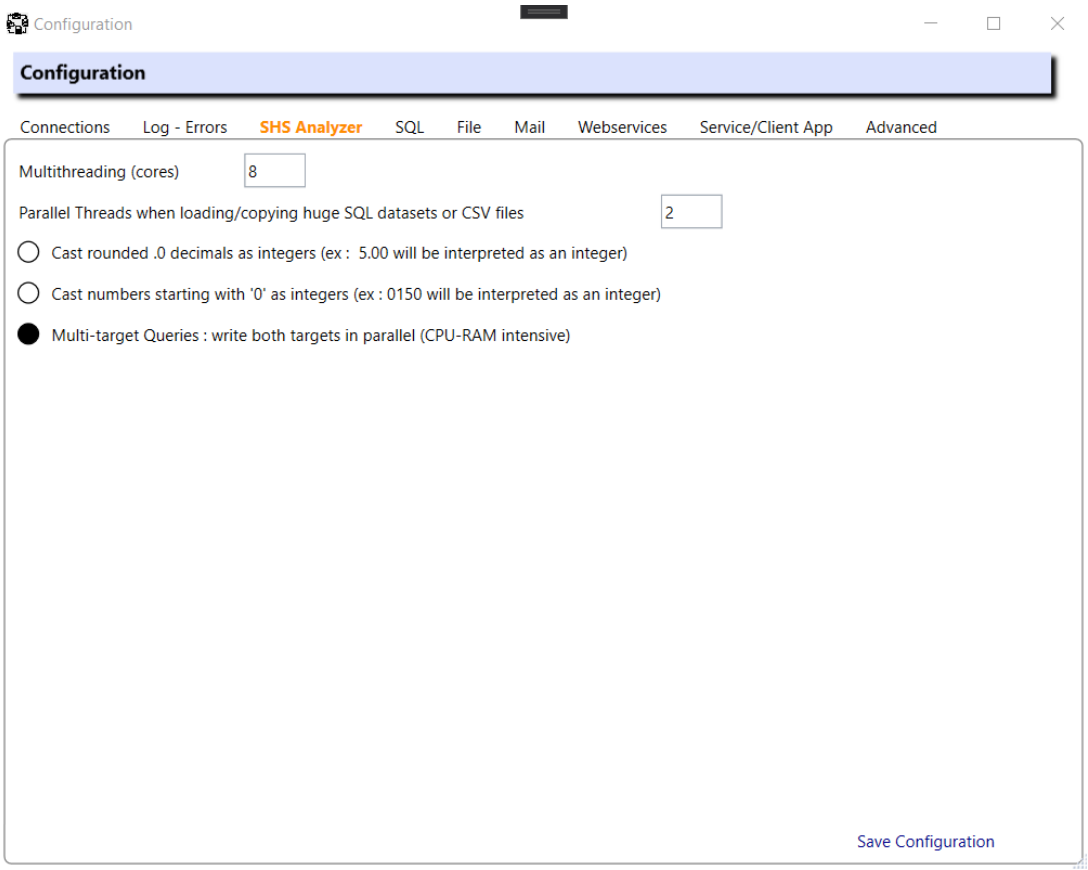
Options

Abort Job when errors exceeds	If this number is exceeded during the execution of the Job, then it is interrupted
Keep log until (days)	The LOG files and SQL entries can clean themselves beyond a number of days you can set here
Keep processed files until (days)	With a "File" connection string, Fuzible can clean up the directory by erasing files that are too old. The number of days of file preservation is defined here.
Don't send mail if job finished without any error	If the job is set up to send an e-mail at the end of the process, and if it went smoothly, you can avoid the mail to be sent
Show CPU/RAM alerts	When the CPU reaches 100%, the warning is logged, allowing you to check the need for additional resources on the server/computer that hosts the app

SHS Analyzer

Fuzible contains a data analysis engine: it analyzes all the source data, and can then create SQL tables as accurately as possible when they are non-existent and a Job uses a database as a Target.

These operations are very resource-intensive and can easily solicit the machine at 100%. To avoid this saturation, Fuzible can be set to use only a part of the available resources.

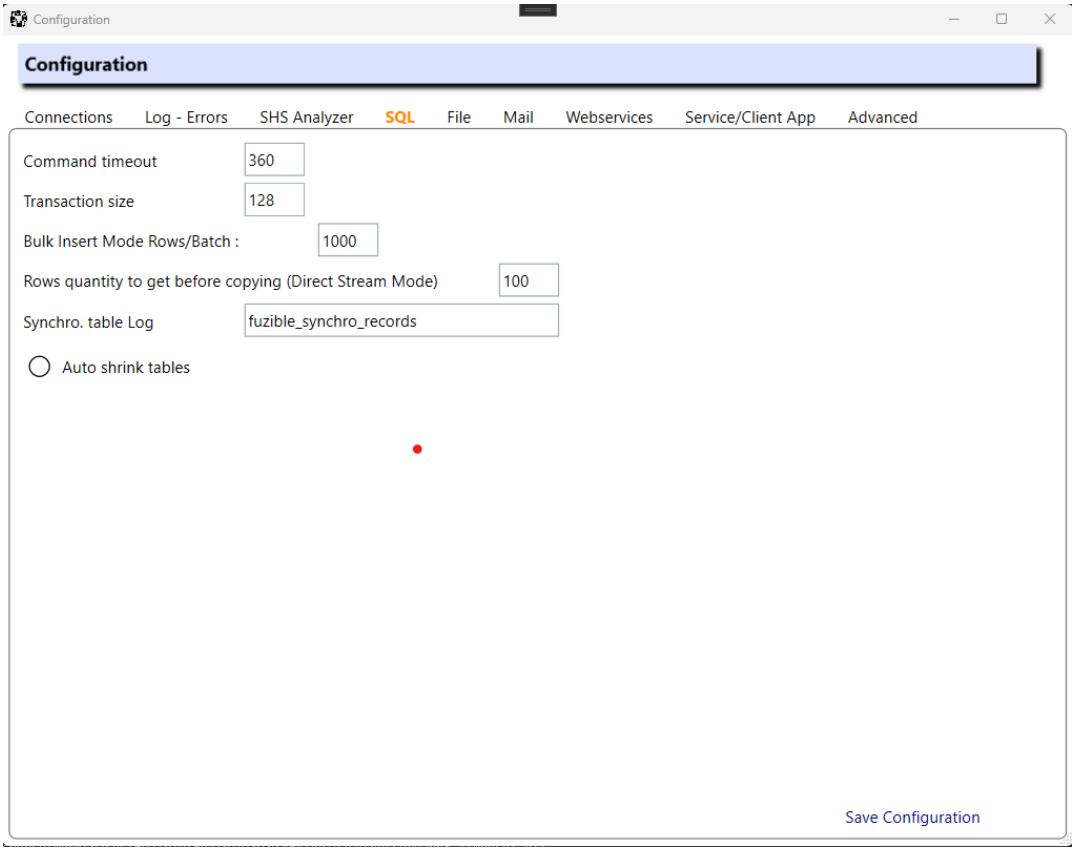


Options

Multithreading	Number of CPU cores that can be used by the software (minimum - 2)
Parallel Threads when loading/copying huge CSV datasets or CSV	Fuzible usually loads the whole Source Data in RAM, and then proceed with the Target replication/synchronization. In case of huge CSV or SQL tables (millions of rows), it loads data in smaller chunks. That way of loading data can be performed using more than one thread to increase speed : the next chunk can be loaded while the last one is copied in the Target. The more you add threads, the more you'll need CPU power and RAM.
Cast rounded .0 decimals as integers	When analyzing numerics, Fuzible can consider 5.00 to be an integer rather than a decimal
Cast numbers starting with '0' as integers	When analyzing numerics, Fuziblz can consider values like 0546 or 0000477 to be integers rathen than strings
Multi-Target Queries: Write both targets in parallel	For a Job using multi-target feature, you can choose to the 2 targets in parallel for better performances (requires LOT of computing power)

SQL

All the settings for the "SQL" connections are grouped here.

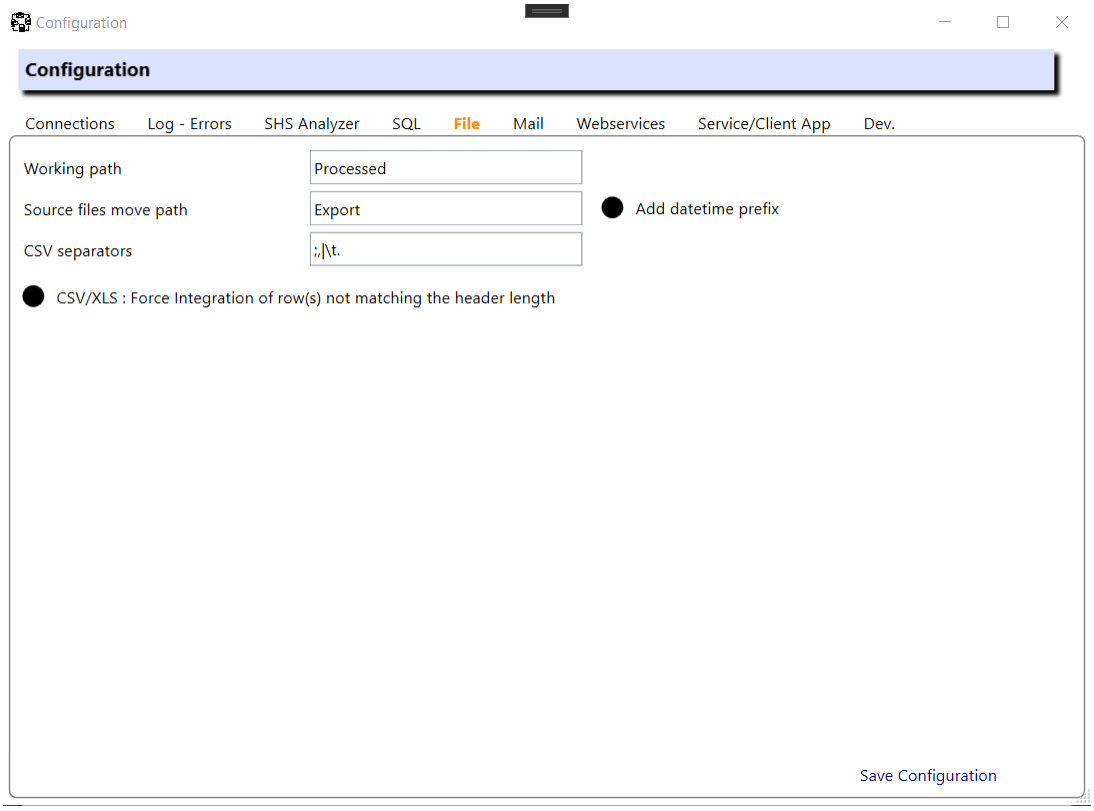


Options

Command Timeout	Timeout to execute an SQL command
Transaction size	When a Job whose target is an SQL database, "INSERT/DELETE/UPDATE" statements are framed in transactions (which allows a ROLLBACK if the query fails). The number of statements / transactions is to be defined here. The larger the number, the more the target database resources will be solicited
Bulk Insert Mode Rows/Batch	Applies when configuring a Job using the “Bulk Insert” mode. It defines the amount of rows you want to copy at one. Ie : if there are 10 000 rows to insert and you did choosed a 1000 rows batch, 10 Bulk batches will be sent to the target Database.
Row quantity to get before copying (Direct Stream Mode)	The « Direct Stream » feature is configured in the Job and is mostly used when querying huge SQL tables (millions of rows) : To avoid memory issues, data is loaded and transferred to target in small chunks. You can set here the length of each chunk (rows quantity). Increasing the value requires more RAM.
Syncho. Table LOG	For Jobs running in "Synchronization" mode, synchronization statuses are stored on a separate table (for consultation and information). You can choose the name here.
Auto shrink tables	An option to clean a Target table after processing it.

File

All the settings for the "FILE" connections are grouped here.

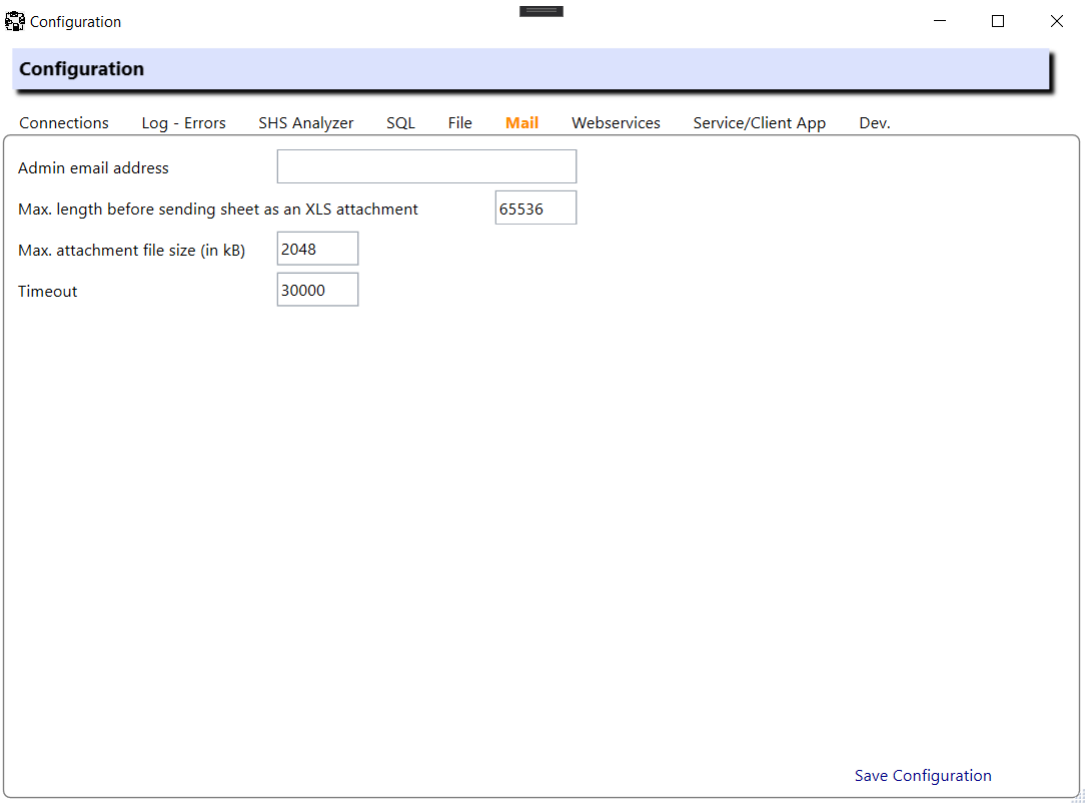


Options

Working path	When a Job is set with a "File" Source connection, and it has been set up to move those files when the Job is finished, the directory in which they are moved is defined here
Source files move path	Unused for now
Add datetime prefix	In addition to moving files at the end of a Job, these files can also have a prefix in the form of "YYYYMMddHHMMss_Myfile.xxx" to eventually facilitate their subsequent search (if needed)
CSV separators	List of accepted CSV separators ("\t" means "tabulation"). You can add more if you are dealing with files using some other separator.
CSV/XLS : Force Integration of row(s) not matching the header length	If for some reason your source files have inconsistent row length, you can force the integration of those rows or bypass them. In any case, a “WARNING” LOG message will be triggered

Mail

All the settings for the "MAIL" connections are grouped here.

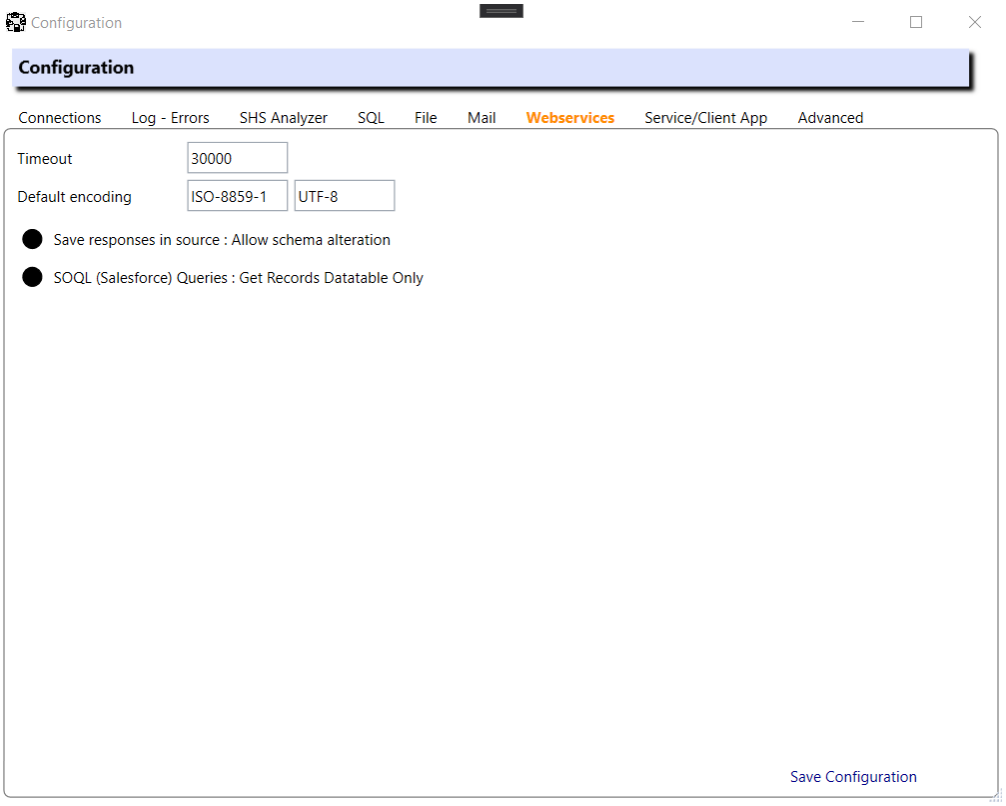


Options

Admin email address	The program administrator's email (will be cci'ed of any mail produced by a Job)
Max. length before sending sheet as an XLS attachment	When a Job is configured with an email address as a target, you can choose the data to be included in the mail body. However, if this content is too big, Fuzible can, instead, create an EXCEL file that will then be attached to the mail. This setting shows the maximum number of characters from the Data Source before the Job switches to "attachment" mode.
Max. attachment file size	Maximum size of an attachment. Beyond this limit, the attachment will not be sent
Timeout	Timeout to run an operation on the mail server

Webservices

All the settings for the "WEBSERVICES" connections are grouped here.



Options

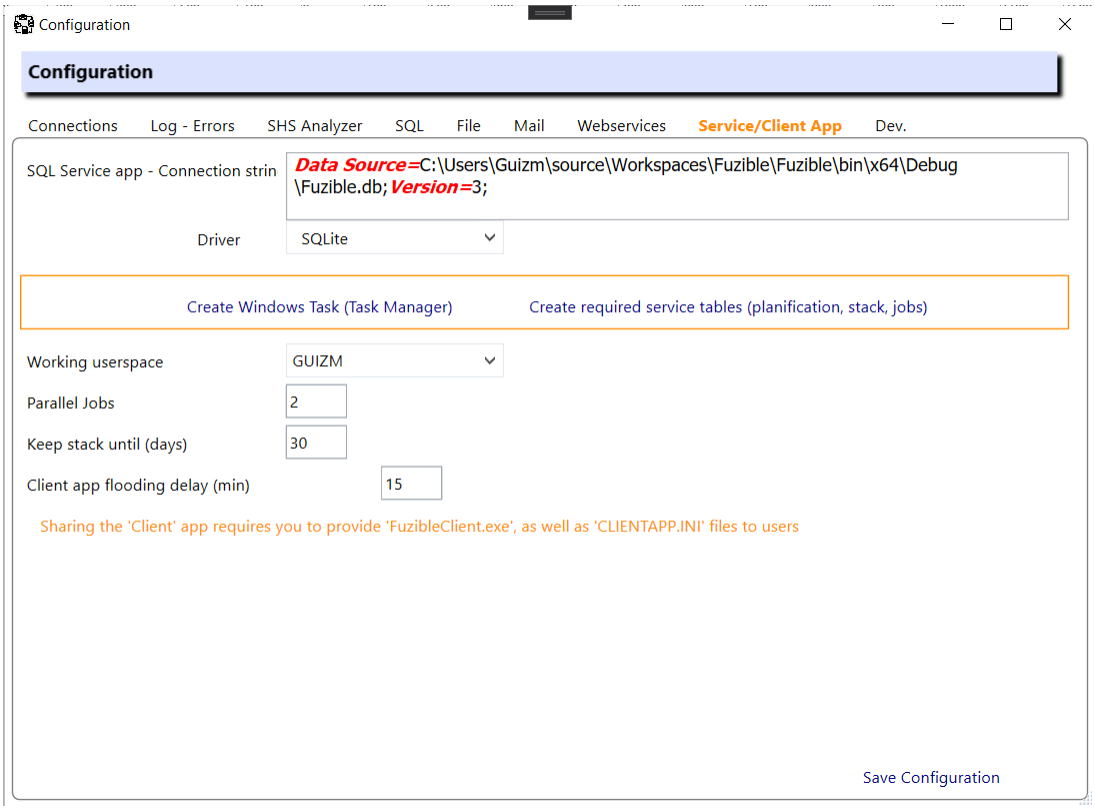
Timeout	Timeout to run an operation on the remote server
Default encoding	When A WS's data is processed, the program determines the encoding from what the server answers. If this information is unavailable, a custom encoding may be forced by default. 2 values to indicate: the 1st for the "REST API's" and the 2nd for the Nuxeo API.
Save responses in Source: Allow add-alter columns	When sending data to a webservice, this one sends back an answer (in XML or JSON format). Those answers can be retrieved and stored in the Source database (if SOURCE-BDD) or as a file (if other). In case of a Database, you could allow Fuzible to modify the SQL tables if the data returned by the API is not compatible with them.
SOQL Queries : Get Records Datatable Only	This setting allows, when using the SOQL language, to retrieve only the records datatable : will not get the additional tables : attributes and query summary

Service/Client App

Fuzible comes with a "Service/Client" module.

The "Service" background application is the subject of a dedicated paragraph, but in short, it executes Jobs that are invoked either by a user (via the "Client" application) or by the Orchestrator (Jobs Orchestrator)

The "Client" app is a mini-application that simply allow any user to trigger a Job execution remotely, whenever he wants.



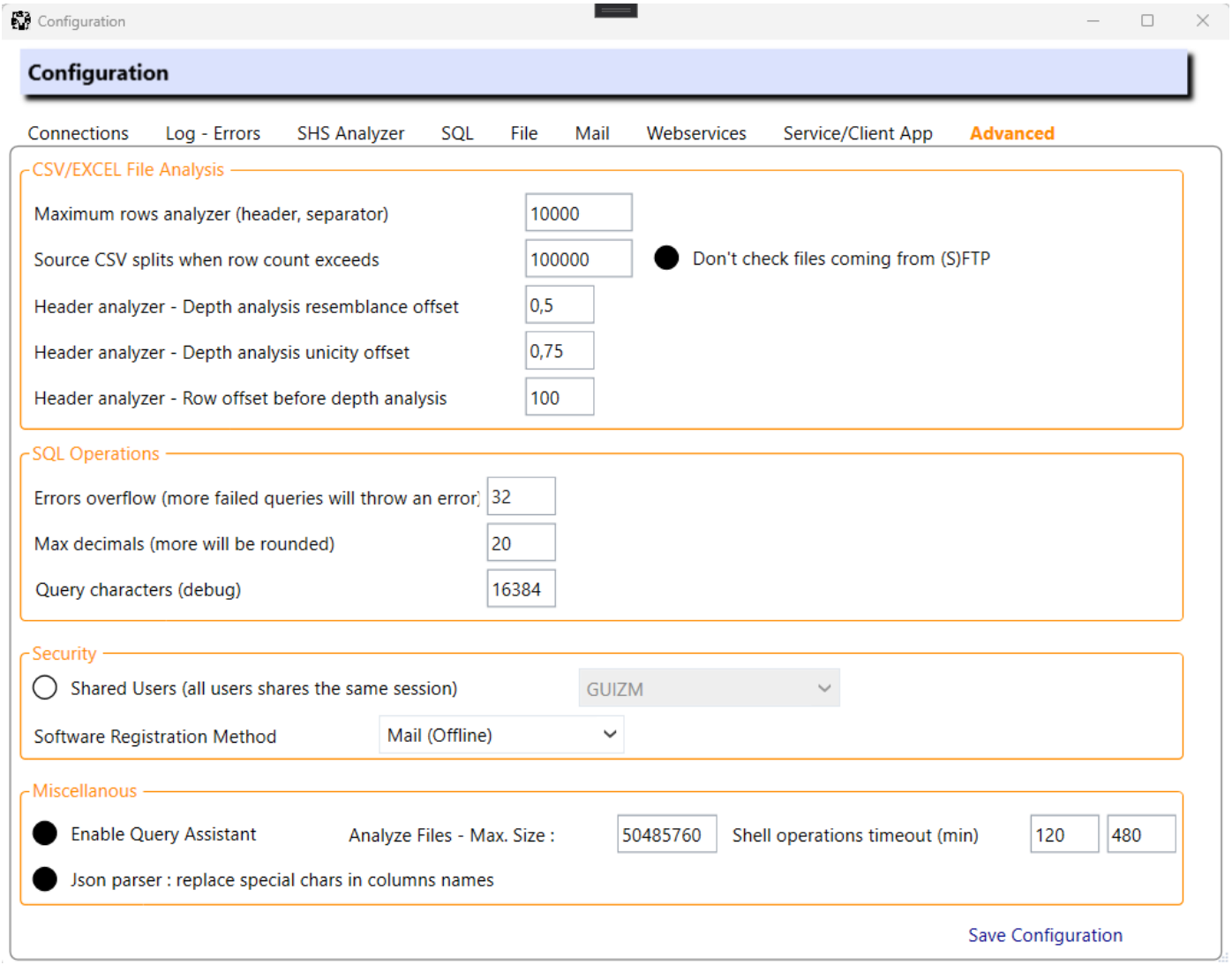
Connection string: By default, Fuzible uses the local SQLite connection, but this way of working is not really recommended. Indeed, the "Service" application is intended to communicate with the "Client" application, distributed to any user. In a network environment, you probably do not want anyone to have the Fuzible network path opened and accessible to anyone. On the other hand, your network can be configured so that client computers can make calls to a database instance.

Options

Create Required Service Tables	This button allows you to automatically create all the required SQL tables for the service app to work properly (in case of a change in connection string)
Create Windows Task	Creates the "Fuzible Service App" task in the Windows task manager.
Working User	This is the account the "Service" app uses to work. This account corresponds to one of the Fuzible users: the service application can only interact with one of the users, to avoid anarchic management of the orchestration and Jobs made available to the Client application.
Parallel Jobs	The Service app detects the Jobs invoked as they go along. It can run several in parallel but beware of the risk of overdlow. Here you choose the number of Jobs it can launch in parallel: This setting should be based on the resources allocated to the server/computer that runs the application.
Keep Stack Until (days)	Retention time before the Jobs stack LOG must be cleaned.
Client App Flooding Delay	The "Client" app allows users to remotely launch Jobs. They could trigger the same Job several times, flooding the system. This setting allows you to set a delay between 2 successive launches of the same Job, in order to prevent them from "spamming" the queue and overloading the system.

Dev.

Here are several parameters related to how the program internally works. Even if you would probably never need to change those settings, I choose to make them available. However, it is not advised to change them without understanding how they work, it could compromise your existing Jobs. The online forum may allow you to chat with other users about it.



XLS/CSV maximum rows analyzer	Fuzible automatically scans the EXCEL and CSV files to determine if the first line is a header. In the case of files with a large number of lines, the ent are analyzerdoes not necessarily need to analyze all the lines to detect it. The maximum number of lines to be analysed can be set here
Source CSV split when row count exceeds	If the source is a CSV file, the program systematically counts the number of lines of it. If this number exceeds a certain amount, Fuzible can process the file into several chunks to avoid overload of the server's RAM. The value indicates the maximum number of rows contained in each chunk.
Don't check files coming from (S)FTP	In connection with the previous option, if the files come from a server (S)FTP, it is possible to bypass the line count, because this analysis requires downloading the file, which can severely penalize the performance of the program.
Header Analyzer - Depth Analysis Resemblance Offset	A threshold that determines the percentage of resemblance between the first line of the analyzed file and all the others. The resemblance is calculated internally by doing several tests on the file. The scan can be displayed if the Job is configured in "Debug" mode
Header Analyzer - Depth Analysis Unicity Offset	When automatically analyzing the contents of an EXCEL or CSV file, Fuzible determines the uniqueness of the first line of the file compared to the others.

	- If the percentage of uniqueness exceeds thethreshold, and in addition, the percentage of resemblance of the first line is above the "Offset Resemblance" threshold, it is considered that the first line is not a header - If the percentage of uniqueness exceeds the threshold, and in addition the percentage of resemblance of the first line is below the "Resemblance Offset" threshold, the first line is considered a header - If the percentage of uniqueness is below the threshold, and in addition the percentage of resemblance of the first line is less than or equal to the threshold "Resemblance Offset", the first line is considered a header - If the percentage of uniqueness is below the threshold, and in addition the percentage of resemblance of the first line is above the "Resemblance Offset" threshold, it is considered that the first line is not a header
Header Analyzer - Row offset before depth analysis	In the case of files with few lines, header detection can be tricky (sample too small to calculate a percentage of resemblance between the first line and all the others). Below the threshold (number of lines) entered here, the program goes into "end" mode. It will finely analyze the file, and play on the 2 parameters mentioned above to determine the header
Max Decimals	In "target- BDD" mode, defines the maximum number of decimals tolerated when inserting data(where source data would have for example 35 digits after the comma and one wishes to limit this amount)
Errors Overflow	If an executed query didn't work, Fuzible marks it as a "warning." You can set here a threshold of queries that have not resulted from whichthe Fuzible considers it to be an error and no longer a "warning" (will trigger an error message)
Query characters (Debug)	When a Job is set up with log in "Debug" mode, and the Job target is a BDD, all INSERT/DELETE/UPDATE queries will be entered into the DEBUG file. The volume can be considerable, so it is possible to limit the amount of characters built into the LOG
Shared Users	By default, any Windows user has its own Fuzible session : connections, configuration, jobs are not shared with the others. In that mode, you can still import Jobs from another session and load another userspace but in a « read-only » mode only. In your organisation, you could need a shared session : any user will be routed to the one who's choosed here. The whole program is then shared with all computer users. It implies some confidentiality compromises (ie : connection strings are visibles by any Fuzible user)
Software Registration Method	How to communicate with Fuzible's server when you want to save the program. You should leave "Webservice" by default. The "Mail" mode will only serve in case you are unable to communicate with the server (offline mode)
Enable Query Assistant	Turns the assistant on interface queries (colorization, consistency controls, input proposals) is activated ordisables. The assistant can consume a lot of resources because in case the Source is a database, he asks him to know the tables available and the fields of each table. If it's a file-type source, it scans the directory to find the names of the files, and scans each of them to find the headers.
Shell Operations Timeout	When a Shell Pre or Post-Job command is scheduled, the maximum default execution time is 60 minutes. Beyond this time, the task is interrupted. So we can intervene here on this parameter This setting is also used by the "Service" app. When she performs Jobs, she casts Fuzible in a set-up manner. The program is then subject to the same Timeout rules
Json Parser: Replace special tanks in columns names	When a JSON file (or webservice response) is interpreted, column names sometimes contain special characters: it can be decided here to replace them with a more conventional character(the underscore)
Analyze Files – Max Size	When using the Query Assistant with "FILE" as the Job Source Connection, it will parse the file you are querying to find all the columns and add them to auto-completion system. If the file reaches a max. size (in kb), it won't be analyzed because it will consume too much power (CPU/RAM). In case you don't know the available columns in the file, you should first perform a "SELECT * FROM…" to show available columns and use them in your Query after that

Tools

Export Job (XML)

This menu allows you to export a Job using XML format: This extraction contains all job settings, associated connections (as well as those that may be called by script fields), and queries. The file is encrypted so that it can only be imported into another environment if the user knows the password: it is the job's default password.

Import Job (XML)

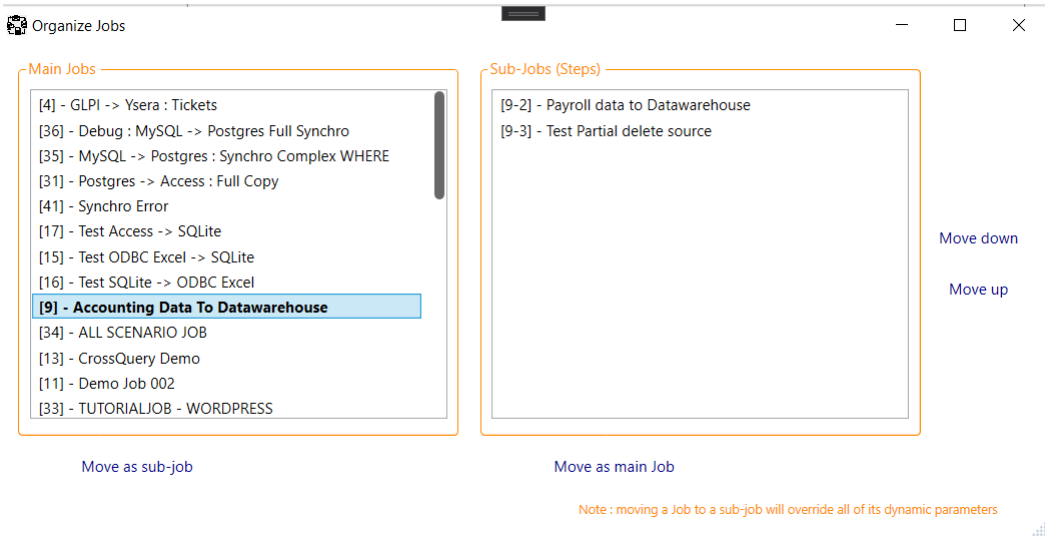
This menu allows you to import a Job using XML format: Integration into your environment includes creating associated connections with the Job, settings, and queries. The Job password is required to perform the importation.

Reorganize Jobs

You can reorganize Jobs here. Indeed, it is possible to create "multi-step" jobs (which are no more or less than several jobs launched one after the other.

This menu allows you to:

- Drop a main job to another job as a step.
- Extract a "step" in a "multi-step" Job to put it as a single Job.
- Reorganize the steps order in a "multi-step" Job.



Multi-step Jobs are bolded. When you click on one of them, you see the list of sub-jobs.

Move as Sub -Job

Moves a Job into another one. It will then become a sub-job. If it is moved to a Job that already has sub-jobs, it will be positioned last.

Move as Main Job

Extracts a sub-job to make it a main job. You will be asked to set a password for him.

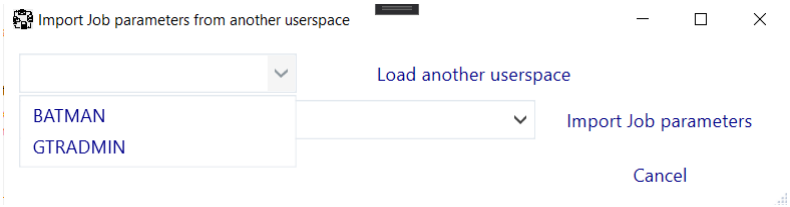
Move Down/Up

Moves a sub-job to change the execution order of a "multi-step" Job.

Note: If you move a Job to a sub-Job, and automatic launch schedules are associated with it, you will be alerted, and the app will ask to delete or maintain this schedule.

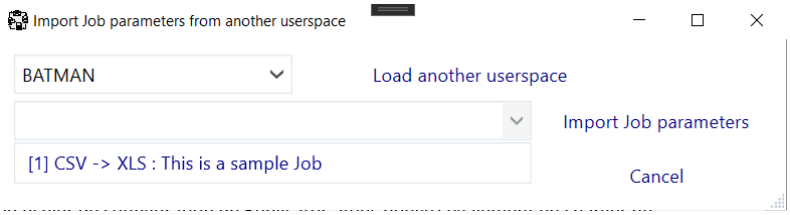
Import External Job Parameters

You can import a job from another user here (you have to know the password)



Shows the list of available users.

... then the list of Jobs associated with it:



This information is read from the Fuzible's configuration file. You can also load another "Load Another Database file" to retrieve Jobs from other environments.

Note: Importing a "multi-step" Job is not possible. They are displayed as normal Jobs, with the difference that they appear in black and not blue.

After entering the password, the Job is imported (filling the settings fields) and the associated connections, imported automatically if you do not have them in your list.

If a connection ID is mentioned in the Job queries (cross-join queries, multi-target), they are also imported and transcoded automatically. You don't have to do anything to set up the Job again.

After import, you then have to click "Save configuration" to create a new job, based on these new parameters.

Load another Userspace (read-only)

You can load the whole configuration of another user, for example to launch one of its jobs.

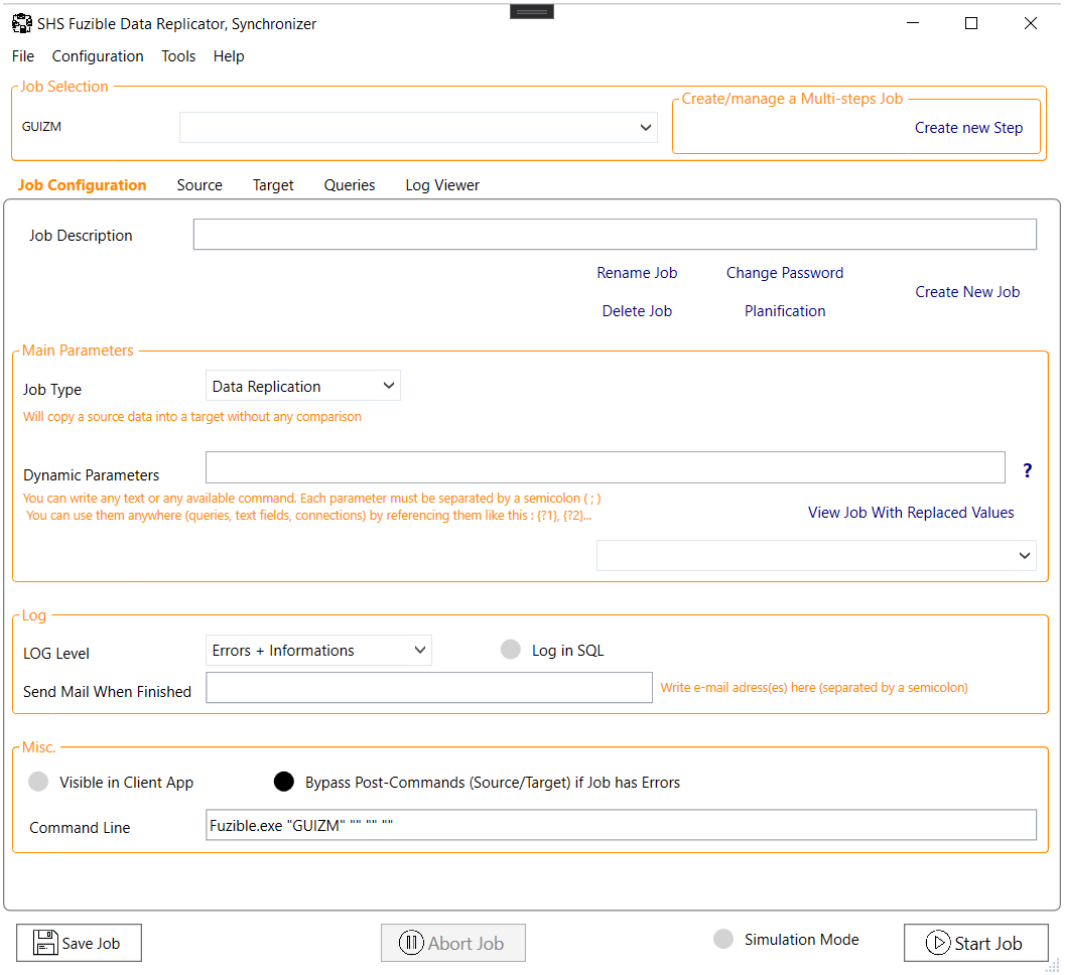
On the other hand, it is impossible to change any parameter of its configuration and its jobs. Only the dynamic parameters field can be changed. This allows you to benefit from the dynamic setting, which may be necessary.

You can also change the orchestration of his Jobs, a crucial feature because if you urgently need to have an hand on the orchestration and the account owner is not here, you will need to be able to change the setting for him.

Job Creation

Job Configuration tab

Time to create Jobs.



Set up the general settings of your new Job here.

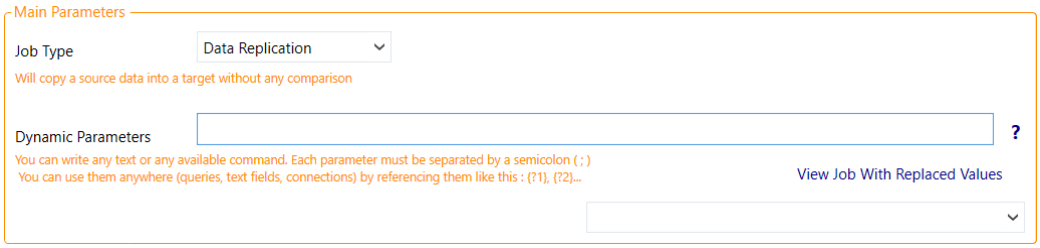
Options

Job description	A text field that allows you to describe the purpose of the Job. This field also serves as an object for a Job whose Target is an inbox.
Job Type	Data Replication (copy) or Data Synchronization ➔ See "Job Type"
Dynamic Parameters	Dynamic variables: allows you to dynamically change the behavior of some Job parameters/queries. Dynamic Parameters %YYYY%MM<1M;[4] Each variable is separated by a "; " Up to 9 variables are accepted ➔ See "Script language"
Visible in Client app	Check this if you want the Job to be available in the Client App (can be then remotely launched)
Abort next steps on errors	During a multi-step Job execution, you may want to abort next steps if errors are detected
Bypass post-commands if job has errors	Avoid post-job commands if it has encountered errors ➔ See "post-commands"
LOG level	The desired LOG level LOG Level: Errors + Informations Send Mail When Finished Errors Errors + Informations Errors, Informations, Details

	- ERRORS: You will only receive WARNING and ERROR messages from the Job - ERRORS - INFORMATIONS: You'll also have informative messages that indicate progression - ERRORS, INFORMATIONS, DETAILS: You will also receive details of internal operations : calculations, information related to data analysis...
Log in SQL	If the SQL connection string is set up (Program Configuration), it is possible to define here if you also want the Job to be logged into that SQL database
Send mail when finished	You can enter e-mail addresses here (separated by a comma) : A report of the Job will be sent to these people (report containing the execution status, as well as the LOG files
Command Line	Code that allows you to launch the Job without launching Fuzible (silent mode) Command Line: Fuzible.exe "GUIZM" "[34]" "88BuRGfN+VN8Ovvp0qCebQ==" "%DD%MM%YYYY" Arguments: - User - Job Number - Job password - Dynamic parameters (those seen in the Dynamic Parameters field)

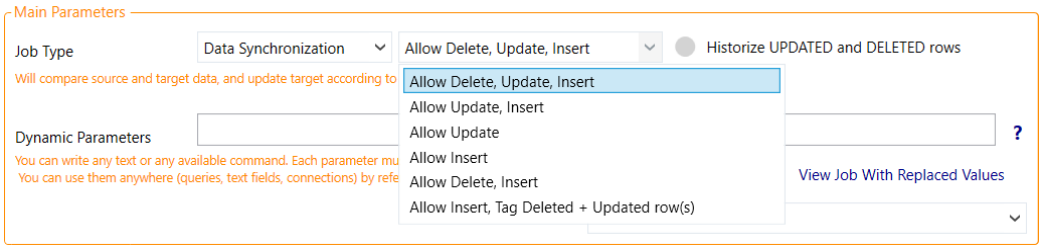
Job Type

Data Replication



This mode is available for any type of target. It consists of simply copying data from a Source connection to a Target.

Data Synchronization



This mode is only available for AD, File and database Target connections.

It compares what is sent from point A to what already exists at point B. To do this, Fuzible will dynamically transcode the Source query to be comprehensible by the Target (assuming that the source and target may be or different or/and share different column names)

The comparison is based on the search for a Primary Key.

Fuzible uses several methods to find a primary key and proceeds from the simplest to the most complex:

In SQL mode

1/ Searching for the primary key by querying the BDD schema (Target first, then Source)
2/ Search for unique keys by querying the BDD schema (Target first, then Source)
3/ Automatic detection by analyzing all the combinations of columns and values that exist in the Source (this method can take a lot of time if the number of rows is huge)

In FILE mode

The automatic detection mode is used.

In ACTIVE DIRECTORY mode

The Primary Key field is defined in the "Target" tab

It is then possible to define the behavior on the Target:

- Insert new Source rows missing in the Target.
- Update rows that already exist in the Target, but whose content is no longer the same in the Source.
- Remove rows that are in the Target, but no more in the Source
- TAG: If data is to be deleted in the Target (not existing in the Source anymore) you can chose to keep it but create a "SYNCHRO_TAG" field using the "D" value (DELETED) on rows that should have been erased Similarly, the updated rows can be tagged "U" (UPDATED)- however, they will be updated.

  The "Historize Updated and Deleted Rows" option creates or fills a table next to the target table (with the _hist extension) that contains all rows that have either been deleted or updated during sync. It basically creates a table with the exact same schema and adds a new index column to it, as well as a timestamp column.

Language Script

All Fuzible "text" fields accept scripted parameters that will be interpreted and replaced by associated values.

Those can be made Dynamic:

- A connection string (for example, changing the server, the user, the database, the path of a file...)
- Queries (for example, making dynamic filters)
- Add additional, custom columns to content.

By design, any "text" area of Fuzible understands and interprets script language. The script zones are framed by brace brackets.

Here are the variables that can be used:

- ?1, ?2, ?3...: parameters from the "Dynamic Parameters" field
- %MM: month number on 2 characters
- %YY: 2-character year number
- %YYYY: 4-character year number
- %DD: 2-character day number of the month
- %WW : 2-character week of the year
- %HH: Time of Day on 2 characters
- %mm: minute on 2 characters
- %SS: second on 2 characters
- %DTTS / DTTSMILLI: current date in Unix Timestamp format
- <XM : removes X months to MM (if existing) (ex : %MM<2M)
- <XY : removes X years to YY (ou YYYY) (if existing) (ex : %YY<2Y)
- <XD : removes X days to DD (if existing) (ex : %DD<2D)
- <XW : removes X weeks to WW (if existing) (ex : %WW<2W)
- >XM : add X months to MM (if existing) (ex : %MM>2M)
- >XY : add X years to YY (ou YYYY) (if existing) (ex : %YY>2Y)
- >XD : add X days to DD (if existing) (ex : %DD>2D)
- >XW : add X weeks to WW (if existing) (ex : %WW>2W)
- %USER: The user's name connected to the app
- %QUERYTARGETNAME: The name of the target.
- -%CT1, %CS1, %CT2, %CS2... : If specified in the dynamic parameters list, this setting will be replaced by the returned value from a Source pre-command (%CS1) or a Target pre-command (%CT1)

Examples (base: March 20, 2018, consider that ?1 = 100 et 2 = 'TEST')

SELECT * FROM MONFICHIER_{ %MM%YYYY}.csv	SELECT * FROM MONFICHIER_032018.csv	
SELECT * FROM MONFICHIER{%MM_%DD_%YYYY}.csv	SELECT * FROM MONFICHIER03_20_2018.csv	
SELECT * FROM MONFICHIER_{%MM%YYYY>3M}.csv	SELECT * FROM MONFICHIER_062018.csv	
SELECT * FROM MONFICHIER_{%MM%YYYY>3M>1Y}.csv	SELECT * FROM MONFICHIER_062019.csv	
SELECT * FROM MONFICHIER_{%DD<10D}.csv	SELECT * FROM MONFICHIER_10.csv	
SELECT * FROM MATABLE WHERE ID_TEST={?1}	SELECT * FROM MATABLE WHERE ID_TEST=100	
SELECT * FROM MATABLE WHERE ID_TEST ={%YYYY01<2Y_?1-?2}	SELECT * FROM MATABLE WHERE ID_TEST= 201601_100-'TEST'	

Additional note about the use of script language:

- In the Dynamic Parameters field, each setting must be separated by a "; " this character reserved for separation cannot therefore be used in a dynamic value!
- Date "codes" can be used as dynamic parameters:
  What if you're assigning "MM" value to {?1} will be replaced by 03 (March if it is March)
  You can then schedule more complex things, for example (it's March 20, 2018):
  {%YYYY01<2Y-?1} will then be transformed as **201601-_20TEST** if ?1 has been set with value _%DDTEST

Job Summary

In addition to this setting, when a Job has been created, you can see a couple of key information (at the top of the screen): Creation date, last execution, and status of the last execution.

When you save a Job, you will be asked to set a password. It prevents anyone to import your Job without asking your agreement, but it also serves to prevent unwanted users to launch it from the "Client" app.

This password can be changed, as is the name of the Job.

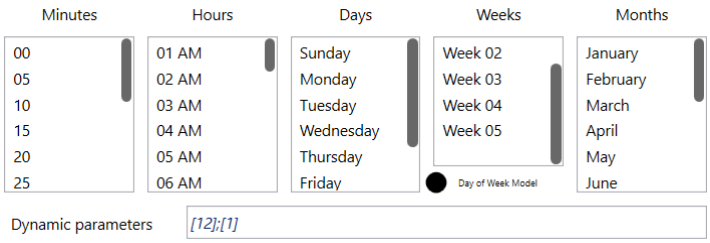
Orchestration

If the "Service" app is set up correctly, the Job Orchestration tool is available to you. The purpose of this is to launch Jobs on pre-defined dates/intervals.

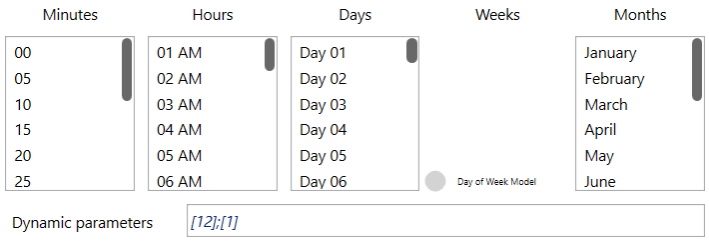
When you open a Job, you click "Orchestration," a new screen opens and allows to create, delete, and change your scheduling plan(s) for that Job.

Creating an Orchestration model works in two modes:

In the form of "days of the week"

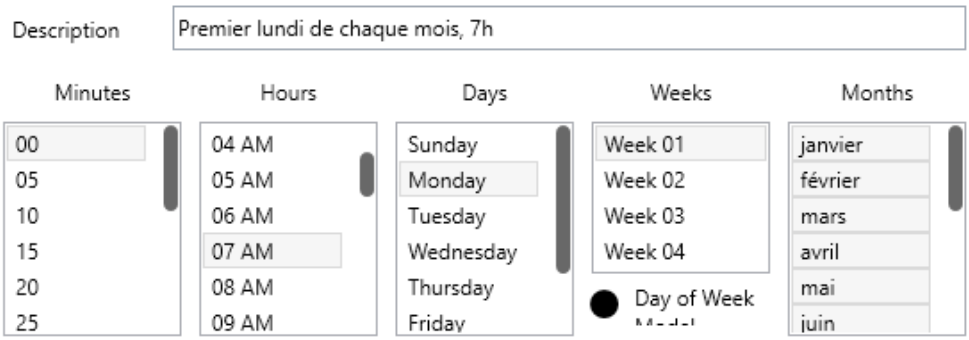


In the form of "days of the month"

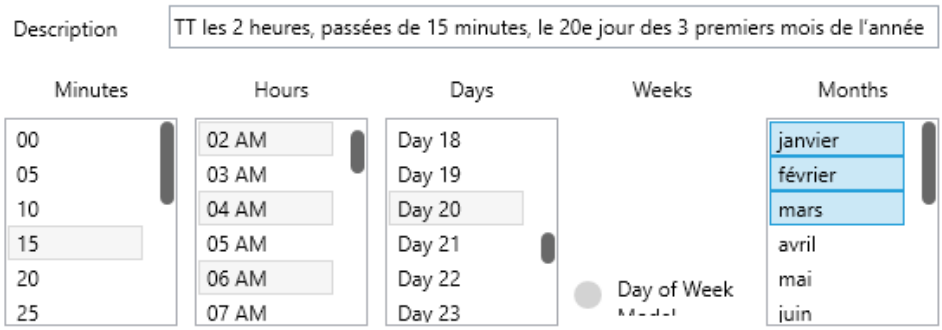


You will necessarily have to choose an item from each column to set an orchestration.

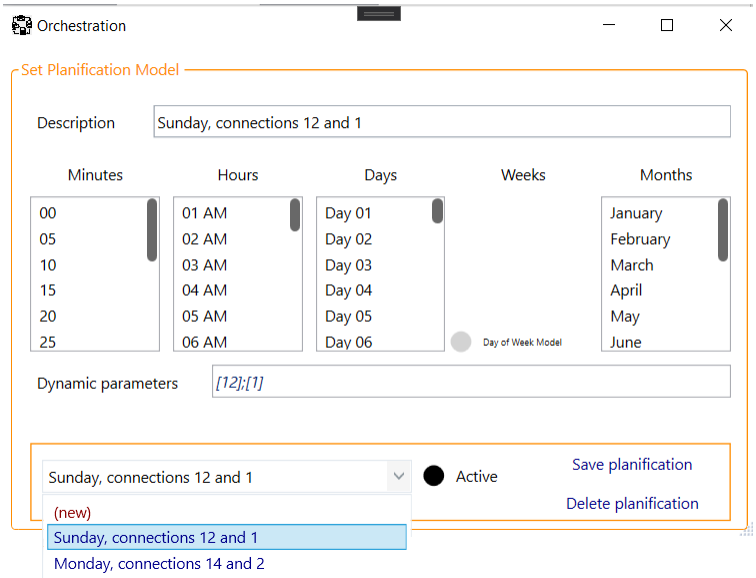
- If, for example, you want to start a Job on the first Monday of each month, at 07:00, you will set the schedule as follows:



- If you want to start a Job every 2 hours, past 15 minutes, on the 20th day of the first 3 months of the year, you will set the orchestration as follows:



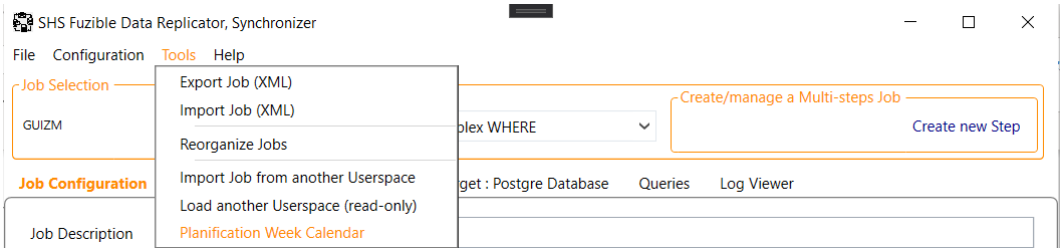
A planification must include a description, and optional dynamic parameters: These are, by default, deferred from the Job but you could, for example, set multiple schedules for the same job with a different setting each time (ex:, launch a data replication on a preprod DB at 07:00, and then launch that same replication on a production DB at 08:00)



You can obviously edit or delete any of the planifications. You can also choose whether to make them active or not.

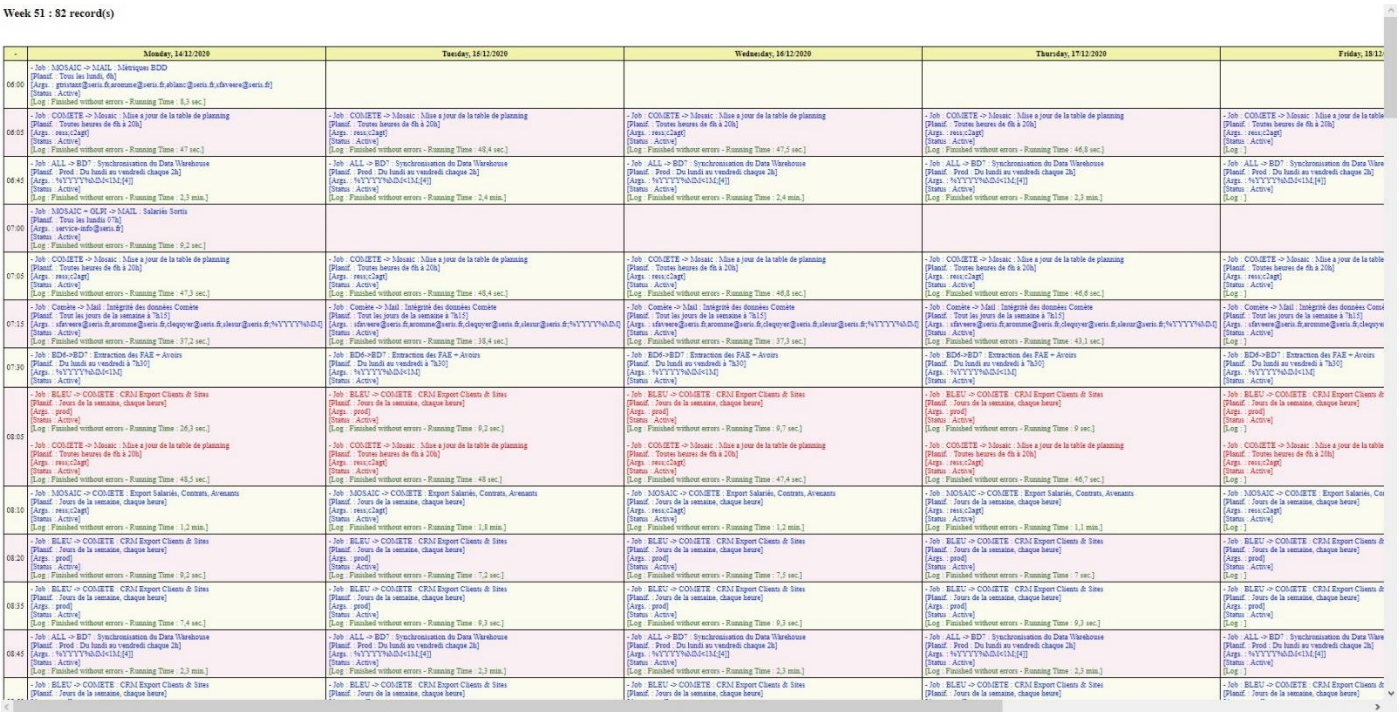
Planning Calendar

Fuzible offers you a complete visualization tool of the current week schedules.



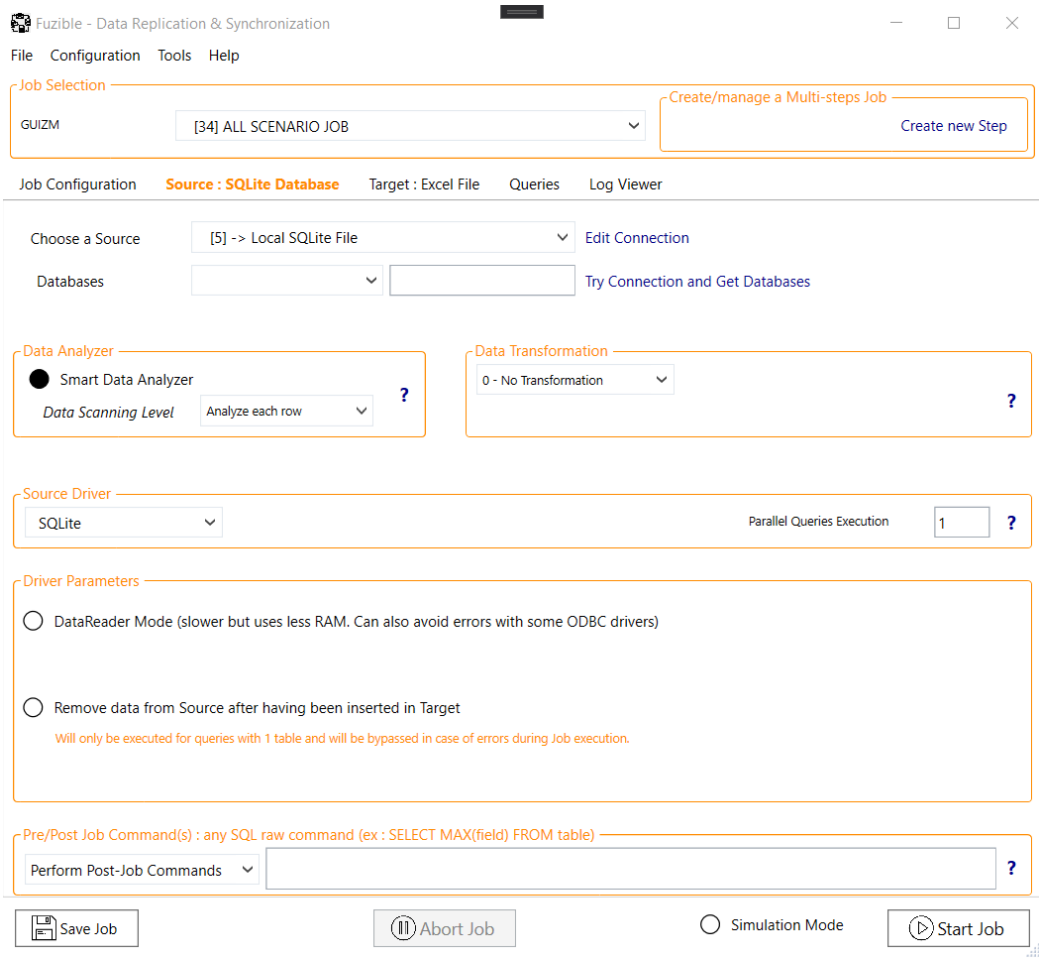
By clicking on this menu, Fuzible will generate a simple HTML file of the current week's schedules and display it in your default browser. This will give you a view of all the scheduled tasks and information on the unfolding of previous iterations.

Example of a calendar:



Source tab

Here you choose the Source connection. In the case of a database, you can choose to view the list of available BDDs and use one that is different from the one from the connection string.



Try Connection	A quick connexion check - BDD: checks the connection to the SQL instance and brings back the databases list - FILE: checks the existence of the path (or (s)ftp) and displays the list of available files - WEBSERVICE/MAIL: pings the server - ACTIVE DIRECTORY: checks AD domain availability
Parallel Queries Execution	If the Job has multiple queries, you can choose to run multiple queries in parallel. Beware of the resource consumption associated with these simultaneous executions.
Intelligent Data Analyzer	This is the data analysis engine, thanks to it, for example, Fuzible can, among other things, to automatically create SQL tables that do not exist with the most accurate data types, resulting from data analysis. If you're dealing with large datasets, the data analysis engine can be resource-consuming,and it's not always useful to analyze all the data. You can set either scan all the rows or all the X rows. ➔ See "Data Analyzer"
Data Transformation	Fuzible can transform source data, like a PIVOT operation would do. ➔ See "Data Transformation"

BDD

Driver Parameters

- (●) DataReader Mode (slower but uses less RAM. Can also avoid errors with some ODBC drivers)
  - ( ) Direct Stream Copy (parallelized read & write data stream)
- ( ) Remove data from Source after having been inserted in Target

  Will only be executed for queries with 1 table and will be bypassed in case of errors during Job execution.

DataReader Mode	This is an alternate way of getting Source data. It can be useful with ODBC drivers which sometimes uses buggy drivers. It's also less RAM consuming.
Direct Stream Copy	In DataReader mode, data is read row-by-row, it means that they can be transferred to Target by chunks (100 rows by default, can be set in program configuration / SQL tab). That method is useful as it parallelizes read and write operations, consumes less RAM, and most of the times, offers great performances. The downside is that in case of your Job's Target is a database, and Fuzible is asked to automatically create the target table, working with small data chunks may prevent the engine to create an accurate data schema. It is advised to use that feature if the Target Table is already created.
Remove Data from Source after having been inserted in Target	If you simply want to transfer Source data to the Target, this option allows you to delete the data that has been retrieved from the Source. If, however, the Job contains errors during execution, this step will be avoided. In addition, if the Source query contains multiple tables (joins), it will also be avoided.

MONGODB

Driver Parameters

- ( ) Remove internal MongoDB ID column from retrieved data

Remove internal MongoDB ID column from retrieved data	Each MongoDB collection identifies its records with an ID. This is shown as an additional column when retrieving data. This option allows you not to get this column

CSV file

Driver Parameters

Are the files to be processed zipped? (Enter the ZIP filename) [ ] Browse Post-process (Source Files) [Nothing]

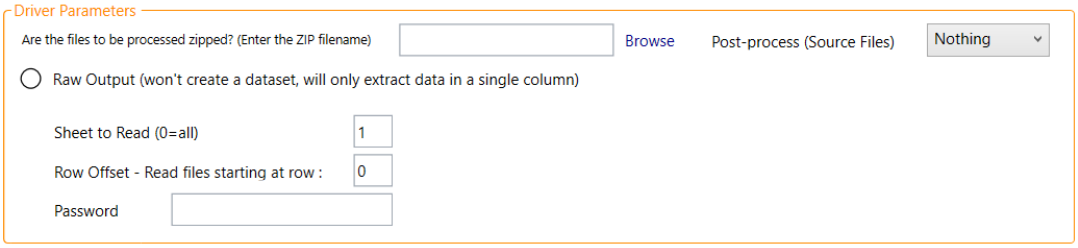
- ( ) Raw Output (won't create a dataset, will only extract data in a single column)

Row Offset - Read files starting at row : [0]

Read Multiple Files At Once - Name has to contain : [^] ?

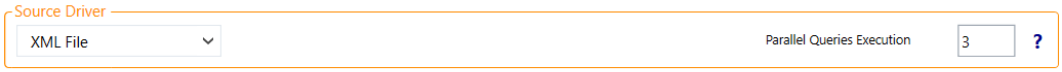
Source File(s) are zipped in	If the files you want to query are in a ZIP file, the filename is defined here (its name may be or contain dynamic parameters)
Post-Process	You choose what you want to do with source files once processed: 1 - Nothing: We leave them where they are. 2 - Move: They are moved in a sub-directory. 3 - Zip: Compact them into a ZIP file. 4- Delete: they are removed. These 4 options also apply in the case of network path and (S)FTP
Raw Output	Instead of creating datatable from the files content, it is simply extracted "raw", as shown in Notepad (for example)
Row offset	Tells Fuzible from which row he should start to read the file. ➔ See term "header detection"
Read multiple Files at once	Allows you to indicate a "pattern" for the file name: Fuzible will then get and merge all data from all files with a name matching that pattern

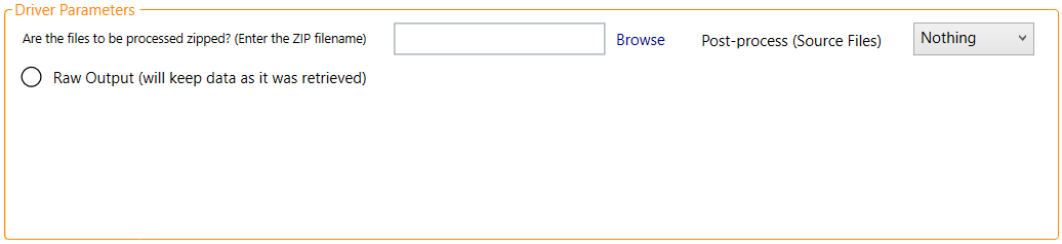
Excel file



Source File(s) are zipped in	If the files you want to query are in a ZIP file, the filename is defined here (its name may be or contain dynamic parameters)
Post-Process	You choose what you want to do with source files once processed: 1 - Nothing: We leave them where they are. 2 - Move: They are moved in a sub-directory. 3 - Zip: Compact them into a ZIP file. 4- Delete: they are removed. These 4 options also apply in the case of network path and (S)FTP
Raw Output	Instead of creating datatable from the files content, it is simply extracted "raw", as shown in Notepad (for example)
Row offset	Tells Fuzible from which row he should start to read the file. ➔ See term "header detection"
Sheet to read	Tells Fuzible the sheet index data will be retrieved from
Password	If the Excel file is password protected, this is where it should be indicated. On the other hand, if you try to write multiple queries using multiple Excel files, which don't all have the same password, you'll be forced to create multiple Jobs.

XML and JSON Files



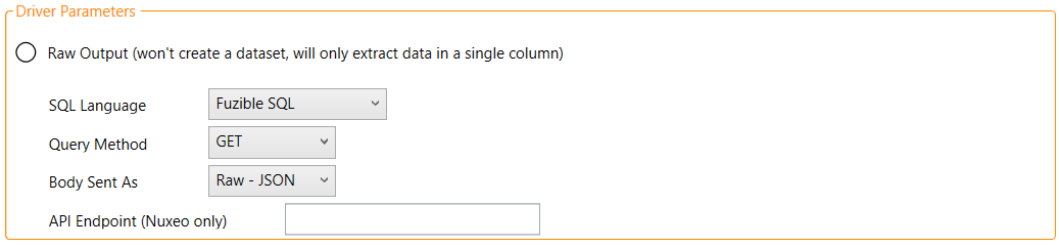


Source File(s) are zipped in	If the files you want to query are in a ZIP file, the filename is defined here (its name may be or contain dynamic parameters)
Post-Process	You choose what you want to do with source files once processed: 1 - Nothing: We leave them where they are. 2 - Move: They are moved in a sub-directory. 3 - Zip: Compact them into a ZIP file. 4- Delete: they are removed. These 4 options also apply in the case of network path and (S)FTP
Raw Output	Instead of creating datatable from the files content, it is simply extracted "raw", as shown in Notepad (for example)

Note on head detection

By default (« Raw Output » unchecked), Fuzible analyzes the contents of CSV and EXCEL files on its own and automatically determinates the presence of a header. Its analysis is based on a set of tests that make it reliable in 99% of cases.

Webservice REST

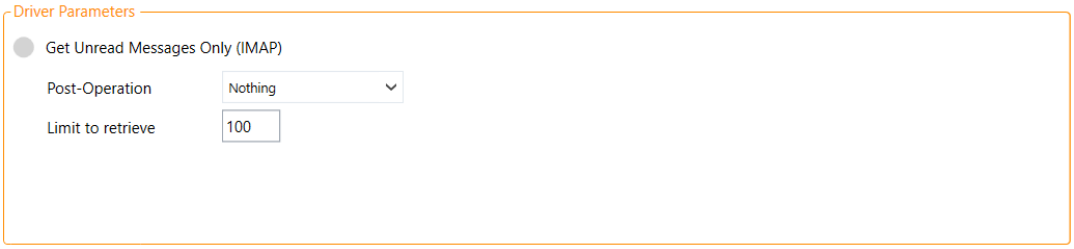


SQL Language	Some APIs can use their own simili-SQL language that can be used instead of "Fuzible SQL", which is the default engine. For example, the Salesforce CRM uses SoQL

Method	Data retrieval method: GET or POST
Body Sent As	You can send some body content in your queries to an API. Here you specify what kind of body it is: Form-Data, JSON, XML
Raw output	The responses returned by the API will be extracted « raw », the content won't be serialized as data table(s).

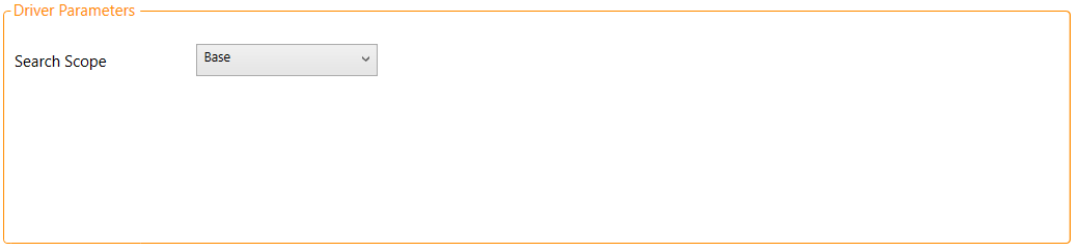
Note that depending on the connection "template" you choose (connection configuration), you may have a slightly different setting; for example, if you use the Salesforce SoQL "template", the settings are automatically set for you:

Mailbox



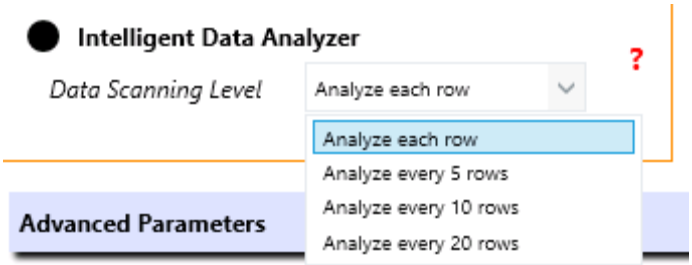
Get unread messages only	Chooses to retrieve only emails that have not yet been read (IMAP only)
Post-Operation	Defines what you do with the email after Fuzible read it. - Leave it as it is. - Move it to a "Fuzible" folder - Delete it. - Mark it "as read". Note: The POP protocol only supports deletion.
Limit to retrieve	By default, the program recovers all the mails from the inbox, which can take an extremely long time. It is best to set a limit here.

Active Directory



Search scope	Search depth for AD objects (base, one level, all levels)

Data Analyzer



Fuzible may need to analyze the data it collects to determinate the most accurate type. This feature is particularly useful in 2 cases:

- The target is a database and the Target table do not exist or has an inaccurate schema: Fuzible can either create them or "improve" them if requested (change the field type(s))
- Sync mode: To best compare Target and Source data that sometimes come from quite different sources, the analysis helps to better translate a value.
- Filtering and aggregating data from Sources other than a database

To date, Fuzible is capable of creating and/or transcoding the most standard fields: CHAR, NVARCHAR, VARCHAR, TEXT, DATE, DATETIME, TIMESTAMP, INTEGER, DECIMAL (X,X), FLOAT, BIGINT, SMALLINT, BIT. Any other type of field will be seen as a "TEXT" field.

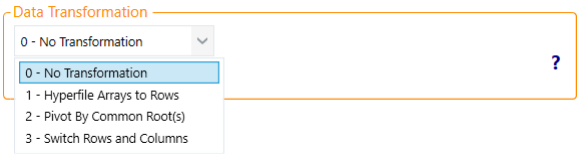
The lower the accuracy requested by the analyzer (scan all X lines), the more it will weight its analysis to ensure that the import is done correctly.

Example: A field identified VARCHAR(5) will be created in VARCHAR(10) in the target if the scan mode scans only all 50 lines)

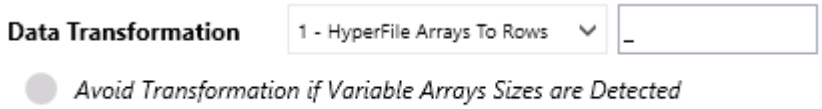
Note: For all queries retrieving less than 100,000 rows, the analyzer scans all rows, regardless of the chosen setting.

Data Transformation

The collected data can be completely transformed before its integration into the Target:



Hyperfile Arrays Transformation:



This is a kind of transformation specific to Hyperfile databases: some fields of this SGBD are in fact arrays, and when these arrays are retrieved by the ODBC driver, it produces as many columns as there are columns in these fields of "array" type (example: PPXRUBP_01, PPXRUBP_02, PPXRUBP_03...) Transformation automatically analyzes and counts all columns of this type (they always have the name of the field, followed by _XX) and flips them so that there is only one, and the data is thus transformed into rows and no longer columns.

Example: If you have na array-type column of 50 entried in HYPERFILE, the ODBC sends you 50 different fields! The transformation engine retains only one to favor a better display of this data (i.e. 50 rows for one originally)

Example: on the right, the initial data source, on the left, the result produced by the transformation:

```
id_sample;data_01;data_02;data_03;data_04;li_sample
1;hello1;hello2;hello3;hello4;firstrow
2;world1;world2;world3;world4;secondrow
3;wolf1;wolf2;wolf3;wolf4;thirdrow
```

id_sample	data	li_sample	IDX_COL
1	hello1	firstrow	1
1	hello2	firstrow	2
1	hello3	firstrow	3
1	hello4	firstrow	4
2	world1	secondrow	1
2	world2	secondrow	2
2	world3	secondrow	3
2	world4	secondrow	4
3	wolf1	thirdrow	1
3	wolf2	thirdrow	2
3	wolf3	thirdrow	3
3	wolf4	thirdrow	4

As can be seen, the fields data_01, data_02,... follow a "label + number" logic, which allows you to take "data" as a label, and to display the index in "IDX_COL"

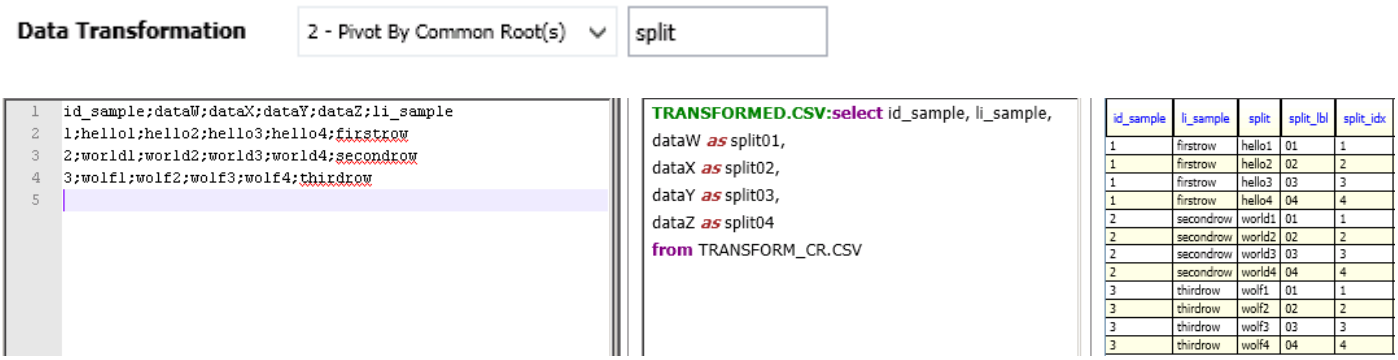
Pivot by Common Root:

This kind of transformation reverses source data according to a common root field name. For example, if you put "x," the program will flip all the fields that start with "x."
To this end, the source will be increased by 3 fields:

- An "x" field, containing the value.
- A "x_lbl" field, containing the rest of the fields name (ex: « x_001 » -> 001)
- A "x_idx" field containing the row index

If 7 fields with the "x" root have been detected, then 7 rows will be produced. These fields will of course be removed from the source and replaced by the 3 fields shown above.

Example: on the right, the initial source data, in the middle, the query, on the left, the result produced by the transformation:



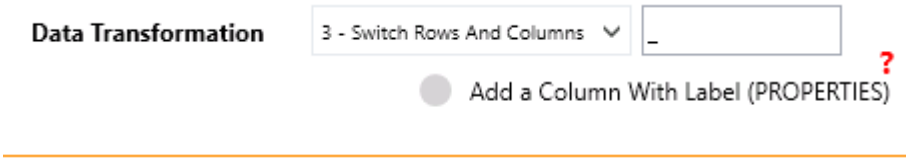
We see that the fields "W,X,Y,Z" were deliberately renamed with a common root in the query.

- "split": the root name

- "split_lbl": the data that was contained in the column

-"split_idx": the original column index

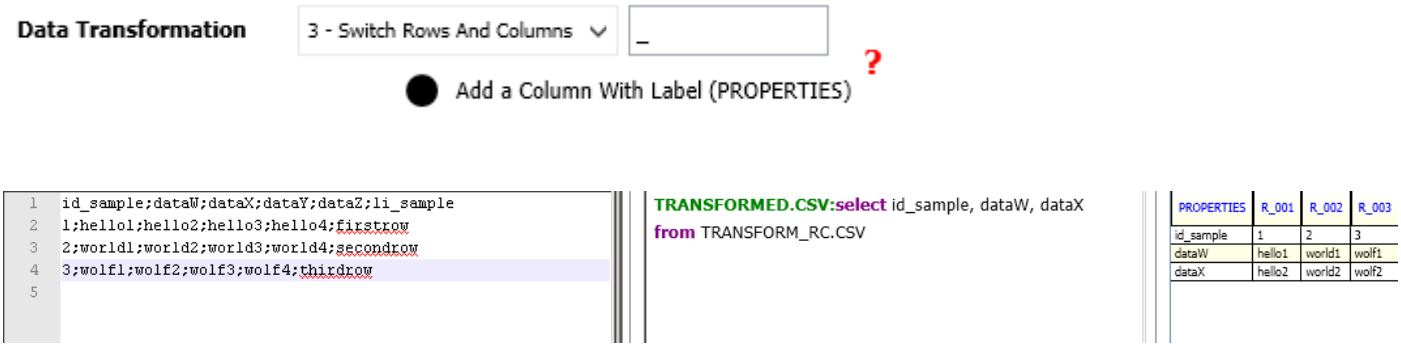
Switch Rows and Columns:



Simply flips the columns into rows and vice versa.
The "Add a column with label (PROPERTIES) option allows you to add a column with the original name of the original column associated with the reversed data.

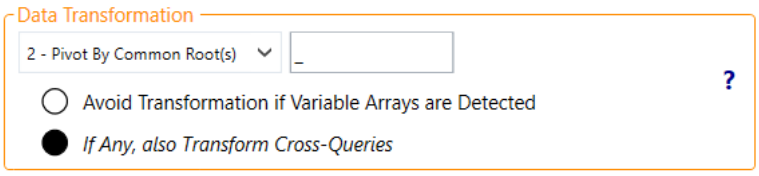
Example: on the right, the initial source, in the middle, the query, on the left, the result produced by the transformation:

... and with the "PROPERTIES" option



If Any, also transform Cross-Queries

If a Query is built with Cross-Queries (data coming from other Sources), the Transformation will only be executed on the Main Query, then all Data coming from Cross-Queries won't be transformed but merged with the already transformed Data from the Main Query. You can choose to Transform any Dataset that is collected through all Cross-Queries which means that Cross-Query behavior will be executed on full transformed datasets. Not only the main one.



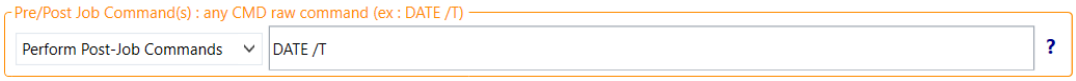
Pre/Post-Job Commands

Accessible from the "Source" tab and the "Target" tab, for "Database" or "File" connections. This feature is an answer for two identified needs:

- Sometimes a simple data copy Job is not enough, you need to launch something before or after this copy, and you want to avoid having to program these scenarios in an external Orchestration software.
- You want to retrieve some data before starting the Job to exploit it (conditioning the behavior of the job according to this data, for example)

This feature allows you to launch one or more commands before or after the Job is executed.

File



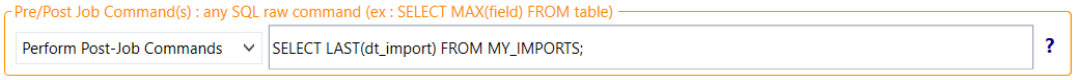
Any command you could launch from the Windows shell is supported. If multiple commands are to be launched, they must be separated by a ";"

Ex : c:\Tools\mycommand.bat

Ex2 : DATE /T (returns the actual date)

Fuzible adds any returned value/message/error from those commands into its LOG.

Database



Any SQL command compatible with the selected SGBD. The execution of a stored procedure, an UPDATE... If multiple commands are to be launched, they must be separated by a ";"

Ex : UPDATE myTable SET sent = 1 WHERE month = {%MM}

Ex2 : EXECUTE myProcedure('1')

In example 1, you see, as a reminder, that a dynamic parameter of the Job can be used.

These commands can return a value. Fuzible can exploit these values as dynamic parameters. For example, if I write in the dynamic settings of Job %CS1, it means that this dynamic setting will be replaced by the first value of the first command of the "Source" connection:

C - Command

S - Source

1 - Command No.1

We can also write %CT1 (Command Target No. 1) or %CT2...

Looped Pre-Job Commands

You may notice that if only one Pre-Job command has been set, an option appears. "Loop Job for each Result ».

This option allows you to make the Job scenario more complex by associating a dynamic parameter to the result of a command, which, if it returns several rows, allows you to loop the Job as many times as there are results, assigning a different dynamic parameter each time.

Explanations :

You call a stored procedure "getMails" (CALL is the MySQL syntax for calling a stored procedure)

... which returns this data set:

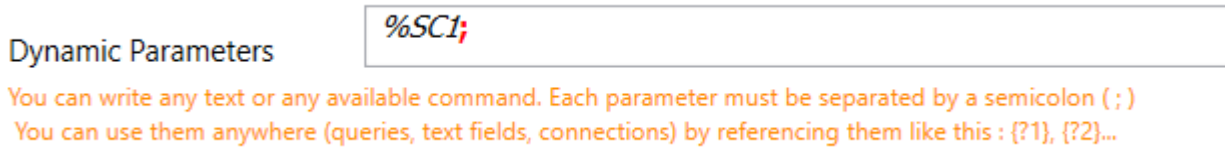
mailField	IdPeople	Comment
Leon@mymail.com	1	Our beautiful CEO
Arthur@mymail.com	2	Our incredible COO
Samantha@mymail.com	3	Our amazing CTO

You might want to send an email to each of these people, with an email containing, for example, their information summary.

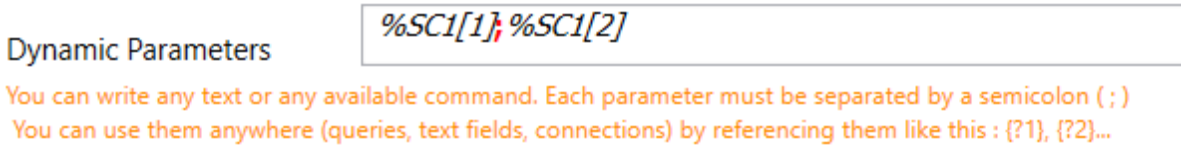
This option makes it possible to carry out this scenario, provided that you program the Job accordingly.

In this case, one or more dynamic parameters must be assigned to the results produced by the call to the stored procedure (or any other command returning a set of results).

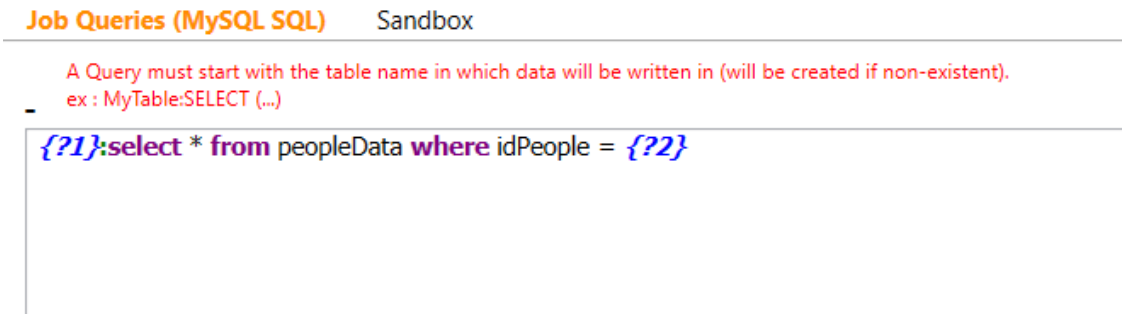
For example, a single variable can be associated in the following way ("Job Configuration" Tab):



Or more than one, by specifying the column number (in base 1) to which the parameter is associated:



You can then set a Query which, for each mail returned by the stored procedure, will send a mail to the person in question, with his personal information:



{ ?1} -> Will be replaced by the dynamic parameter N°1, which is filled with the data of the first column of the stored procedure, i.e. the person's email.

{ ?2} -> Will be replaced by the dynamic parameter N°2, which is filled with the data of the second column of the stored procedure, i.e. the ID of the person.

Thus, the Job will run in a loop until the result set of the stored procedure called in Pre-Job has been consumed.

In our example, the Job will run 3 times, with the following values

Itération	**Param. Dynamique 1**	**Param. Dynamique 2**
Première itération	{ ?1} => %CS1[1] => Leon@mymail.com	{ ?2} => %CS1[2] => 1
Deuxième itération	{ ?1} => %CS1[1] => Arthur@mymail.com	{ ?2} => %CS1[2] => 2
Troisième itération	{ ?1} => %CS1[1] => Samantha@mymail.com	{ ?2} => %CS1[2] => 3

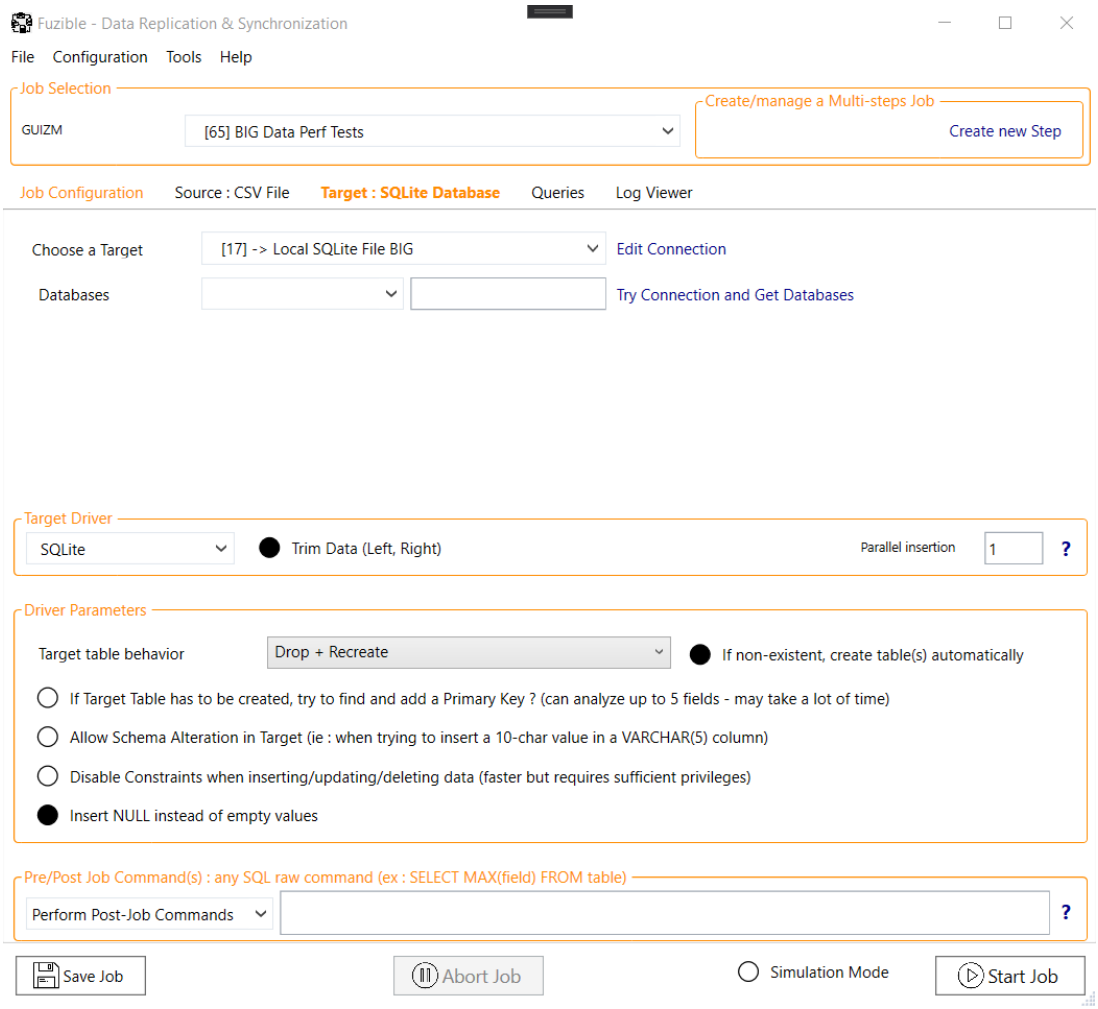
With this option, it is therefore possible to make your job a little more scenic and dynamic.

Restriction: Only one pre-job command can be entered for this option to be available

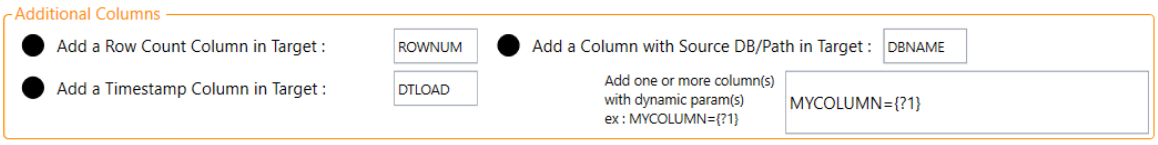
Furthermore, if it has been activated on the "Source" tab, it cannot be activated on the "Target" tab (and vice versa), even if a pre-Job command has been entered. This is to avoid making the Job too confusing and to avoid scenarios that require a more visual presentation of the Job's behavior (in the form of a diagram, for example)

Target tab

Here you choose the Target connection. In the case of a database, you can choose to view the list of available databases and use one that is different from the one in the connection string.



Common settings to all targets:

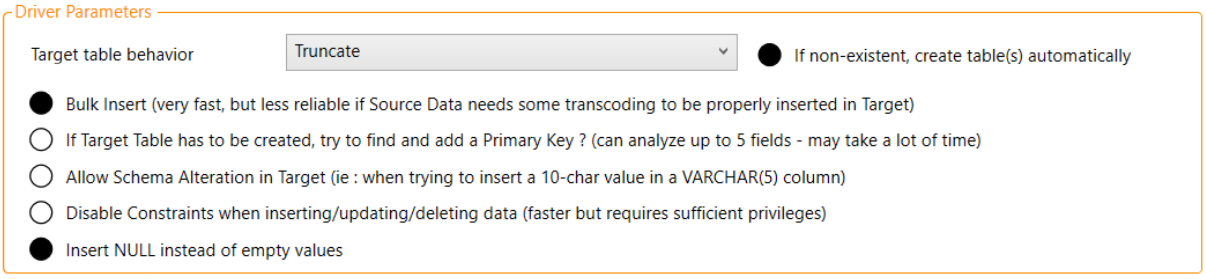


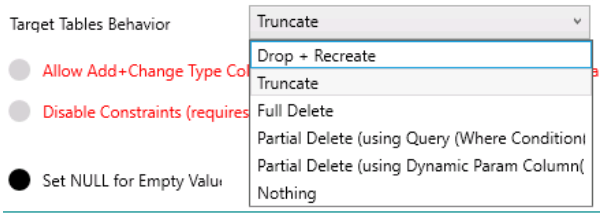
All of the following special columns use a default name that can be changed by the user.

Add a row count column in target	Adds a "ROWNUM" column to the data retrieved from the source, which is simply a row counter
Add a timestamp column in target	Adds a "DTLOAD" column to the data retrieved from the source that contains the data retrieval date
Add a column with source database/path	Adds a "DBNAME" column to the data retrieved from the source that contains the source of that data
Add a column with a dynamic param	Adds one or more optional columns to the data retrieved from the Source (ex:, MYCOLUMN={ ?1} : will add a "MYCOLUMN" column and fill in its data with the dynamic setting n°1 It is possible to add several columns by separating them like this: MYCOLUMN1={ ?1} ;MYCOLUMN2='test'

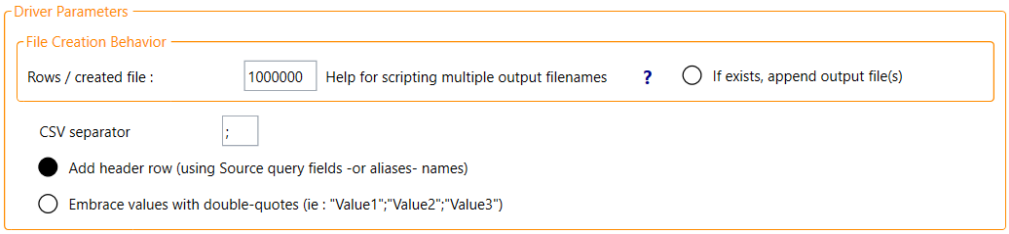
Trim data	Removes any whitespace before and after a string
Parallel Insertion	Option available only in "TARGET = Database" mode: It allows you to perform the INSERT, UPDATE, DELETE operations in multi-thread: Requires a high-performance Target database, especially if you combine parallel insertion with parallel queries execution (Source tab)!

Database



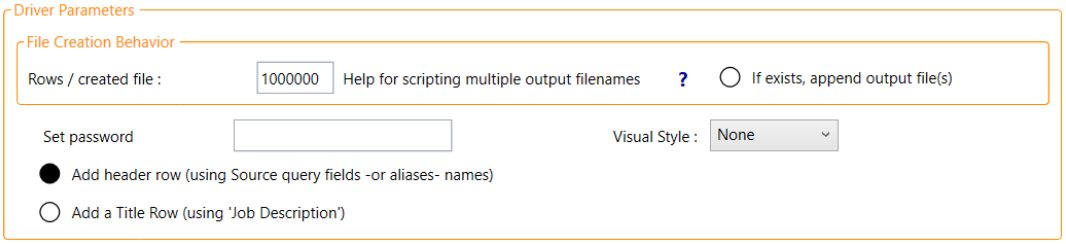
Target Tables Behavior	Available in "Data Replication" mode: Defines what to do on the target table when you fill it out.  1 - Drop - Recreate: delete the destination table and then rebuild it. 2 - Truncate: delete all data from the table using a "TRUNCATE" statement. 3 - Full Delete: delete all data from the table using a "DELETE" statement. 4 - Partial Delete using Query (Where): will use the "WHERE" filter(s) from the source query to remove data with the same filter in the destination table. Ex : MYTARGET :SELECT * FROM MYSOURCE WHERE id > 50 ➔ Fuzible will remove all "id > 50" in "MYTARGET" before inserting new data 5 - Partial Delete using Dynamic Param Column: Will use the dynamic column as a filter Ex : If you set an additional column MYCOLUMN={ ?1} with { ?1} using 'TEST' as a value, the DELETE FROM myTargetTable WHERE MYCOLUMN = 'TEST' statement will be performed before inserting new data. 6 – Nothing : Nothing will be done before inserting new data.
Bulk Insert	This is a very fast way of copying data into the Target Database. While it offers amazing performances, it is sometimes less reliable than traditional transactional SQL, especially when data needs to be converted on the fly between Source and Target.
If non-existent, create table(s) automatically	By default, if the Target table does not exists, Fuzible will automatically create it on-the-fly with the most accurate data schema. You can bypass this behavior. If the Target table does not exists, an error will be inserted in the LOG
Allow add+change type columns in target	Allows Fuzible, thanks to its data analysis engine, to modify the target table schema if necessary (change of column types). Requires significant privileges on the target database. This is especially useful if Source data is often changing and Target table needs to be adjusted accordingly
Disable constraints	Allows insertions to be performed by disabling foreign key constraints. Requires significant privileges on the target database.
Try to add primary key	If the target table does not have a primary key, Fuzible can create it on its own by analyzing all possible combinations of fields and values. The analysis is limited to a maximum of 5 fields. If the number of fields and data is huge, the scan can last an extremely long time and it is not advisable to use this feature. Also requires important privileges on the target database.
Set NULL for empty values	All "empty" data in the Source can be replaced as a "NULL" value in the Target data

CSV file



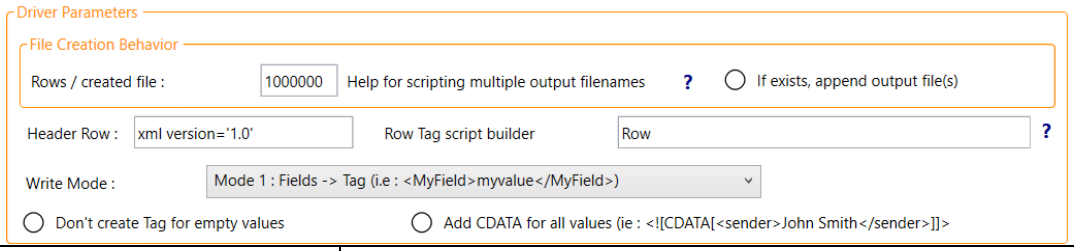
Rows/ File	How many rows you want to copy into a single file? If the number of rows in the Source data exceeds this value, a "pattern" must be set to name the files that are going to be created. ➔ See "Multiple files naming pattern"
Append Existing File	If the target file already has rows, you can decide not to overwrite it but add data into it.
CSV separator	Sets the separator character of the target file
Add header row	The header is built using Source field names
Embrace values with double quotes	Double quotes will be added before and after the value (ex : “test”;“125”;“hello”)

Excel file

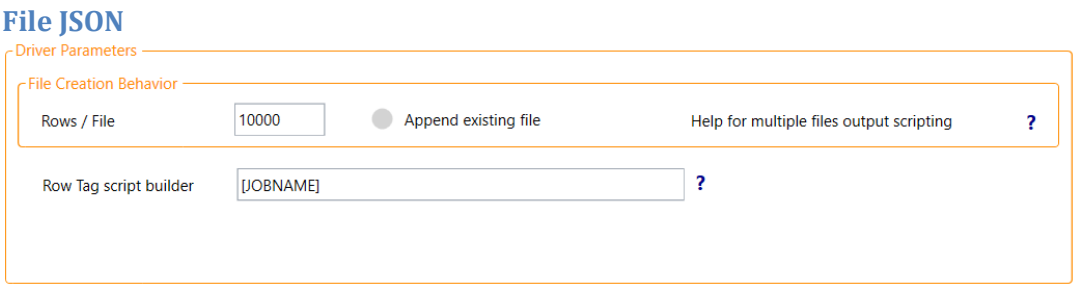


Rows/ File	How many rows you want to copy into a single file? If the number of rows in the Source data exceeds this value, a "pattern" must be set to name the files that are going to be created. ➔ See "Multiple files naming pattern"
Append Existing File	If the target file already has rows, you can decide not to overwrite it but add data into it.
Set password	Sets a password on the Excel file
Add header row	The header is built using Source field names
Add a title Row	Adds a general head row, the value will be the Job’s description
Style	Allows you to pick-up a graphical style.

File XML



Rows/ File	If the number of rows in the Source data exceeds this value, a "pattern" must be set to name the files that are going to be created. ➔ See "Multiple files naming pattern"
Append Existing File	If the target file already has rows, you can decide not to overwrite it but add data into it.
Header	Choose the XML header (usually: xml version=’1.0’)
Write Mode	Choose the way you want to build the XML schema : ➔ Mode 1 : Each row is written inside a main row tag (Row Tag Script Builder), each field is a sub-tag, and contains the associated value Ex : <Row><MyField>myValue</MyField><MyField2>myValue2</MyField2></Row> ➔ Mode 2 : Each row is a row from the Source, each value is an attribute (field name). Ex : <Row MyField= "myValue" MyField2= "myValue2"/>
Add CDATA tag for each value	(Mode 1 only) If the source contains exotic values, the standard tag "CDATA" allows the data to be framed so that an XML interpretation engine understands that the values framed by this tag contain special characters
Don't create tag for empty value	If a value is empty, lets tell if you still want to generate an empty tag in the output file
Row Tag script builder	Sets a behavior script for the tag of each row. ➔ See "Tag Builder"



Rows/ File	If the number of rows in the Source data exceeds this value, a "pattern" must be set to name the files that are going to be created. ➔ See "Multiple files naming pattern"
Append Existing File	If the target file already has rows, you can decide not to overwrite it but add data into it.
Row Tag script builder	Sets a behavior script for the tag of each row. ➔ See "Tag Builder"

TAG BUILDER

By default, the structure of an XML and JSON file produced by Fuzible:

The JSON file will be as follows:


```
[
    {
    "id_sample": 5,
    "li_sample": "five",
    "dt_random_date": "14/09/2020 13:10:49",
    "nb_random_number": 25,
    "li_random_string": "iidbknotshqpzijf",
    "id_group": "005",
    "id_ssgroup": "000"
    },
    {
    "id_sample": 28,
    "li_sample": "two-eight",
    "dt_random_date": "18/10/2020 13:10:49",
    "nb_random_number": 13,
    "li_random_string": "ysisszfpzpcmohdx",
    "id_group": "008",
    "id_ssgroup": "003"
    },
    {
    "id_sample": 66,
    "li_sample": "six-six",
    "dt_random_date": "15/01/2021 13:10:49",
    "nb_random_number": 56,
    "li_random_string": "inpljrzhwvkmvmfb",
    "id_group": "006",
    "id_ssgroup": "001"
    }
]
```

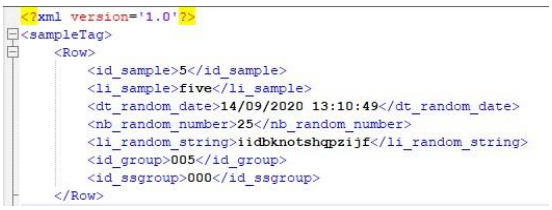

... and the XML file:


```
<?xml version='1.0'?>
<sample_table_1>
    <Row>
        <id_sample>5</id_sample>
        <li_sample>five</li_sample>
        <dt_random_date>14/09/2020 13:10:49</dt_random_date>
        <nb_random_number>25</nb_random_number>
        <li_random_string>iidbknotshqpzijf</li_random_string>
        <id_group>005</id_group>
        <id_ssgroup>000</id_ssgroup>
    </Row>
    <Row>
        <id_sample>28</id_sample>
        <li_sample>two-eight</li_sample>
        <dt_random_date>18/10/2020 13:10:49</dt_random_date>
        <nb_random_number>13</nb_random_number>
        <li_random_string>ysisszfpzpcmohdx</li_random_string>
        <id_group>008</id_group>
        <id_ssgroup>003</id_ssgroup>
    </Row>
    <Row>
        <id_sample>66</id_sample>
        <li_sample>six-six</li_sample>
        <dt_random_date>15/01/2021 13:10:49</dt_random_date>
        <nb_random_number>56</nb_random_number>
        <li_random_string>inpljrzhwvkmvmfb</li_random_string>
        <id_group>006</id_group>
        <id_ssgroup>001</id_ssgroup>
    </Row>
</sample_table_1>
```


The main tag uses the name of the input table (the first if the query contains joins)

Now, if I want to change the main XML tag, I just need to ask an alias in my query:

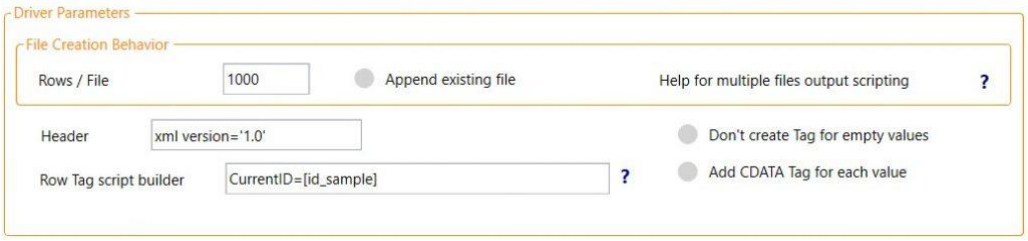
"SampleTag" is the alias that will be used in the XML body



Supported keywords:

- [JOBNAME]
- [DATETIME]
- [ROWCOUNT]
- [FILECOUNT]
- [USER]
- [anyField]

I will set the script in the "Target" menu:



The

The JSON file creates:

... and the XML file:


```xml
<?xml version='1.0'?>
<sampleTag>
    <CurrentID=5>
        <id_sample>5</id_sample>
        <li_sample>five</li_sample>
        <dt_random_date>14/09/2020 13:10:49</dt_random_date>
        <nb_random_number>25</nb_random_number>
        <li_random_string>iidbknotshqpzijf</li_random_string>
        <id_group>005</id_group>
        <id_ssgroup>000</id_ssgroup>
    </CurrentID=5>
    <CurrentID=28>
        <id_sample>28</id_sample>
        <li_sample>two-eight</li_sample>
        <dt_random_date>18/10/2020 13:10:49</dt_random_date>
        <nb_random_number>13</nb_random_number>
        <li_random_string>ysisszfpzpomohdx</li_random_string>
        <id_group>008</id_group>
        <id_ssgroup>003</id_ssgroup>
    </CurrentID=28>
    <CurrentID=66>
        <id_sample>66</id_sample>
        <li_sample>six-six</li_sample>
        <dt_random_date>15/01/2021 13:10:49</dt_random_date>
        <nb_random_number>56</nb_random_number>
        <li_random_string>inpljrzhwvkmvmfb</li_random_string>
        <id_group>006</id_group>
        <id_ssgroup>001</id_ssgroup>
    </CurrentID=66>
</sampleTag>
```


MULTIPLE FILES NAMING PATTERN

When using a File Connection as the Target, you can decide to split the result into several files from a number of rows. For example, if the Source data contains 1000 rows, you can split the result into a single file containing 1000 rows, or 5 files of 200 rows each, or 1000 files of 1 row each.

This multi-file pattern can be smartly configured from the “Queries” tab (see below)

[QUERYALIAS]	Alias from the first source table of the query **Ex** : MYFILE_[QUERYALIAS].CSV : SELECT * FROM MYTABLE AS MYQUERY **Gives** : MYFILE_MYQUERY.CSV
[FILECOUNT]	File counter: returns the number of files created. If the program has already created 3 files, it will return "4" to the next [FILECOUNT] pattern. **Ex :** MYFILE_[FILECOUNT].CSV : SELECT * FROM MYTABLE **Gives** : MYFILE_1.CSV, MYFILE_2.CSV...
[ROWCOUNT]	Row counter: returns the source query row number to the start of the file. If we define a file change every 1000 rows, at the creation of the 2nd file, the software will return 1001 (to the 3rd file, 2001) **Ex :** MYFILE_[ROWCOUNT].CSV : SELECT * FROM MYTABLE **Gives** : MYFILE_1.CSV, MYFILE_1001.CSV...
[COLUMN]	Returns the value from a field when the new file is created If the value of the "my_field" field is "Hello" at the time of the creation of the new file, then "World" when the next one is created, the engine will return "Hello" and so on. **Ex :** MYFILE_[myField].CSV : SELECT myField FROM MYTABLE **Gives** : MYFILE_Hello.CSV, MYFILE_World.CSV...

What you can do:

- Use those keywords in any order.
- Use them multiple times.
- [COLONNE] can be used several times, with several different columns (make sure the column exists in the source, if any, the name of the column is returned and not its value!)
- Intersperse characters between each keyword (ex: [FILECOUNT]-_-[ROWCOUNT])

Restriction:

- The special characters will be automatically replaced with a "_"

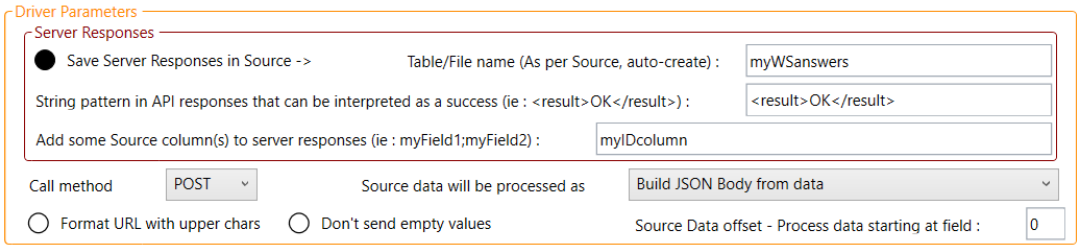
Some examples:

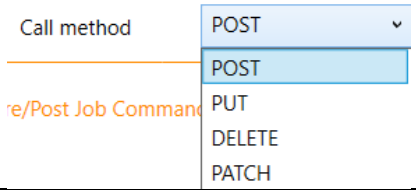
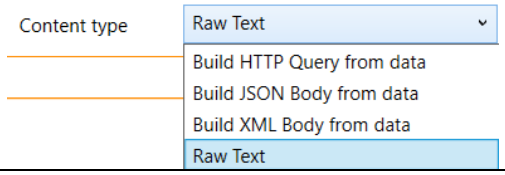
Postulate: Source produced 2000 rows and we want to have 1000 rows/file. Filename = TEST, The output file is a CSV.

2 files will be created:

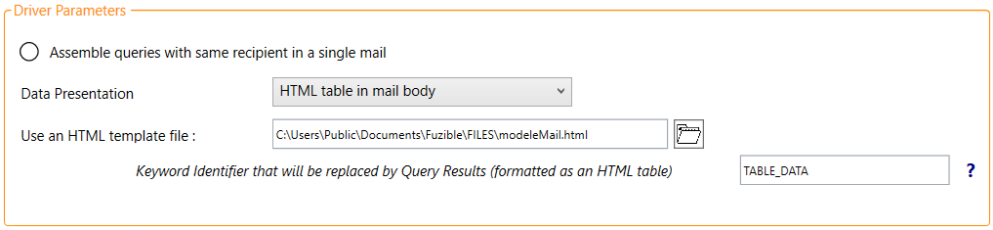
hello[ROWCOUNT]world[NOM_CLIENT].CSV *: SELECT ...*	- hello0worldFNAC.CSV - hello1001worldAUCHAN.CSV
Hello[FILECOUNT]_[ID_CLIENT] *.CSV : SELECT ...*	- Hello1_283.CSV - Hello2_81036.CSV
[QUERYTARGETNAME][ROWCOUNT][FILECOUNT] .CSV : *SELECT ...*	- TEST11.CSV - TEST10012.CSV
[QUERYTARGETNAME]%§£[FILECOUNT] .CSV *: SELECT ...*	- TEST___1.CSV - TEST___2.CSV

Webservice REST/NUXEO



Save HTTP responses in source	Webservices usually generate answers (XML, JSON) that Fuzible can retrieve and integrate into the connection that served as Source (if it's a BDD, in a table, if it's another connection, in files)
Table Log/File Name	(Optional) table name (or file) that will receive answers from the queries made to the API
Track source column(s) in responses	(Optional) If your Source query contains 100 rows, it means there will be 100 calls to the API, and it is not easy to find your way around the list of answers it will deliver. You can define one or more fields from the source to be kept and stored in the answers table/file to track down the calls.
String that says success in WS Answers	If you know the answer format of n API, you can set a "piece of content" of these answers that identifies the call as having been a success. If this piece of string is not in the answer, Fuzible will produce a "WARNING" in the LOG
Format URL with upper chars	Compatibility mode: Some API for which data is sent as HTTP parameters only accept capital-formatted URLs
Don't Send Empty Values	When building the HTTP query, Fuzible will avoid adding fields with empty data
Call Method	Call method supported by the API (POST, PUT, DELETE, PATCH) 
Content Type	Determines how the content of the source data will be sent to the webservice. Either in the form of JSON or XML data in the body, as HTTP parameters or in "raw" mode (when your source file is a raw JSON file for example) 
Columns send offset	A query to an HTTP webservice builds a concatened chain of fields and their values. However, if the source query returns for example 10 fields, you can decide to send only the last 8 in the API if you set an offset of 2

Mailbox



Assemble queries with same recipient in one mail	If the Job has multiple queries with the same email address as the Target, you can decide to group all the results into one email rather than send 1 mail / query.
Data Presentation	Here, we choose how the Source data will be presented in the email: - HTML table in mail body: an HTML table in the body of the mail - Excel file with (or without) a header: an attachment in Excel format - CSV file with (or without) a header: an attachment in CSV format Note: If the table contains too many rows (to be set in the software configuration), and one has chosen "HTML table", it is a CSV attachment that will be attached to the mail rather than a table in the body Note 2: The name of the table in HTML table mode will match the alias of the first table of the query. Ex : SELECT * FROM MYTABLE -> MyTable will be the table header Ex2 : SELECT * FROM MYTABLE as My_Reporting -> My Reporting will be the table header (any underscore will be replaced by a whitespace as well)
Use an HTML Template file	Is showned only if you choosed « HTML table in mail body » Fuzible creates HTML content using data retrieved from Source Queries. The default Template is quite simple, that's why you can choose a customized one. In that case, Fuzible needs to know where to include the Source Query data into that Template. This is where the Keyword option stands for : 1/ If no keyword is specified, Fuzible will behave like this : It will take the Query Alias (ie : mymail@mail.com:select * from myCustomers as MyAlias) and try to find it in the HTML Template. - If found (in the example, MyAlias), it will be replaced by the HTML code that has been produced from Query results. - If not found, the results will be concatenated to the Template HTML code. 2/ If a keyword has been specified : - Fuzible will replace that keyword by the HTML code that has been produced from Query results. - If the keyword has not been found into the Template, the results will be concatenated to the Template HTML code. 3/ Special case when using "Assemble Queries with same Recipient in a single mail" If you have multiple queries that will be merged in a single mail, the Template can be populated smartly. - Your Template uses a keyword (ie : MYTABLE01) that is intended to handle results from the first Query - Your Templace uses another keyword (ie : MYTABLE02) that is intended to handle results from the second Query -> You can set your Queries like this : - MyMail@mail.com:SELECT * FROM MyCustomers as MYTABLE01 WHERE last_transaction = CURRENT_TIMESTAMP - MyMail@mail.com:SELECT * FROM MySuppliers as MYTABLE02 WHERE last_transaction = CURRENT_TIMESTEMP Results from 'MyCustomers' Query will be injected into the Template by replacing 'MYTABLE01' keyword. Results from 'MySuppliers' Query will be injected into the Template by replacing 'MYTABLE02' keyword. **Practical note: if you have injected dynamic parameters recognized by Fuzible in your HTML Template (e.g.: {?1}), these will be replaced by the dynamic values of the Job!**
Keyword Identifier	The optional Keyword included in the Template that will be replaced by the HTML table

Note: The mail subject will be the description of the Job

Active Directory

Key Property	Sets the unique property that identifies an object in the AD (for example, "name" is the default single property for a user account)
Existing Objects	Chooses how to behave when writing an AD object: - Remove: It is removed for re-creation - Ignore: Leave the object as it is, without overwriting it Note: In "Data Synchronization" mode, this option is disabled because data is compared, so inputs will be updated, inserted or deleted depending on the mode of sync chosen in the main settings of the Job
Activate New Entries	When a new entry is created, it will not be activated by default. You can force its activation by checking this box

Queries tab

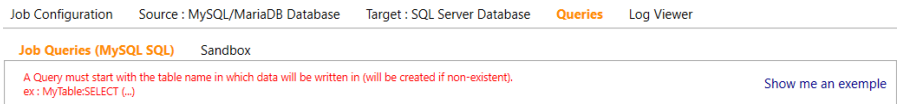
The heart of Fuzible is here. As the software is an IN/OUT reader/writer, a Data Source is no more than a group of fields and with values.

The software aims to greatly simplify the tedious work of mapping and transforming data. It uses the principle of SQL queries to work.

In case the source is an SGBD, no problem, it is the SQL language of the SGBD that will work, you can enter any query (simple or complex) compatible with it to extract data.

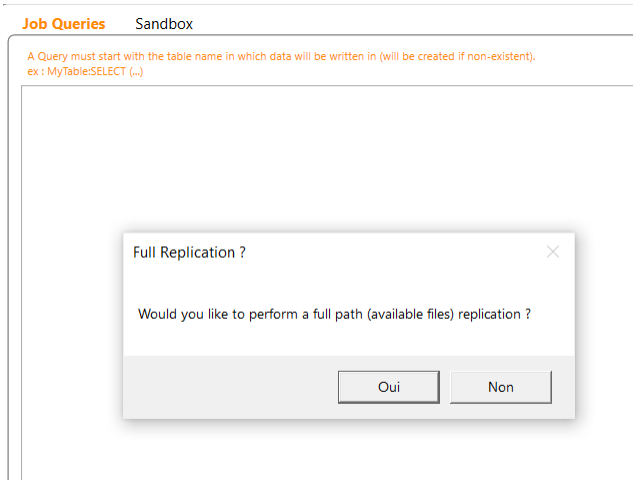
In all other cases, Fuzible relies on the standard SQL language (SQL92 compatible): The queries you write are translated and applied to the type of Source you are querying.

The sub-menu reminds you of the SQL language you are using when building your queries.

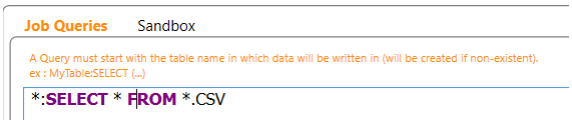


Each query for a Job is written as such: **OUTPUT:SELECT [...]**

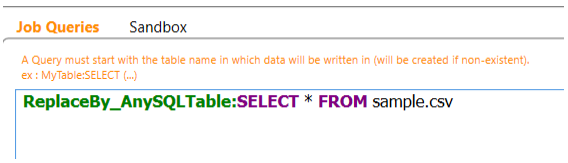
If you click right over the query when it is empty, you'll be proposed a demo Query if you're not familiar with how it works. If the Source is a database or a file, you will be asked to perform a full Replication of everything that is available:



This will produce the following result (in the example, my Source connection is a local path containing CSV files):



If you don't want a full replication, a demo Query will be added:



Now let us see how to write a query manually, since all the interest is there!

Output	
BDD	Destination table name A Query must start with the table name in which data will be written in (will be created if non-existent). ex : MyTable:SELECT (...) AX_f_fae_avoirs:select * from (SELECT T1.RECID AS id_ecriture, CAST(T1.ACCOUNTINGCURRENCYAMOUNT AS numeric(12,3)) AS nb_montant, CAST(LEFT(CONVERT(varchar, T2.ACCOUNTINGDATE,112),6) AS INTEGER) AS id_anneemois,
FICHIER	Destination filename A Query must start with the filename in which data will be written in. ex : MyFile.CSV:SELECT (...) CLI_[CLI_code]_{%YYYY%MM%DD%HH%mm%SS}.XML:select * from cli_viewcli_export_comete as clients SIT_[SIT_code]_{%YYYY%MM%DD%HH%mm%SS}.XML:select * from cli_viewsit_export_comete as sites *Note: In the example, the name of the output file is dynamic!*
MAIL	The recipient's email address. It is possible to define several by putting a ";" between each! A Query must start with the mail address(es) for which you want to send data to (addresses have to be separated by a ','). ex : mymail@gmail.com,anymail@yahoo.fr:SELECT (...) mymail@hotmail.com:select * from (select snp.id_matricule as Matricule, concat(snp.li_nom, ' ', snp.li_prenom) as Nom_Prenom, ptc.li_typecontrat, pmfc.li_motiffincontrat as Motif_Fin_Contrat, pqc.li_qualifcontrat as Qualification_Contrat, pa.li_agence as Agence,
WEBSERVICE	Name of the API object that will receive the data (basically, the tail of the URL) A Query must start with the target Webservice Object (URL queue) in which the source data will be sent in. ex : /ws/mywebservice:SELECT (...) sobjects/Account/:select * from Account.csv
ACTIVE DIRECTORY	Name of the AD object on which we will write data. The Source field names must match an existing attribute from the AD. You can alias the fields if they don't have a valid attribute name. (ex : SELECT myname as name, account as sAMAccountName FROM...) users:select addata.sAMAccountName, addata.name, addata.description from ad_users.csv as addata

How field mapping works

By simplification, a SELECT statement is written as follows:

SELECT * FROM [SOURCE]

Or

SELECT field1, field2 [...] FROM [SOURCE]

Or

SELECT field1 **as** dest1, field2 **as** dest2 [...] FROM [SOURCE]

Or

SELECT CONCAT(field1, field2) **as** dest1, TRIM(field2) **as** dest2 [...] FROM [SOURCE]

The field alias serves as a reference to Fuzible to build and fill the Target connexion. If it is not present, the name of the field is used, otherwise, it is the alias:

On a « SELECT * », the Target's column names will be the same as the Source.

On a « SELECT field1 », the column name in the target will be "field1."

On a « SELECT field1 **as** dest1 », « dest1 » will be used as a field name in the Target.

... and so on.

Hence the usefulness of preparing your Source query well.

Important note: If you want to put an alias on the fields you retrieve, you must use "AS"
Indeed, if a CSV file includes field names with whitespaces, Fuzible will be unable to separate the alias and the fields. Field framing is not supported (quotes or hooks around the fields)
Example:

SELECT mon champ, mon deuxième champ **FROM** monfichier.CSV	OK
SELECT mon champ 1erchamp, mon deuxième champ 2emechamp **FROM** monfichier.CSV	NON-OK
SELECT mon champ **as** 1erchamp, mon deuxième champ **as** 2emechamp **FROM** monfichier.CSV	OK

In case the Target is a SGBD, Fuzible compares the Source and Target fields, performs its "INSERT, DELETE, UPDATE" operations based on what is available in the Target only. For example, if you're querying a Source that has 50 fields and the Target has only 25 of these fields, that's no problem. The reverse is also true.

Regarding inter-compatibility (the Source would be SQL Server, the Target would be MySQL), Fuzible transcodes the data on the fly to make it compatible between both Source and Target, you don't have to worry about the data types.

Please also note that Fuzible SQL understands field framing if column/table names do not only use numbers/letters :

ex : SELECT "my,weird field/with !strangechars" FROM myFile

Special cases of synchronization queries:

Sync. works by comparing Source and Target data. This way of working requires the same query to be performed in both environments. The software knows how to transcode most queries but there are some limitations:

Example 1:

CIBLE :SELECT champ1 **as** dest1, champ2 **as** dest2 FROM SOURCE WHERE champ1 = 'TEST'

➔ The query that will be executed on the Target will be :

SELECT dest1, dest2 FROM **CIBLE** WHERE dest1 = 'TEST'

The « WHERE » filter has been transcoded.

Another case :

CIBLE :SELECT champ1 **as** dest1, champ2 **as** dest2 FROM SOURCE WHERE champ3 = 'TEST'

➔ The query that will be executed on the Target will be :

SELECT dest1, dest2 FROM **CIBLE** WHERE champ3 = 'TEST'

The problem is that "champ3" does not exist in the SELECT statement. Fuzible cannot know what "field3" refers to in the Target table. If by chance this field exists (ISO-perimeter tables), it will not be a problem, but if this field does not exists (because the source query is complex, the conditions refer to fields on join tables, sometimes very complex conditions (nested SELECT ...)) the query will not succeed, and the sync. will surely fail.

An SQL trick to make up for this particularity: have the source SGBD execute a nested query:

Let us take this complex query in Dynamics AX. It contains several transformations and conditions. By framing it as a sub-query, Fuzible will then only worry about the main query to make its transcoded query.

```
AX_f_ecritures:SELECT * FROM
(SELECT T1.RECID AS id_ecriture,
T1.ACCOUNTINGCURRENCYAMOUNT AS nb_montant,
T2.ACCOUNTINGDATE AS dt_ecriture,
CAST(LEFT(CONVERT(varchar, T2.ACCOUNTINGDATE,112),6) AS INTEGER) AS id_anneemois,
T6.DATAAREA AS id_societe_ax,
MA.MAINACCOUNTID AS id_comptecomptable,
T3.DISPLAYVALUE as li_analytique,
CASE WHEN SUBSTRING(T3.DISPLAYVALUE, 8, 3) = '--' OR CHARINDEX('-', T3.DISPLAYVALUE) = 0 THEN NULL ELSE
SUBSTRING(T3.DISPLAYVALUE, 8 + LEN(MA.MAINACCOUNTID) - 6, 3) END as id_bu_ax,
SUBSTRING(T3.DISPLAYVALUE, 12 + LEN(MA.MAINACCOUNTID) - 6, 3) as id_activite_ax,
SUBSTRING(T3.DISPLAYVALUE, 16 + LEN(MA.MAINACCOUNTID) - 6, 3) as id_agence_ax,
CASE WHEN LEN(LTRIM(SUBSTRING(T3.DISPLAYVALUE, 20 + LEN(MA.MAINACCOUNTID) - 6, 9))) IN (3,5) THEN '' ELSE
REPLACE(SUBSTRING(T3.DISPLAYVALUE, 20 + LEN(MA.MAINACCOUNTID) - 6, 9), '-', '') END as id_chantier_ax,
CASE WHEN LEN(LTRIM(SUBSTRING(T3.DISPLAYVALUE, 20 + LEN(MA.MAINACCOUNTID) - 6, 9))) = 3 THEN SUBSTRING(T3.DISPLAYVALUE, 20 +
LEN(MA.MAINACCOUNTID) - 6, 3) ELSE '' END as id_metier_ax,
CASE WHEN LEN(LTRIM(SUBSTRING(T3.DISPLAYVALUE, 20 + LEN(MA.MAINACCOUNTID) - 6, 9))) = 5 THEN SUBSTRING(T3.DISPLAYVALUE, 20 +
LEN(MA.MAINACCOUNTID) - 6, 5) ELSE SUBSTRING(T3.DISPLAYVALUE, CASE WHEN LEN(LTRIM(SUBSTRING(T3.DISPLAYVALUE, 30 +
LEN(MA.MAINACCOUNTID) - 6, 5))) = 4 THEN 29 + LEN(MA.MAINACCOUNTID) - 6 ELSE 30 + LEN(MA.MAINACCOUNTID) - 6 END, 5) END as
id_destination_ax,
t1.text AS li_ecriture,
t2.SUBLEDGERVOUCHER AS li_numero_piece,
ljt.JOURNALNAME as id_code_journal,
T2.JOURNALNUMBER as id_journal,
T1.QUANTITY AS nb_quantite,
T2.CREATEDDATETIME as dt_saisie,
CONVERT(nvarchar(6), T2.CREATEDDATETIME, 112) as id_anneemois_saisie,
T2.CREATEDBY as li_utilisateur
FROM GENERALJOURNALACCOUNTENTRY T1
LEFT JOIN GENERALJOURNALENTRY T2 ON (T1.GENERALJOURNALENTRY=T2.RECID AND (T1.PARTITION = T2.PARTITION))
LEFT JOIN LEDGERENTRYJOURNAL lej on t2.LedgerEntryJournal=lej.Recid
LEFT JOIN LEDGERJOURNALTABLE ljt on  lej.JournalNumber=ljt.JournalNum and ljt.DATAAREAID=T2.SUBLEDGERVOUCHERDATAAREAID
LEFT JOIN DIMENSIONATTRIBUTEVALUECOMBINATION T3 ON (T1.LEDGERDIMENSION=T3.RECID AND (T1.PARTITION = T3.PARTITION))
LEFT JOIN LEDGER T4  ON (T2.LEDGER=T4.RECID AND (T2.PARTITION = T4.PARTITION))
LEFT JOIN FISCALCALENDARPERIOD T5 ON (T2.FISCALCALENDARPERIOD=T5.RECID AND (T2.PARTITION = T5.PARTITION))
LEFT JOIN DIRPARTYTABLE T6 ON (((((T6.PARTITION=T1.PARTITION) AND (T6.PARTITION=T1.PARTITION)) AND (T6.PARTITION=T1.PARTITION))
AND (T4.PRIMARYFORLEGALENTITY=T6.RECID AND (T4.PARTITION = T6.PARTITION))) AND (T6.INSTANCERELATIONTYPE IN (41) ))
Left join MAINACCOUNT MA  on MA.RECID = T3.MAINACCOUNT
LEFT JOIN DIMENSIONHIERARCHY H ON T3.ACCOUNTSTRUCTURE = H.RECID AND H.PARTITION=T3.PARTITION
WHERE 1 = (CASE WHEN t2.SUBLEDGERVOUCHER LIKE 'CLOTURE%' AND MONTH(T2.ACCOUNTINGDATE) = 12 AND MA.MAINACCOUNTID <
600000 THEN 0 ELSE 1 END)
AND 1 = (CASE WHEN (T6.DATAAREA IN ('AIR', 'MPY') AND T2.ACCOUNTINGDATE >= '01/01/2019') THEN 0 ELSE 1 END)
) as REQ
WHERE REQ.id_anneemois_saisie >= 202001
```

Will be transcoded for the target as: SELECT * FROM **AX_f_ecritures** WHERE id_anneemois_saisie >= 202001

Now let's see what can be done with an SQL query:

SELECT - From a database, to a database

A Query must start with the table name in which data will be written in (will be created if non-existent).
ex : MyTable:SELECT (...)

Quick Help

```
AX_dim_agences:select distinct value as id_agence_ax, name as li_agence from DimAttributeOMCostCenter
AX_dim_societes:select ID as id_societe_ax, NAME as li_societe from BICOMPANYVIEW
AX_dim_activites:select value as id_activite_ax, name as li_activite from DimAttributeOMBusinessUnit
AX_dim_metiers:select value as id_metier_ax, name as li_metier from DimAttributeOMDepartment
AX_dim_plancomptable:SELECT DISTINCT(A.[MAINACCOUNTID]) as id_comptecomptable,
LEFT(A.[MAINACCOUNTID], 1) as id_racine_comptecomptable,
A.[NAME] as li_comptecomptable,
A.[TYPE] as id_type,
CASE A.[TYPE] WHEN 0 THEN 'Bilan' WHEN 3 THEN 'Résultat' ELSE 'Inconnu' END as li_type
FROM [MAINACCOUNT] A
INNER JOIN (SELECT MIN(RECID) AS RECID, MAINACCOUNTID FROM MAINACCOUNT GROUP BY MAINACCOUNTID) B ON A.R
ORDER BY A.[MAINACCOUNTID]
```

Any "SELECT" statement that is compatible with the source SGBD. This can be a simple or a more complex one.

SELECT - From a database, to a file

A Query must start with the filename in which data will be written in.
ex : MyFile.CSV:SELECT (...)

Show me an exemple

```
MyTutorial.csv:SELECT user as userColumn, connstring_id as idColumn, connstring_name as nameColumn
from user_connstrings WHERE user LIKE 'GUIZ'
MyTutorial2.csv:SELECT COUNT(*) as nbConn
from user_parameters as up
inner join user_connstrings uc on up.user = uc.user
order by up.user ASC
```

Any "SELECT" query that is compatible with the source SGBD. This can be simple or complex.

SELECT - From a database, to an email address

A Query must start with the mail address(es) for which you want to send data to (addresses have to be separated by a ',').
ex : mymail@gmail.com,anymail@yahoo.fr:SELECT (...)

Quick Help

```
{?1}:select * from v_adm_metriques as Métriques_Principales;
{?1}:SELECT nspname || '.' || relname AS "Nom_table",pg_size_pretty(pg_relation_size(Top_10_des_tables_les_plus_lourdes.oid))
WHERE nspname NOT IN ('pg_catalog', 'information_schema') and relkind = 'r'
ORDER BY pg_relation_size(Top_10_des_tables_les_plus_lourdes.oid) DESC LIMIT 10
{?1}:SELECT Top_10_des_tables_avec_le_plus_de_lignes.relname as Nom_table, reltuples as Nombre_de_lignes
FROM pg_class as Top_10_des_tables_avec_le_plus_de_lignes
JOIN pg_stat_user_tables AS tabstat ON Top_10_des_tables_avec_le_plus_de_lignes.relname = tabstat.relname
ORDER BY reltuples DESC LIMIT 10
```

Any "SELECT" query that is compatible with the source SGBD. This can be simple or complex.
Note: The source query makes use of Dynamic Parameters, in this example, a dynamic @mail address is set in the "Job Configuration" tab, and used as the Output

Apart from a query on a SGBD, Fuzible's SQL engine takes over:

SELECT - From a file

A Query must start with the filename in which data will be written in.
ex : MyFile.CSV:SELECT (...)

```
SAMPLE_OUTPUT_LEFTJOIN.XLSX:SELECT fileA.id_sample AS IdInitial, fileB.id_sample_join AS IdJoin,
fileB.li_sample_join as LiJoin, ISNULL(fileB.li_sample_join, 'Nothing') as IsNullField
FROM SAMPLE.CSV AS fileA
LEFT JOIN SAMPLE_JOIN.CSV AS fileB ON fileA.id_sample = fileB.id_sample_join
SAMPLE_OUTPUT_INNERJOIN.XLSX:SELECT fileA.id_sample AS IdInitial, fileB.id_sample_join AS IdJoin,
fileB.li_sample_join as LiJoin FROM SAMPLE.CSV AS fileA
INNER JOIN SAMPLE_JOIN.CSV AS fileB ON fileA.id_sample = fileB.id_sample_join
SAMPLE_INNERJOIN_SUBQUERY.XLSX:SELECT fileA.id_sample AS IdInitial, fileB.id_sample_join AS IdJoin,
fileB.li_sample_join as LiJoin FROM SAMPLE.CSV AS fileA
INNER JOIN (SELECT * FROM SAMPLE_JOIN.CSV) AS fileB ON fileA.id_sample = fileB.id_sample_join
```

See here a somewhat complex example of queries made on multiple files, with joins, and even a sub-query. The SQL syntax is strictly the same as that of a traditional SGBD.

SELECT - From a webservice using Fuzible SQL (A)

A Query must start with the table name in which data will be written in (will be created if non-existent).
ex : MyTable:SELECT (...)

Quick Help

```
glpi_tickets:select
Entity.id as id_entite,
Entity.name as li_entite,
Ticket.id as id_ticket,
Ticket.name as li_ticket,
Ticket.date as dt_ticket,
Ticket.closedate as dt_cloture,
Ticket.urgency as nb_urgency,
Ticket.impact as nb_impact,
Ticket.priority as nb_priority,
Ticket.status as id_status,
Cat.id as id_category,
Cat.completename as li_category,
CASE Ticket.status WHEN 1 THEN 'Nouveau' WHEN 3 THEN 'Planifié WHEN 4 THEN 'En Attente' WHEN 6 THEN 'Cloturé' WHEN
Ticket.users_id_recipient as id_user,
User.name as li_user,
ISNULL(Item.id, 1) as id_item,
Item.itemtype as li_itemtype,
CASE Item.itemtype WHEN 'Software' THEN Soft.id ELSE '' END as id_software,
CASE Item.itemtype WHEN 'Software' THEN Soft.name ELSE '' END as li_software,
ISNULL(Grp.id, 0) as id_groupe,
Grp.name as li_groupe,
usrTicket.users_id as id_user_attribue,
UserAffecte.name as li_user_attribue,
Req.id as id_requesttype,
Req.name as li_requestype,
CASE Ticket.type WHEN 1 THEN 'Incident' ELSE 'Demande' END as li_type
FROM Ticket/?order=desc&range={?1}&is_deleted=0 as Ticket
INNER JOIN RequestType/ AS Req ON Req.id = Ticket.requesttypes_id
INNER JOIN Entity/ As Entity ON Entity.id = Ticket.entities_id
LEFT JOIN (SELECT tickets_id, MAX(groups_id) as groups_id FROM group_Ticket?order=desc&range=0-15000 GROUP BY ticket
LEFT JOIN (SELECT MAX(users_id) as users_id, tickets_id from Ticket_User/?range=0-20000&order=desc where type = 2 grou
LEFT JOIN User/?range=0-5000 as UserAffecte ON usrTicket.users_id = UserAffecte.id
LEFT JOIN Group/ As Grp ON Grp.id = GrpTicket.groups_id
LEFT JOIN User/?range=0-5000 as User ON User.id = Ticket.users_id_recipient
LEFT JOIN ITILCategory/ as Cat ON Cat.id = Ticket.itilcategories_id
LEFT JOIN Item_Ticket/?order=desc&range=0-20000 as Item ON Item.tickets_id = Ticket.id AND Item.itemtype <> 'Plugi
LEFT JOIN Software/?order=desc&range=0-20000 as Soft ON Soft.id = Item.items_id
```

This example is quite complex and shows how much a query can be made in such a way that it would be carried out in a SGBD.
On the other hand, unlike a SGBD, the header is not known in advance (fields retrieved from the API), it is advisable to do first some "SELECT *" to see what's returned by the API before using joins and other transformations.

The "Sandbox" tab allows you to do some testing.

Example of the GLPI software API:

Simple query:
SELECT * FROM Computer/?range=0-5000
➔ Brings back the list of all computers (5000 entries)

I can of course use some filtering:
SELECT * FROM Computer/?range=0-5000 where states_id = 1
➔ I know that the field "states_id" exists in what brings me the API so I can filter on this field

Similarly, if I know the list of fields, I can ask as follows:
SELECT serial as Serial, name as Machine, users_id as Glpi FROM Computer/?range=0-5000 WHERE states_id=1

SELECT - From a webservice using Fuzible SQL (B)

A Query must start with the table name in which data will be written in (will be created if non-existent).
ex : MyTable:SELECT (...)

```
sample3:SELECT TABLE 1 id_sample,
 substring(li_random_string, 1, 5) as li_substr
FROM /ws/public/sample_data?order=desc[{"id_ssgroup":"'001'"}]
```

For example, this API returns 2 data tables. I can choose to get both of them (will create 2 tables in the Target), or get only one of the 2 :
The "TABLE x" function here determines the data table on which the query applies. The other table will be returned as a simple "SELECT *" statement, because it is not possible to query several tables from a single query.

It is possible to get only one of the 4 tables thanks to the syntax "ONLY":

A Query must start with the table name in which data will be written in (will be created if non-existent).
ex : MyTable:SELECT (...)

```
sample3:SELECT TABLE 1 ONLY id_sample,
 substring(li_random_string, 1, 5) as li_substr
FROM /ws/public/sample_data?order=desc[{"id_ssgroup":"'001'"}]
```

Regarding the API we are querying, we can integrate some body content into the query. The API documentation of the webservice specify how to filter one of the fields using a body using JSON.
➔ The hooks are used to integrate some body content (XML, JSON or Form-data)

On the other hand, unlike a SGBD, the header is not known in advance (fields retrieved from the API), it is advisable to do first some "SELECT *" to see what's returned by the API before using joins and other transformations.
The "Sandbox" tab allows you to do some testing.

SELECT – From Salesforce API using SoSQL

When you have set up the Source connection with a template that has a specific simili-SQL language (graphQL, NxQL, SoQL), you can use this language to query the API instead of using Fuzible SQL, which will make the transcoding work much easier for you.
In the following example, we're querying the Salesforce CRM using SoQL :

Job Queries (SOQL) Sandbox

A Query must start with the filename in which data will be written in.
ex : MyFile.CSV:SELECT (...)

```
salesforce.csv:select name, BillingCity from Account WHERE isDeleted = false
```

While the same query, using Fuzible SQL mode, looks like this, which is much less convenient:

Job Queries (Fuzible SQL) Sandbox

A Query must start with the filename in which data will be written in.
ex : MyFile.CSV:SELECT (...)

```
salesforce.csv:select * from /query/?q=SELECT+name,BillingCity+from+Account+WHERE+isDeleted+=+false
```

SELECT - From an e-mail box

A Query must start with the table name in which data will be written in (will be created if non-existent).
ex : MyTable:SELECT (...)

```
myOutputTable:select TOP 3 DATE, FROM, SUBJECT, UID from mymail@gmail.com
myOutputTable2:select TOP 10 DATE as myDate, FROM as From, SUBJECT as mySubject, UID as IdMail
from myothermail@gmail.com[myPassword]
WHERE DATE >= '01/01/2020'
```

Example of a GMAIL address:

SELECT * FROM mymail@gmail.com[mypassword]

➔ Will bring back the emails list from the address "mymail@gmail.com" using the password entered in brackets.

SELECT * FROM mymail@gmail.com

➔ Will bring back the emails list from the "mymail@gmail.com" address using the password entered in the connection string.

I can of course filter and name the columns if I know them (recall: the SANDBOX is made for this):
SELECT SUBJECT, SENDER, TEXTBODY, TO FROM mymail@myProvider.com WHERE DATE >= '01/01/2019'

SELECT - From Active Directory

A Query must start with the filename in which data will be written in.
ex : MyFile.CSV:SELECT (...)

Quick Help

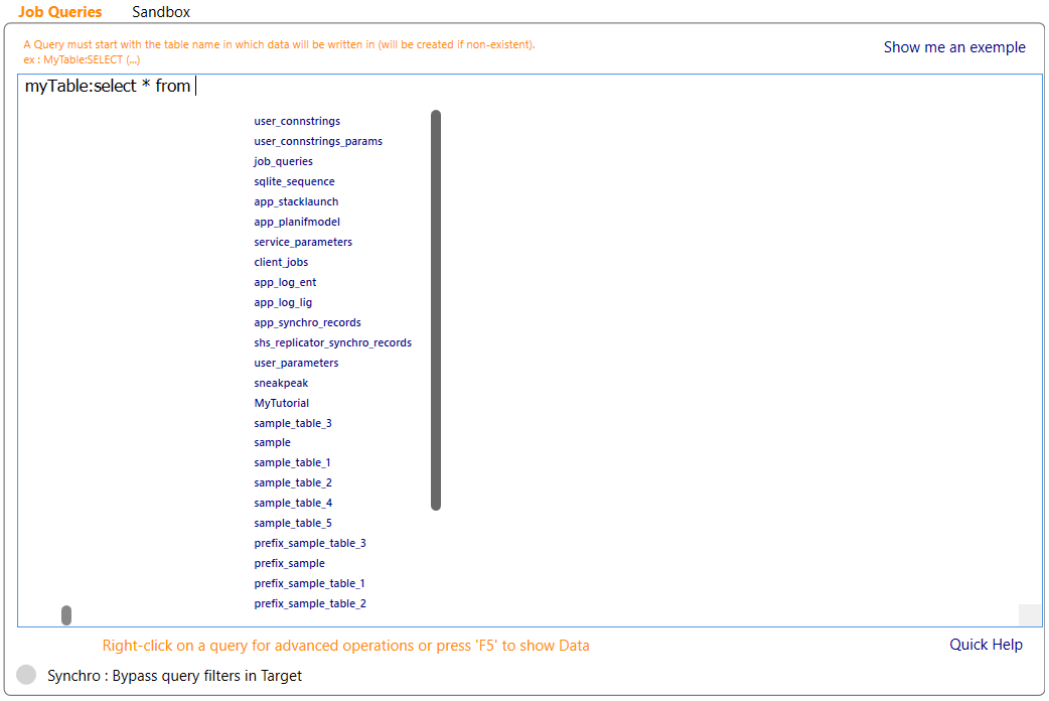
```
test_ad_grp.csv:select * from groups
```

As with webservices and maiboxes, Fuzible engine does not know in advance all the fields that can be returned by the AD domain. It is advisable, to test to make a simple SELECT * FROM USERS
"FROM" refers to the AD object being queried (USERS or GROUPS)
The example brings back the list of AD groups that exist.

QUERY ASSISTANT

Like SQL Tools, the software offers query entry facilities through a pop-up menu that displays based on the words you are writing:

In the following example, I am querying an SGBD and just have entered "FROM": the menu then shows me a list of available tables. A simple "TAB" allows me to access this floating menu, the arrows allow to choose a table, while "ENTER" inserts the chosen item in the query.



In the following example, I already have a table and I complete my "SELECT" statement: the list of fields is presented, as well as the classic SQL functions that can be used.

The Query Assistant works for all types of Sources, it is much more advanced for file queries and SGBD because for other cases, there is no method to, for example, expose the list of objects of a webservice and even less the list of available fields. In this case, it simply serves as an assistant to the creation of an SQL function (CONCAT, ISNULL…).

ADVANCED "SELECT" STATEMENT

The previous examples are relatively simple, they show how one can basically query a Source in the form of "SELECT" when it is not a database.

Fuzible's SQL language understands more complex syntax, and writing an SQL query into the tab is augmented by a Query Assistant that allows you to see and use all available query options on the fly.

If for example I type "SELECT C", a small pop-up menu will offer me several things:

- CONCAT
- CASE
- COALESCE
- CHARINDEX
- COUNT
- …

I can then use one of these elements to manipulate the source data. Examples:

- SELECT CONCAT(champ1, champ2) as dest1 [...]
- SELECT CASE champ1 when 'OUI' then 1 WHEN 'NON' then 2 ELSE 0 END as dest2 [...]
- SELECT COUNT(champ1) as dest1, champ2 FROM MONFICHIER GROUP BY champ2

To measure the full range of possibilities offered by the SQL engine, you have a default "Sample" Job that has several queries using all of these advanced features. You can learn from it.

There's also some kind of "Anonymization" feature :

- SELECT ANONYMIZE(myField) FROM myFile
  Will mix "myField" values between them
- SELECT ANONYMIZE(myField, "RANDOM") FROM myFile
  Will randomize "myField" values
- SELECT ANONYMIZE(myField, "myRandomizationFile.csv") FROM myFile
  Will load the specified file and use the values to replace original "myField" values

Fuzible SQL Technical limitations:

1) Nested SELECT only work on "tables" and "where" filters (see example "webservices A"), you can't make a sub-select for a field:

Valid:

A Query must start with the filename in which data will be written in.
ex : MyFile.CSV:SELECT (...)

Quick Help

```
test:select *
from monfichier as a
inner join (select MAX(id) as id, monclient as id_client from unfichier where typeclient = 'local') as b ON a.id = b.id
```

Valid:

A Query must start with the table name in which data will be written in (will be created if non-existent).
ex : MyTable:SELECT (...)

Quick Help

```
test:select *
from (select * from monfichier) as a
```

Invalid:

A Query must start with the filename in which data will be written in.
ex : MyFile.CSV:SELECT (...)

Quick Help

```
test:select a.*,
(select MAX(id) as id, monclient as id_client from unfichier where typeclient = 'local') as b
from monfichier as a
```

2) You cannot nest the sub-selects. The query will not succeed.
ex: *SELECT * FROM (SELECT * FROM (SELECT * FROM monfichier) as data) as data*

Additional note:

Some sources may return multiple data tables (webservices, JSON or XML files).

By default, the software will create as many targets as returned tables. However, all SQL processing operations associated with your query (SQL, Group By, Where, Order By...) will be applied to the first table only.

For some reason, if you want your query to apply to another table, it's possible to indicate it by doing so:

OUTPUT :SELECT **TABLE x** champ1, ...

... indicating, thanks to "TABLE x," the table number to which the query applies.

If "x" is invalid (example: I put TABLE 10 when the source returns only one table, all operations associated with the query will be cancelled.

*Of course, you cannot guess in advance what the data source is made of. The "sandbox" tab is there for that. Make a simple "SELECT * FROM masource" and an **F5** to find out all the results returned by the query. If the result returns multiple tables, you will see it, and then you'll be able to see the contents of each table.*

You can also say "ONLY" if you only want to retrieve the requested table (otherwise, the others will also be processed).

Exemple : SELECT TABLE x ONLY champ 1,...

Look at the example job to understand what can be done and most importantly, experiment using SANDBOX. The program is verbose enough to allow you to debug step by step a query that would be wrongly written.

SELECT Multi-tables, multi-files

BDD

You can get multiple tables at once, if for example you want to retrieve all the tables that contain the word "param," you can enter the query as well:

OUTPUT : SELECT * FROM %param%

Fuzible detects the use of the "%" and will search for all the tables that match this pattern. It then turns a single query into several. It goes without saying that unless the names of the fields of these tables are all the same, the "SELECT *" is recommended...

There are three scenarios for driving the output:

- If you put « * » (ex: *:select * from %param%), « * » will be replaced by the name of the table.
- If you put a forced name (ex : param:select * from %param%), the data will all go in the same output, here: "param". All data coming from any query with that pattern will be merged.
- If you put "import_*" (ex : param_* :select * from %param%), the "*" will be replaced by the name of the table, but the final table will have a name that will begin with « import_ »

FICHIERS

In the same way as with a database, you can query multiple files at once, for example, if you want to retrieve all the files that contain the word "param," you can enter the query like this (like a "DIR" in the MSDOS command prompt)

OUTPUT :SELECT * from *.CSV

Fuzible detects the use of the "*" and will query all CSV files from the connection string (path).

The use of the '*' can be extended as follows:

OUTPUT :SELECT * from FI*test*.*

- ➔ This means that files starting with "FI" and then having something else afterwards, then "test", then something else, will be loaded.

As with databases, you can name fields and work the query in an advanced way (with transformation functions... etc. ..) but in this case, the files must be of the same structure and, if they have a header, it must be the same.

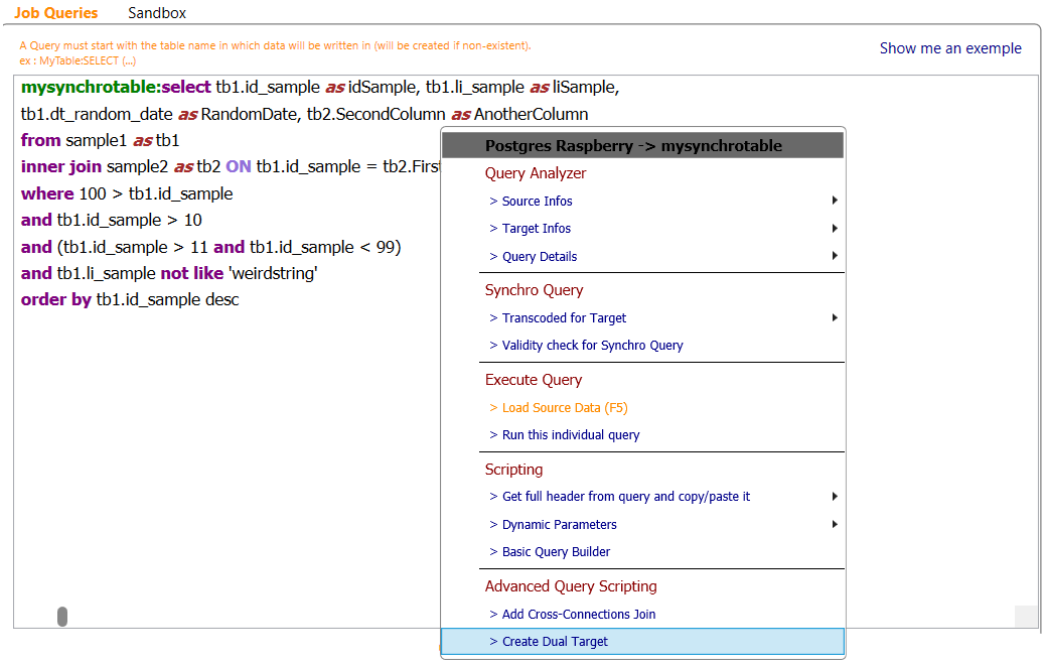
Driving the Output works in the same way as previously explained.

Multi-Target Queries

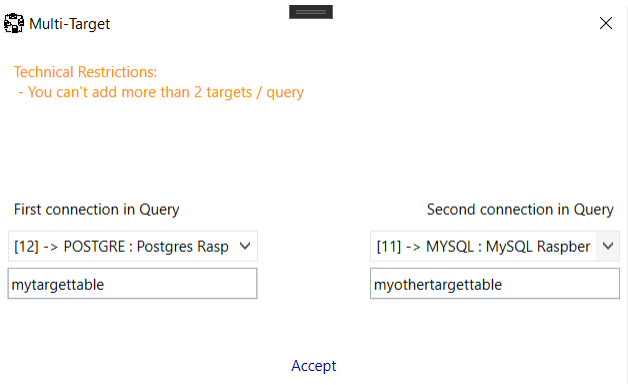
By default, a Job works with a single Target.

However, this can be changed to work with 2 Targets in parallel (in case for example we want to feed 2 databases identically, simultaneously)

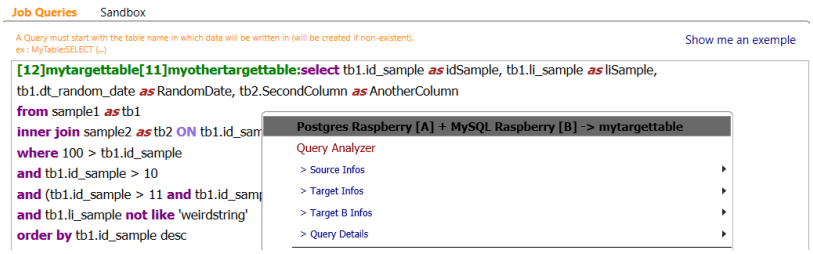
Let us take a simple example: build a Job query, then click right on it. A pop-up menu appears and offers you several options including "Create Dual Target"



A prompt will then ask you to choose your 2 Targets, and the associated output. Confirm.



The query will be augmented by a small script:



Between brackets, the Connection ID, followed by the name of the target table. A new right-click to view the pop-up menu, to which a "Target B Info" item has been added, that provides information about the chosen Target.

But we can also imagine filling a database table, and a file, in parallel. Anything is possible. Target connections can even be dynamic using the Dynamic Parameters:

The query:

Job Queries Sandbox

A Query must start with the table name in which data will be written in (will be created if non-existent).
ex : MyTable:SELECT (...)

```
{?1}mysynchrotable{?2}mysynchrofile.csv:select tb1.id_sample as idSample, tb1.li_sample as liSample,
tb1.dt_random_date as RandomDate, tb2.SecondColumn as AnotherColumn
from sample1 as tb1
inner join sample2 as tb2 ON tb1.id_sample = tb2.FirstColumn
where 100 > tb1.id_sample
and tb1.id_sample > 10
and (tb1.id_sample > 11 and tb1.id_sample < 99)
and tb1.li_sample not like 'weirdstring'
order by tb1.id_sample desc
```

Dynamic parameters:

Main Parameters

Job Type: Data Synchronization | Allow Delete, Update, Insert | Historize UPDATED and DELETED rows

Will compare source and target data, and update target according to what's in the source

Dynamic Parameters: [12];[1]

You can write any text or any available command. Each parameter must be separated by a semicolon (;)
You can use them anywhere (queries, text fields, connections) by referencing them like this : {?1}, {?2}...

View Job With Replaced Values

This is of interest when you want to switch data from one environment to another (pre-production,production, developmen, etc.) on the fly.

Moreover, one can choose to fill the 2 Targets in parallel or one after another, it all depends on the performance of the computer that hosts Fuzible (see General Configuration / SHS Analyzer)

<u>Technical limitation:</u>

Multi-Target is limited to 2 targets.

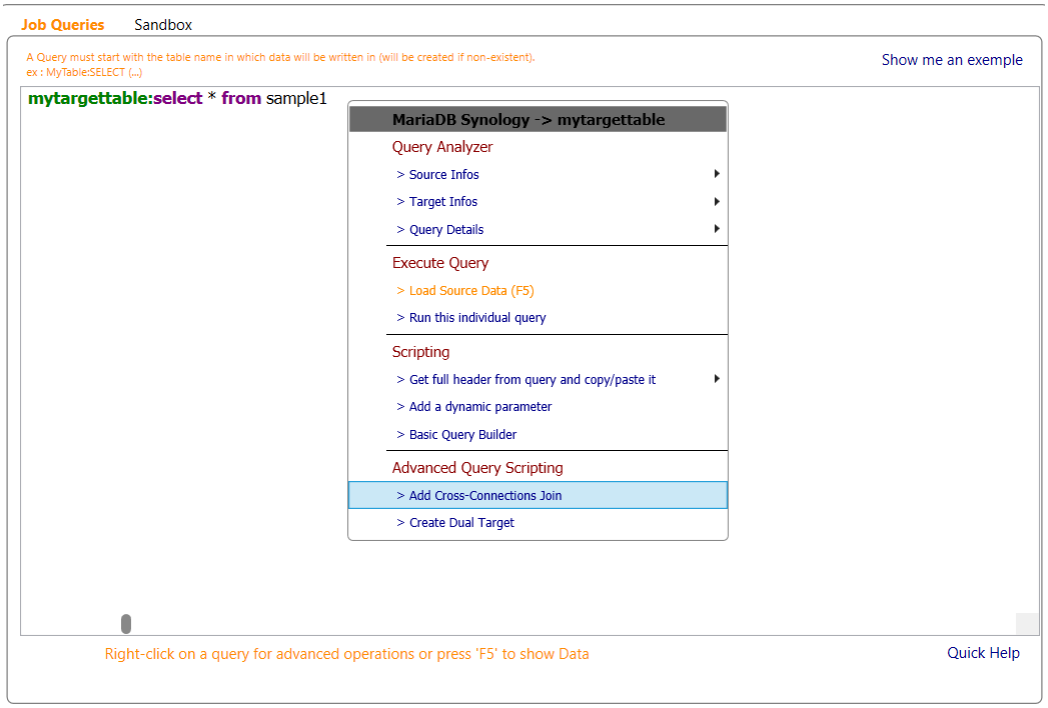
Cross-Queries

This feature is one of Fuzible's most powerful. It allows you to query different data sources within a single query: you can get data from a database from a file and complete it with data from a webservice. Anything is possible and can be achieved in a fairly simple way.

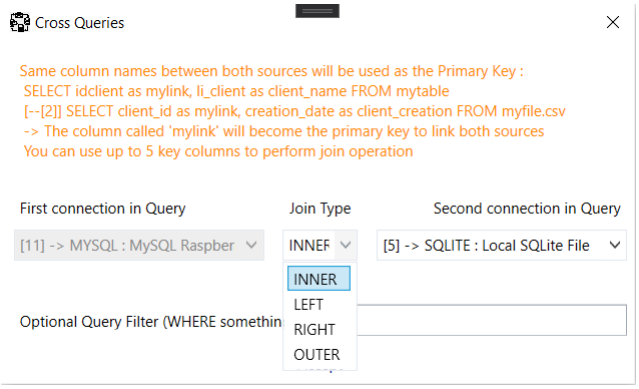
To make this concept clearer, imagine that instead of joining between several SQL tables to complete a dataset, you make joins between several different connections.

Let us take the following example. My main connection is a MySQL database.

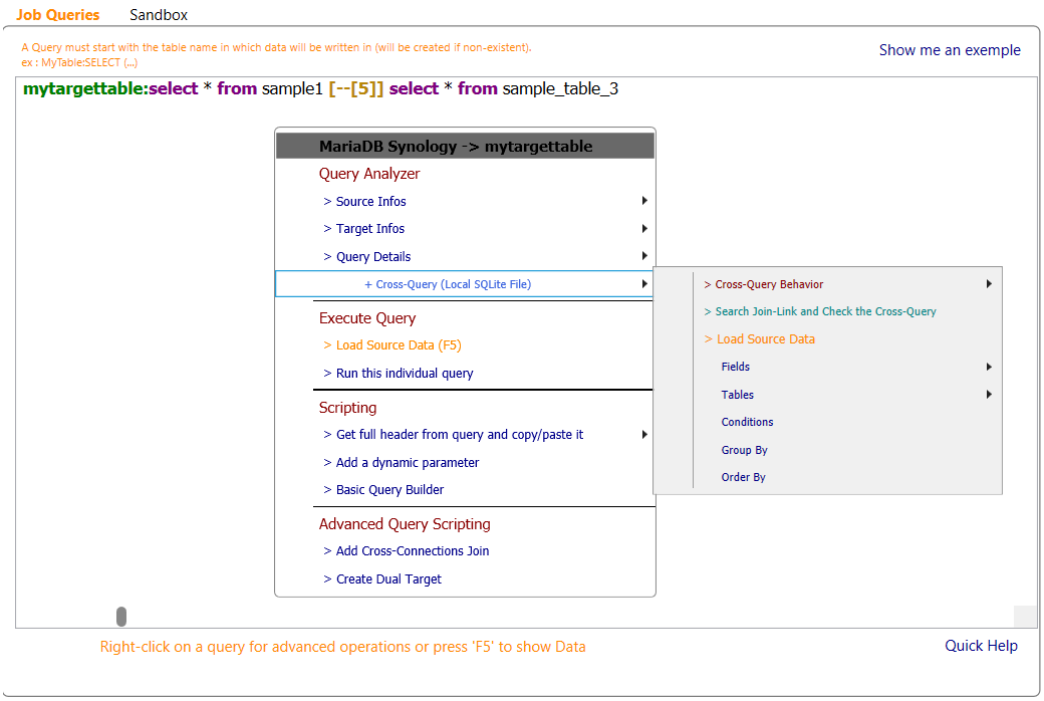
Build a simple query, then click right on it. The pop-up menu appears and offers several options including "Add Cross-Connections Join."



A prompt then asks you to choose a new data Source, and the type of join that will be made between the two Sources. I pass on the very advanced "Optional Query Filter" feature, which filters the results.



A script area [--[5]] will appear in the query. You can then write a new query. This one will be associated with the second connection. A right-click and you will find that the menu contains a new sub-menu associated with this cross-query.



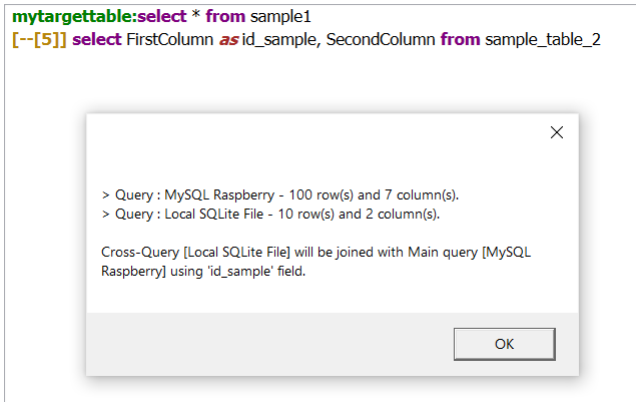
In the script area, the type of join expresses itself as follows:

--	INNER JOIN
>-	RIGHT JOIN
<-	LEFT JOIN
<>	OUTER JOIN

The connection is shown between brackets.

Fuzible has only one way to define how to join the two sources between them: he uses the fields with the same name to make his join. In the previous example, a "SELECT *" is performed on both sources, but I know that the "id_sample" field is exists on both sources, Fuzible will join using this field.

On the other hand, if the fields names are different from one connection to another, the fields should be specifically named. For example:

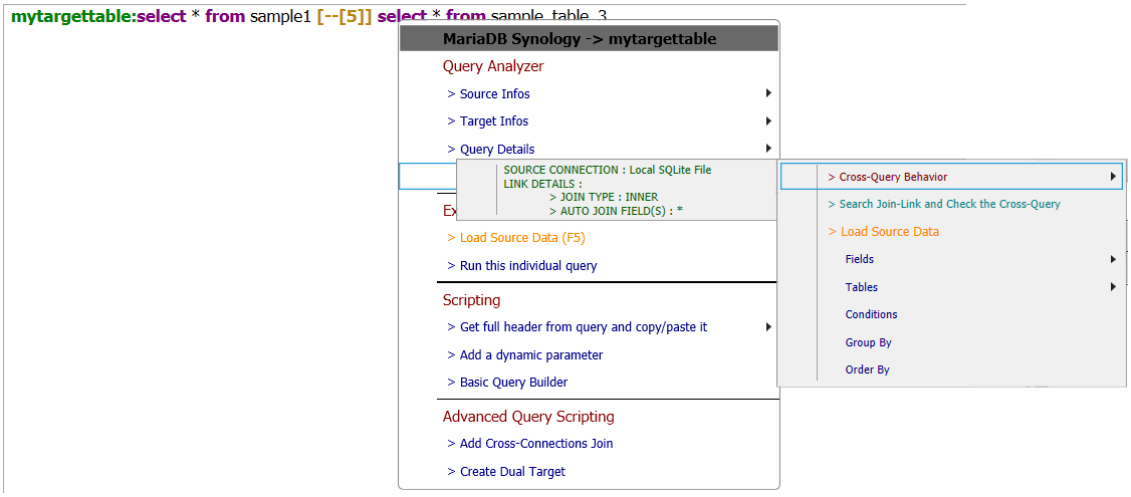


Thanks to the alias, I force the link between the two sources with "id_sample".

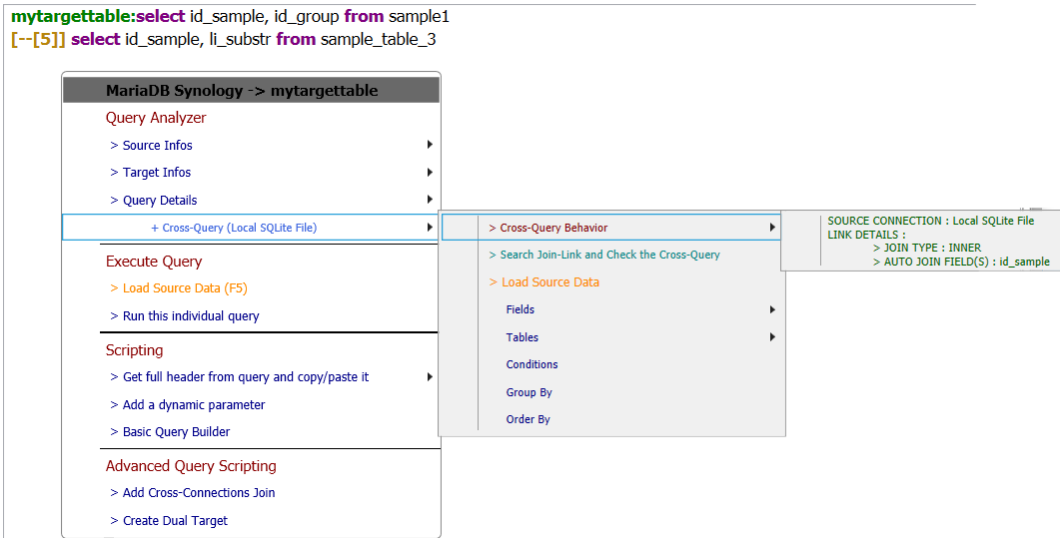
Behavior of an cross-query

The contextual menu makes it easier for you to understand Fuzible's behavior related to the cross-query. In the example below, SELECT * being used in both connections, the engine will only be able to determine the link when the queries are executed: You need to be sure that the "id_sample" field exists in both datasets.

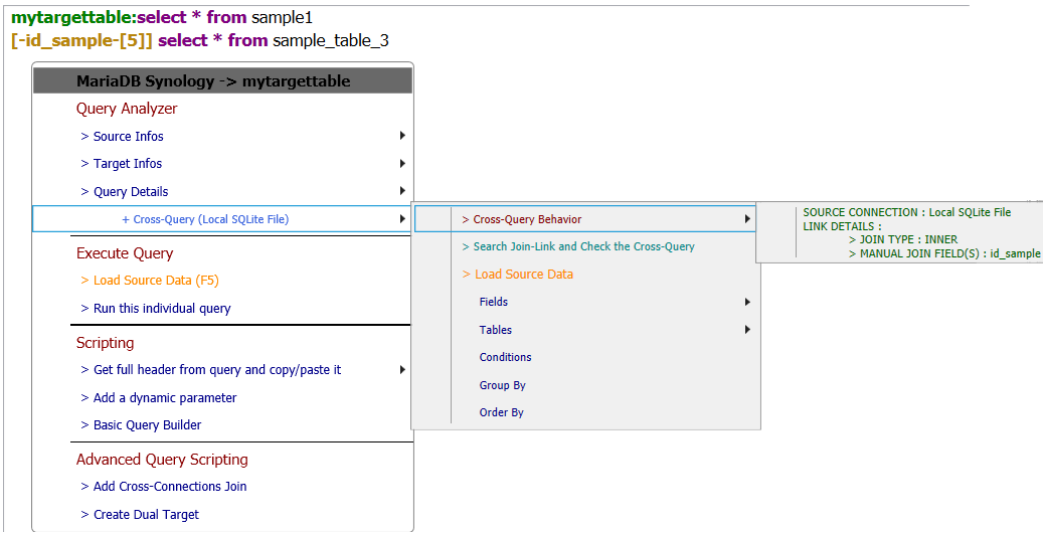
Warning: Any other field with the same name would also be considered as part of the key!



Another way is to make a SELECT statement that specifies fields names. The contextual menu then tells you which join field or fields are associated with it. There is no need to wait for the execution to know the behavior that will be performed.

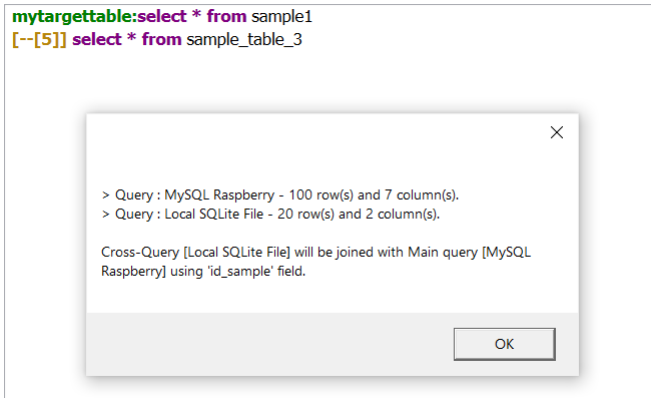


Finally, the third way is to change the cross-query script: you can manually define the join field(s) by separating them by a comma. The contextual menu specifies that the type of join is now manual.

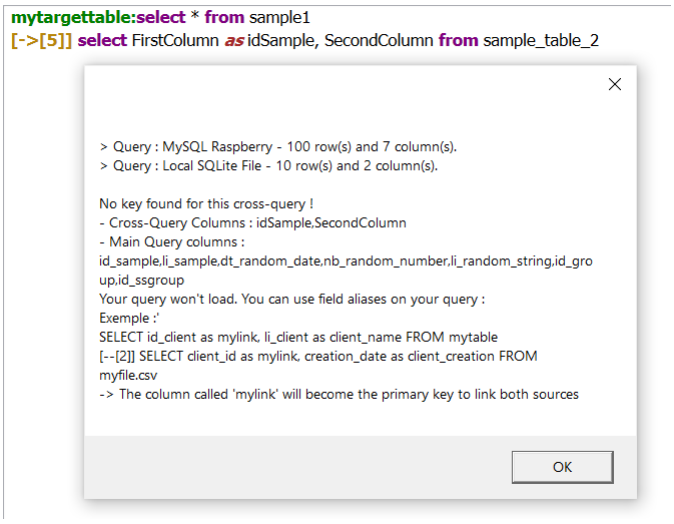


Results

By going to "Cross-Query/Search Join-Link and Check the Cross-Query", Fuzible will perform a test to check the viability of the cross-query. The result will be presented as follows:



On the other hand, if there is no link, here is what will be displayed; in this example, I forced the names of the columns of the second Data Source, putting an "idSample" name that does not exist in the first Data Source (id_sample).

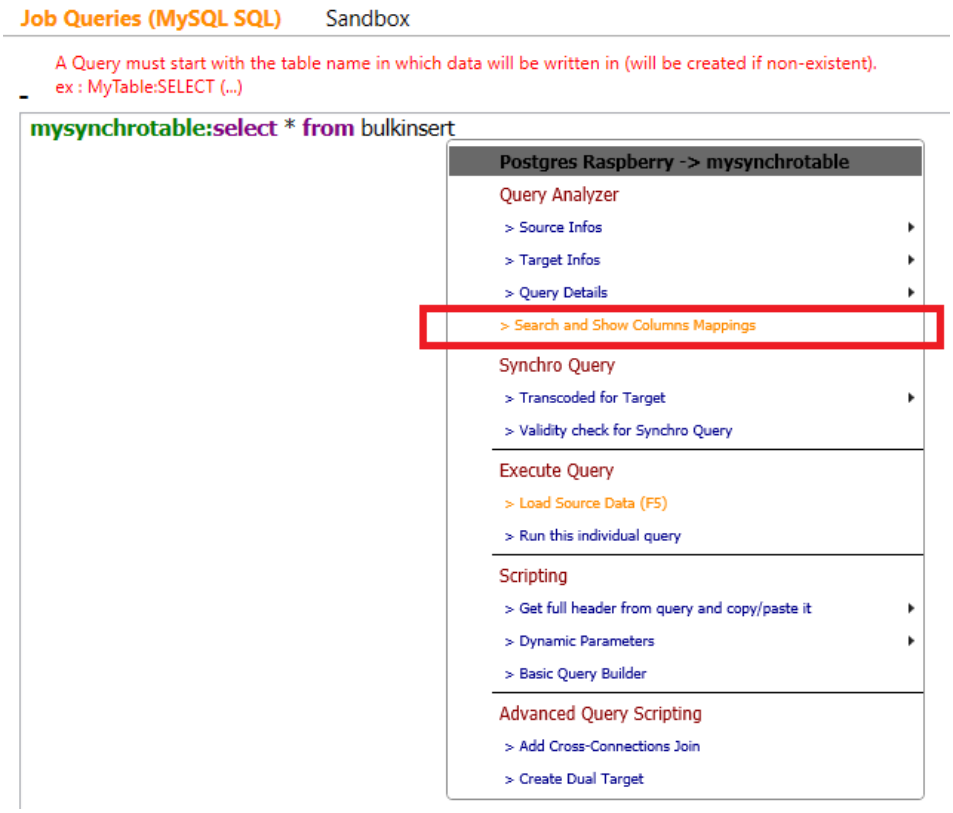


Search and Show Column Mappings

Because one of the aspects of Fuzible is to avoid the well-known ETL mapping tasks, there's an option that allows you to see the mapping that will be performed between Source and Target columns, and if there are some orphan columns, a Query suggestion will also be showned if you need those columns to be mapped as well.

Fuzible will automatically find the closest name correspondances from the orphan Source and Target columns.

You'll be able to copy-paste the suggested Query instead of your existing one.



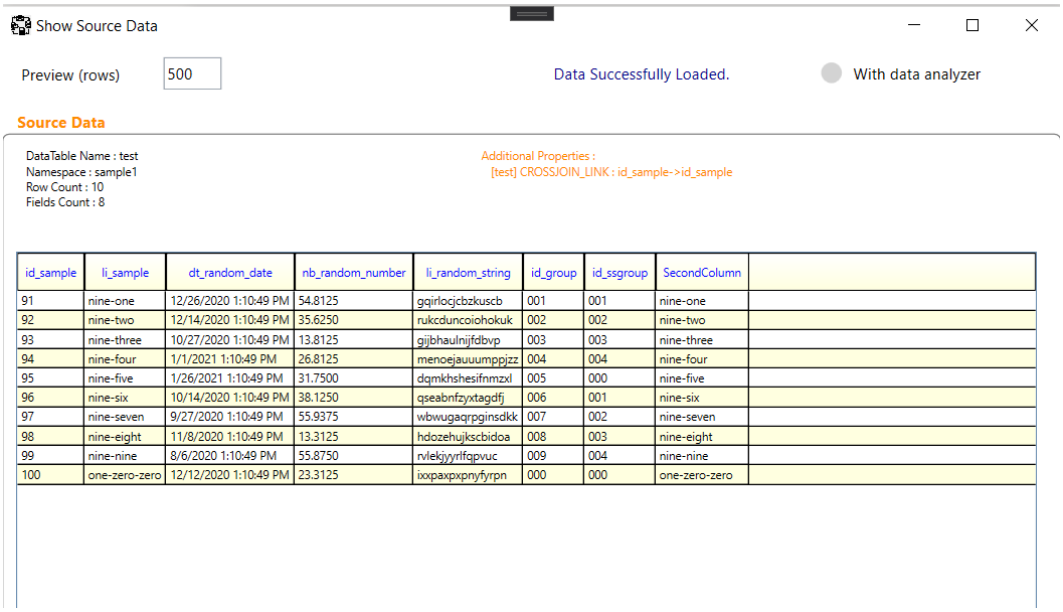
Showing Source Data

The contextual menu contains a "Load Source Data" option that allows you to view the results, whether it is all the data from the cross-query, or just the data from each individual query.

In the table below, I made a cross-query using a "LEFT JOIN", and loaded the data from the cross-join between the two Sources. Only "id_sample" (from 90 to 100) records are present in the second connection, which explains why "SecondColumn" is empty in other cases.

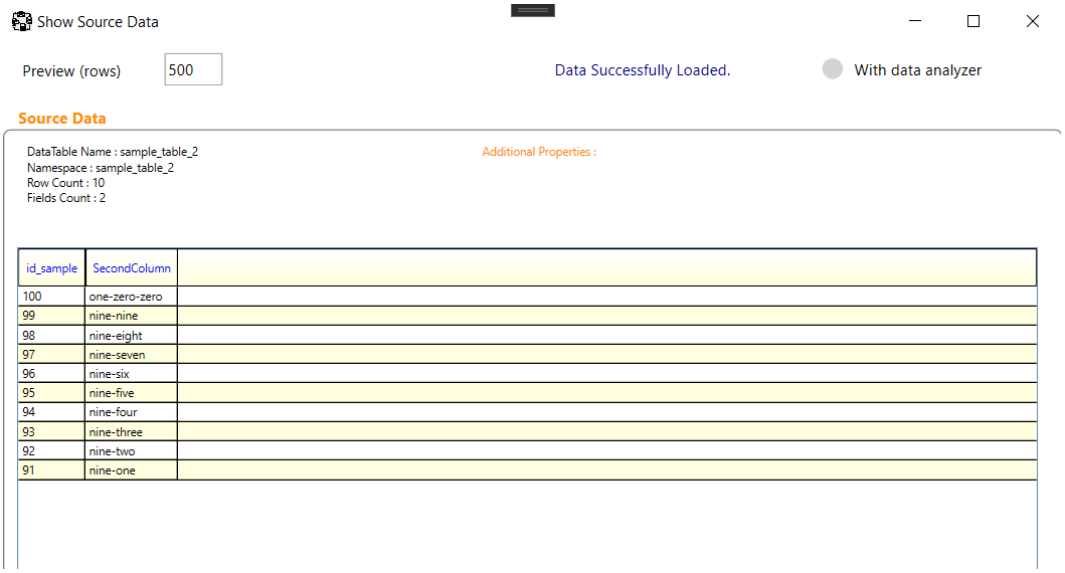
id_sample	li_sample	dt_random_date	nb_random_number	li_random_string	id_group	id_ssgroup	SecondColumn
70	seven-zero	10/8/2020 1:10:49 PM	9.0625	piynoetghykxighe	000	000	
71	seven-one	10/19/2020 1:10:49 PM	31.0625	hsxnmgwrqwhvjnyc	001	001	
72	seven-two	10/27/2020 1:10:49 PM	59.6250	kudzjrucwhciottx	002	002	
73	seven-three	12/19/2020 1:10:49 PM	33.6250	dbvsqxnxzmpxzgui	003	003	
74	seven-four	1/22/2021 1:10:49 PM	41.6250	iijftkkooihnclam	004	004	
75	seven-five	11/13/2020 1:10:49 PM	29.1250	dsdfmlsimyiwnfzc	005	000	
76	seven-six	8/31/2020 1:10:49 PM	16.4375	dxhkhbjeqljdxnhj	006	001	
77	seven-seven	9/24/2020 1:10:49 PM	9.0625	mcvxnlymgsubozwv	007	002	
78	seven-eight	11/19/2020 1:10:49 PM	7.1250	cskbwhvhyiavoeay	008	003	
79	seven-nine	9/12/2020 1:10:49 PM	33.1250	ifkwmvvqjpwkhceh	009	004	
80	eight-zero	10/25/2020 1:10:49 PM	34.4375	qkgocfmplkdqbpjk	000	000	
81	eight-one	9/17/2020 1:10:49 PM	12.4375	pffogfxgkobyrgwn	001	001	
82	eight-two	12/2/2020 1:10:49 PM	16.8750	gudejxklntohaxpb	002	002	
83	eight-three	8/18/2020 1:10:49 PM	8.6875	hceaqwvynmswtnhe	003	003	
84	eight-four	9/5/2020 1:10:49 PM	45.5000	zlwieuserdoejppv	004	004	
85	eight-five	8/3/2020 1:10:49 PM	18.3750	afegjljxsqzazvtx	005	000	
86	eight-six	8/26/2020 1:10:49 PM	20.3125	fyxkileskjddhyhc	006	001	
87	eight-seven	11/25/2020 1:10:49 PM	58.3125	nsbkygcbfrdslmni	007	002	
88	eight-eight	9/28/2020 1:10:49 PM	22.0625	hqxscraatpibsxqv	008	003	
89	eight-nine	12/23/2020 1:10:49 PM	25.5000	euaqhggnorxrpyha	009	004	
90	nine-zero	8/19/2020 1:10:49 PM	38.0000	avqkjxnntdygcqce	000	000	
91	nine-one	12/26/2020 1:10:49 PM	54.8125	gqirlocjcbzkuscb	001	001	nine-one
92	nine-two	12/14/2020 1:10:49 PM	35.6250	rukcduncoiohokuk	002	002	nine-two
93	nine-three	10/27/2020 1:10:49 PM	13.8125	gijbhaulnijfdbvp	003	003	nine-three
94	nine-four	1/1/2021 1:10:49 PM	26.8125	menoejauuumppjzz	004	004	nine-four
95	nine-five	1/26/2021 1:10:49 PM	31.7500	dqmkhshesifnmzxl	005	000	nine-five
96	nine-six	10/14/2020 1:10:49 PM	38.1250	qseabnfzyxtagdfj	006	001	nine-six
97	nine-seven	9/27/2020 1:10:49 PM	55.9375	wbwugaqrpginsdkk	007	002	nine-seven
98	nine-eight	11/8/2020 1:10:49 PM	13.3125	hdozehujkscbidoa	008	003	nine-eight
99	nine-nine	8/6/2020 1:10:49 PM	55.8750	rvlekjyyrlfqpvuc	009	004	nine-nine
100	one-zero-zero	12/12/2020 1:10:49 PM	23.3125	ixxpaxpxpnyfyrpn	000	000	one-zero-zero

Conversely, an "INNER JOIN" would have given this:



id_sample	li_sample	dt_random_date	nb_random_number	li_random_string	id_group	id_ssgroup	SecondColumn
91	nine-one	12/26/2020 1:10:49 PM	54.8125	gqirlocjcbzkuscb	001	001	nine-one
92	nine-two	12/14/2020 1:10:49 PM	35.6250	rukcduncoiohokuk	002	002	nine-two
93	nine-three	10/27/2020 1:10:49 PM	13.8125	gijbhaulnijfdbvp	003	003	nine-three
94	nine-four	1/1/2021 1:10:49 PM	26.8125	menoejauuumppjzz	004	004	nine-four
95	nine-five	1/26/2021 1:10:49 PM	31.7500	dqmkhshesifnmzxl	005	000	nine-five
96	nine-six	10/14/2020 1:10:49 PM	38.1250	qseabnfzyxtagdfj	006	001	nine-six
97	nine-seven	9/27/2020 1:10:49 PM	55.9375	wbwugaqrpginsdkk	007	002	nine-seven
98	nine-eight	11/8/2020 1:10:49 PM	13.3125	hdozehujkscbidoa	008	003	nine-eight
99	nine-nine	8/6/2020 1:10:49 PM	55.8750	rvlekjyyrlfqpvuc	009	004	nine-nine
100	one-zero-zero	12/12/2020 1:10:49 PM	23.3125	ixxpaxpxpnyfyrpn	000	000	one-zero-zero

The data from the cross-query:



id_sample	SecondColumn
100	one-zero-zero
99	nine-nine
98	nine-eight
97	nine-seven
96	nine-six
95	nine-five
94	nine-four
93	nine-three
92	nine-two
91	nine-one

Cross-queries, data comparison feature

This mode allows you for example to compare data from two databases (to make a report of differences between the two, for example)

By default, the "cross-query" feature is simple to use. But like any automatic feature, it is rigid. However, it can be made more flexible by filtering the results.

This example compares tables in a production database with a pre-production database (Postgres)

A Query must start with the mail address(es) for which you want to send data to (addresses have to be separated by a ',').
ex : mymail@gmail.com,anymail@yahoo.fr:SELECT (...)

```
mail@mail.com:select table_name, table_type,
'Oui' as présente_en_prod
from information_schema.tables As Tables_absentes_en_prod_ou_en_préprod
where table_name not like 'pg%' and table_type <> 'VIEW'
order by table_type, table_name
[<>[75]WHERE présente_en_prod IS NULL OR  présente_en_préprod IS NULL]
select table_name, table_type,
 'Oui' as présente_en_préprod
from information_schema.tables
where table_name not like 'pg%' and table_type <> 'VIEW'
order by table_type, table_name
```

You can see that the framed script area [<>[75]...] uses a conditional statement that allows comparison between the two result sets.

➔ This way of scripting allows, once the data is collected, to filter the results according to the filter shown in the script area.

If the target is an email address, the result will be as follows:

Tables absentes en prod ou en préprod : 21 record(s)

Table name	Table type	Présente en prod	Présente en préprod
param_webservices	BASE TABLE		Oui
cometepreprod_cii_agents	FOREIGN TABLE		Oui
cometepreprod_cii_mois_agent	FOREIGN TABLE		Oui
cometepreprod_planning_cloture	FOREIGN TABLE		Oui
cometepreprod_ressources	FOREIGN TABLE		Oui
dbg_absence_comete	BASE TABLE	Oui	
dbg_absence_rhpi	BASE TABLE	Oui	
dbg_calendar	BASE TABLE	Oui	
dbg_contrat_comete	BASE TABLE	Oui	
dbg_contrats_rhpi	BASE TABLE	Oui	
dbg_generateur_documents	BASE TABLE	Oui	

Contextual Query Menu

The principle of using SQL to Synchronize or Replicate data can represent a fairly significant level of abstraction. That is why Fuzible has a useful contextual menu, accessible when the mouse is on a query. A right click and you can do the following:

- Get advanced information about the query

- Test the validity of the Sync. query (Synchronization Mode)

- Execute any of the job's queries without necessarily executing them all

- To have a script assistant for advanced functions

QUERY ANALYZER - News Source

Source Connection Summary

FILE	For a local file, the path and type of connection MyTutorial:SELECT id_sample, SUBSTRING(li_sample, 0, 5) as li_sample, id_group as RenommedColumn, dt_random_date, li_sample as li_test, Local SQLite File -> MyTutorial Query Analyzer > Source Infos > Target Infos DRIVER : CSV File CONN. NAME : Local Path INPUT PATH : C:\Users\guizm\source\Workspaces\Fuzible\Fuzible\bin\x64\Debug\FILES\ SOURCE FILE(S) : SAMPLE.CSV For a (S)FTP, the configuration of the (S)FTP and the path on the server test:select * from /var/www/sftp/SAMPLE.CSV Local Path -> test Query Analyzer > Source Infos > Target Infos > Query Details Execute Query DRIVER : CSV File CONN. NAME : SFTP Fuzible (S)FTP URL : PATH : IS SFTP : True USERNAME, PASSWORD : PORT : 22 AUTHENTIFICATION BY KEY FILE : False SOURCE FILE(S) : /var/www/sftp/SAMPLE.CSV
BDD	For an SQL connection, the driver, and the name of the database MyTutorial.csv:SELECT user as userColumn, connstring_id as idColumn, connstring_name as nameColumn from user_connstrings WHERE user LIKE '{74}' Local Path -> MyTutorial.csv Query Analyzer > Source Infos > Target Infos DRIVER : Sqlite Database CONN. NAME : Local SQLite File DATABASE :
WS	Shows the whole behavior Fuzible uses to call the API. Check the URL, the authorization method. sample1:SELECT * FROM /ws/public/sample_data MySQL Raspberry -> sample1 Query Analyzer > Source Infos > Target Infos > Query Details DRIVER : Webservice REST CONN. NAME : Fuzible Webservice de démonstration BASE URL : https://www.fuzible-app.com FULL URL : https://www.fuzible-app.com/ws/public/sample_data AUTHORIZATION METHOD : API_AUTH_HEADER > TOKEN WILL BE ADDED TO QUERY HEADER : Authorization: METHOD : POST
MAIL	Summary of settings used to connect to a mailbox mytable:select TOP 3 DATE, FROM, SUBJECT, UID from gui Hotmail Connection -> mytable Query Analyzer > Source Infos > Target Infos > Query Details Execute Query > Load Source Data (F5) > Run this individual query DRIVER : Mailbox CONN. NAME : Gmail Connection SMTP HOST : smtp.gmail.com POP/IMAP HOST : imap.gmail.com SMTP PORT : 587 POP/IMAP PORT : 993 SOURCE MAIL : SENDER : PASSWORD : SSL : True PROTOCOL : IMAP AUTHENTIFICATION PROTOCOL : TLS12 UNKNOWN FIELD(S) : SUBJECT,UID INFO : Use 'SELECT *' if you don't know Mail fields names. It will retrieve all available fields. Otherwise, you can for exemple write a query like this : 'SELECT BODY, FROM, DA
AD	The search query, the perimeter ad_users:SELECT * FROM users MySQL Raspberry -> ad_users Query Analyzer > Source Infos > Target Infos > Query Details DRIVER : Active Directory CONN. NAME : Fuzible Active Directory Demo AD OBJECT : (&(objectClass=user)(objectCategory=person)) SEARCH SCOPE : AD_BASE INFO : Use 'SELECT *' if you don't know AD fields names. It will retrieve all available fields. Otherwise, you can for exemple write a query like this : 'SELECT name, samaccountname FROM users'

QUERY ANALYZER - Target Info

BDD	- The name of the target database - The destination table - Any pre/post-Job command(s) that have been set **mytable:select** * **from** sample **MySQL Raspberry -> mytable** Query Analyzer > Source Infos > Target Infos > Query Details DRIVER : MySQL/MariaDB Database CONN. NAME : MySQL Raspberry DATABASE : test TABLE NAME : mytable
FILE	- All the information about the target: the path, and, in case of an "OUTPUT"with a pattern (creation of several files), the behavior that will be adopted. - Any pre/post-Job command(s) that have been set CSV: **myfile.csv:select** id_sample, li_sample **from** sample **Local Path -> myfile.csv** Query Analyzer > Source Infos > Target Infos > Query Details DRIVER : CSV File CONN. NAME : Local Path OUTPUT PATH : C:\Users\guizm\source\Workspaces\Fuzible\Fuzible\bin\x64\Debug\FILES\ OUTPUT FILENAME : myfile.csv CSV HEADER : id_sample;li_sample XML: **myfile.xml:select** id_sample, li_sample **from** sample **Local Path -> myfile.xml** Query Analyzer > Source Infos > Target Infos > Query Details DRIVER : XML File CONN. NAME : Local Path OUTPUT PATH : C:\Users\guizm\source\Workspaces\Fuzible\Fuzible\bin\x64\Debug\FILES\ OUTPUT FILENAME : myfile.xml XML STRUCTURE : <?xml version='1.0'?><sample><Row><id_sample>value</id_sample><li_sample>value</li_sample></Row></sample> EXCEL: **myexcelfile.xlsx:select** * **from** sample ***as*** MySpreadSheet **Local Path -> myexcelfile.xlsx** Query Analyzer > Source Infos > Target Infos > Query Details Execute Query > Load Source Data (F5) DRIVER : Excel File CONN. NAME : Local Path OUTPUT PATH : C:\Users\guizm\source\Workspaces\Fuzible\Fuzible\bin\x64\Debug\FILES\ OUTPUT FILENAME : myexcelfile.xlsx EXCEL SHEET NAME : MySpreadSheet TITLE ROW : My Excel File STYLE : Light1 CSV HEADER : Will be determined at runtime. ('SELECT *' was used in Source Query. Fuzible cannot retrieve column names now)
WS	**webService=dossier.** ***{?2}.{?1}*** **:SELE** **Webservice RHPI (Personnel/Contrats/RIC) -> webService=dossier.ric.modifier** Query Analyzer > Source Infos > Target Infos > Load Source Data (F5) > Query Details Execute Query > Run this individual query Scripting > Get full header from query and copy/paste it > Dynamic Parameters > Basic Query Builder Advanced Query Scripting > Add Cross-Connections Join > Create Dual Target BASE URL : https://[illegible]?webServiceApplication=STANDARD FULL URL : https://[illegible]?webServiceApplication=STANDARD&webService=dossier.ric.modifier AUTHORIZATION METHOD : HTTP > AUTH. PARAMS WILL BE ADDED TO QUERY PARAMS : https://[illegible] HTTP QUERY : Will be determined at runtime('SELECT *' was used in Source Query. Replicator cannot retrieve column names now) RESPONSES SAVED IN SOURCE : log_nebula_webservice_ric_vers_rhpi Responses fields : - WS_QUERYTARGETNAME : modifier_ric - WS_QUERY : Will be determined at runtime('SELECT *' was used in Source Query. Replicator cannot retrieve column names now) - WS_ANSWERDATA : Will be determined at runtime : will contain webservice answer message - WS_ROWID : Source row position - DTLOAD : Date of query - DBNAME : Will be determined at runtime - ROWNUM : Insertion position - OPTIONAL FIELDS : id_type_situation;idt_situation;id_matricule;id_contrat Note : Webservice will automatically create, add and fill its own fields in target table Shows the entire URL Fuzible has built to send data to an API. Checks the URL and behavior of the call.
MAIL	A Query must start with the mail address(es) for which you want to send data to (addresses have to be separated by a ';'). ex : mymail@gmail.com;anymail@yahoo.frSELECT (...) Show me an exemple **fuzible@fuzible-app.com:select** id_sample,li_sample,dt_random_date,nb_random_number,li_random_string,id_group,id_ssgroup **from** sample1 ***as*** MyChart **WHERE** id_sample > ***{?1}*** **Fuzible Email -> fuzible@fuzible-app.com** Query Analyzer > Source Infos > Target Infos > Query Details DRIVER : Mailbox CONN. NAME : Fuzible Email MAIL SUBJECT : My Email RECIPIENT(S) : fuzible@fuzible-app.com CHART TITLE : MyChart Indicates: - The subject of the mail (retrieved from the job description) - Recipients (OUTPUT of the query) - The name of the data table (retrieved from the alias of the query: *SELECT * from matable AS My_Chart -> Affichera My Chart* *(Underscores are systematically replaced by a whitespace)*

AD	**users:select** user *as* name, param1 *as* description **from** user_parameters Fuzible Active Directory Demo -> users Query Analyzer > Source Infos > Target Infos > Query Details DRIVER : Active Directory CONN. NAME : Fuzible Active Directory Demo AD QUERY : (&(objectClass=user)(objectCategory=person)(name=[DATA FROM : name])) AD OBJECT TARGET : users Indicates: - The AD object in which data will be written in - The search query that will be performed

QUERY ANALYZER - View Data

This option opens a new window that will allow you several things:

- Load source data to preview it

- Have information on each source field

- Test the sync. mode

By clicking on "Load Source Data," the software will load the source data and display a 500-row preview (can be changed).

If the Job is a Sync. Job, each tab (Target, Insert, Update, Delete) will show you everything the sync. will do.

Show Source Data

Data Preview Data Successfully Loaded. With Data Analyzer

Preview (rows) : 500

Source Data Target Data To Insert To Update To Delete

DataTable Name : AXPROD
Namespace : A
Row Count : 2503
Fields Count : 8

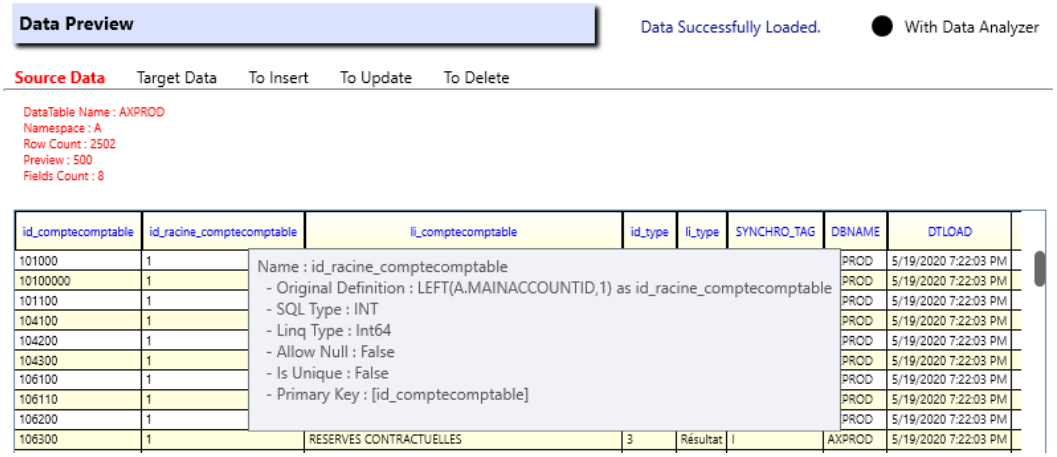
Additional Properties :
[AX_dim_plancomptable] SYNCHRO_PRIMARY_KEY : id_comptecomptable

id_comptecomptable	id_racine_comptecomptable	li_comptecomptable	id_type	li_type	SYNCHRO_TAG	DBNAME	DTLOAD
101000	1	CAPITAL APPELE	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
10100000	1	Capital souscrit (Sociétés de capitaux -	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
101100	1	CAPITAL SOUSCRIT NON APPELE	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
104100	1	PRIME D EMISSION	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
104200	1	PRIME DE FUSION	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
104300	1	PRIME D APPORT	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
106100	1	RESERVE LEGALE	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
106110	1	RESERVE LEGALE	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
106200	1	Réserves indisponibles.	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
106300	1	RESERVES CONTRACTUELLES	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
106400	1	Réserves réglementées	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
106480	1	RESERVE SPEC ART 238B CGI	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
106800	1	AUTRES RESERVES	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
109000	1	ACTIONNAIRE CAPITAL SOUSCRIT NON AP	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
110000	1	REPORT A NOUVEAU BENEFICE	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
119000	1	REPORT A NOUVEAU PERTE	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
120000	1	RESULTAT EXERCICE BENEFICE	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
121000	1	RESULTAT PROVISOIRE	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
129000	1	RESULTAT EXERCICE PERTE	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
129100	1	ACOMPTES SUR DIVIDENDES EN ATTENTE D'AFFECTATION	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
13100000	1	Réserve légale	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
13810000	1	Autres réserves disponibles	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
13821000	1	Réserve pour l'impôt sur la fortune	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
14110000	1	Résultats reportés en instance d'affectation	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
14120000	1	Résultats reportés (affectés)	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
145000	1	AMORTISSEMENTS DEROGATOIRES	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
151100	1	PROVISION POUR LITIGES	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
151110	1	LABOR TERMINATION COSTS	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
151120	1	SOC - Divers provisions litige	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
151130	1	SOC - Divers provisions litige	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM
151800	1	AUTRES PROVISIONS RISQUES	3	Résultat	I	AXPROD	5/22/2020 11:44:59 AM

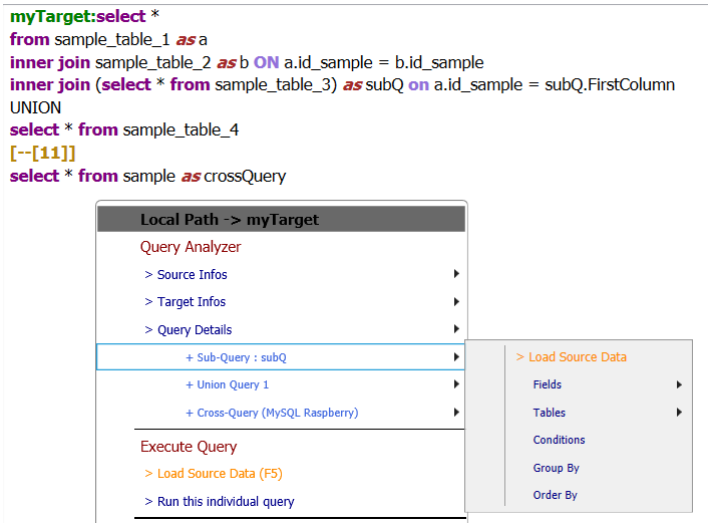
Export Content as CSV

It is possible to define the number of rows to be displayed in the preview window, but also to make a quick and simple export of data in CSV format (useful for making quick comparisons of data)

In addition, by clicking "With Data Analyzer", you can see the details of each field from the Source. Useful for understanding how Fuzible interprets data types.



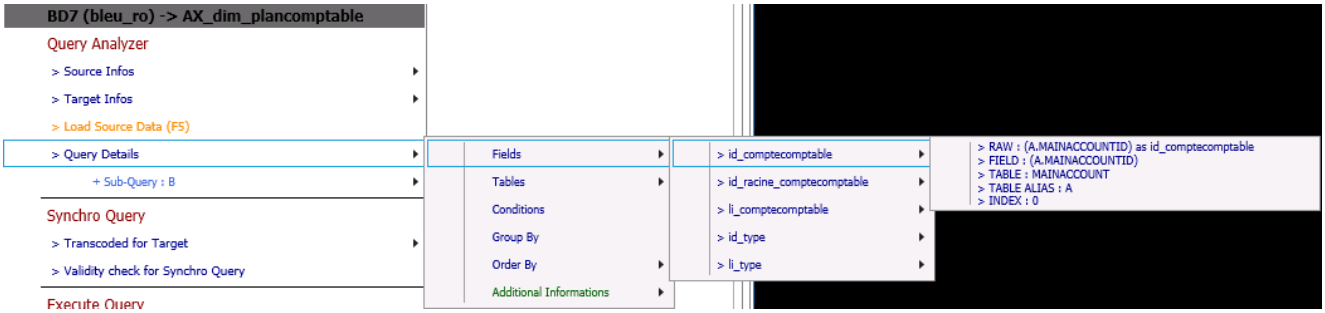
Note that each sub-query, each cross-query appear in the contextual menu and the data from each of them, loaded independently of the rest. In the example below, the Source query contains a sub-query, a UNION, and an cross-query.



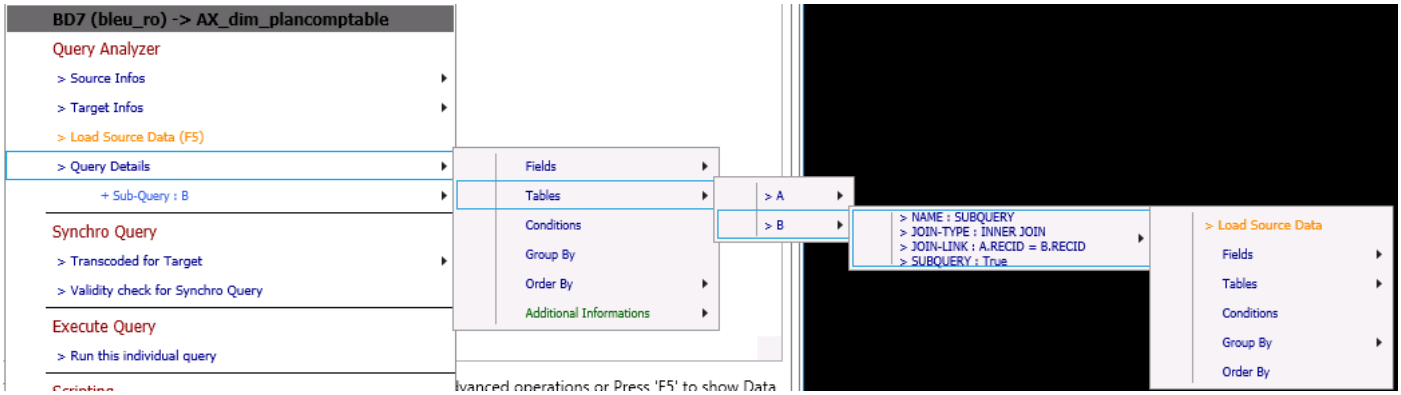
QUERY ANALYZER - Query Details

Allows you to fully deconstruct a query to verify that it is compliant, and that it has no syntax errors.

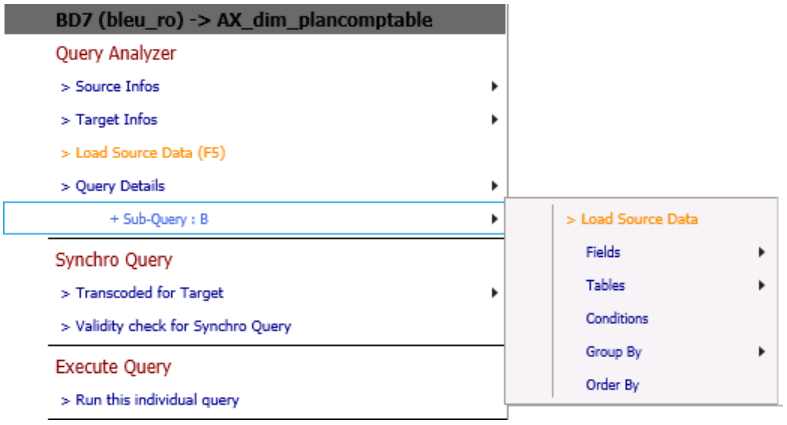
For example, you can go through all the fields...



All tables and understand their joins…

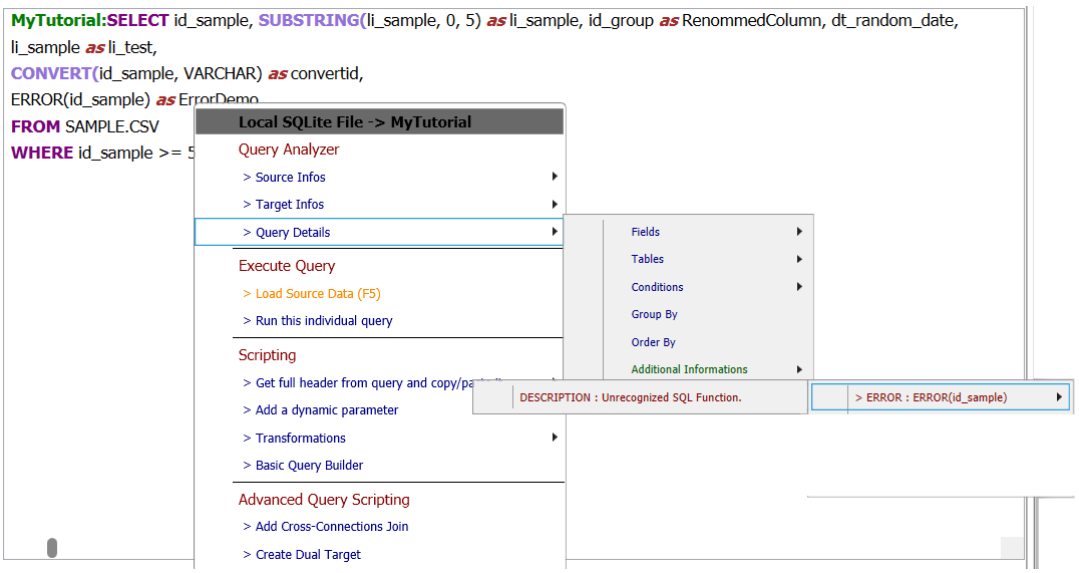


View sub-queries and try them…



Check syntax errors…

In this example, the query contains an unknown transformation: ERROR(id_sample)



The syntax errors detection is especially useful for "non-SGBD" queries, which allows you, if the query fails, to understand why it didn't work.

In the case of a query on a SGBD, the detection is essentially informative, as Fuzible does not know all the twisted cases that a query may contain.

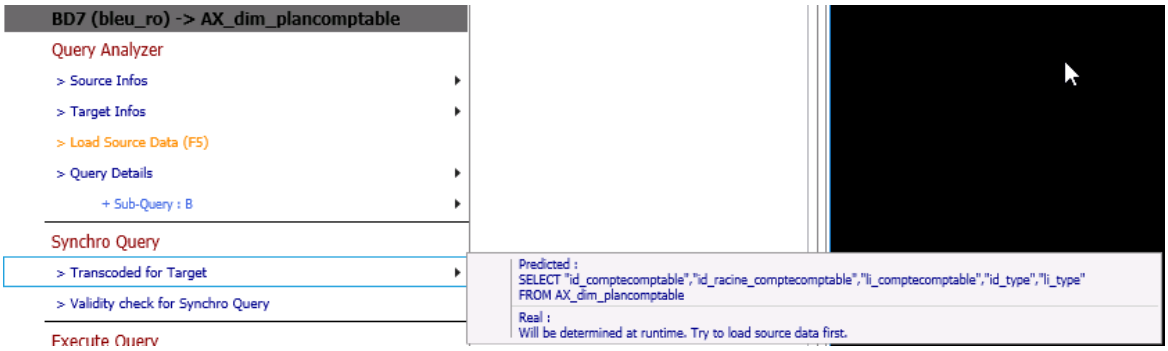
EXECUTE QUERY - Run this individual Query

You can only execute a specific query rather than the entire Job. By choosing this option, only the query on which you are positioned will be executed.

In this mode, the "LOG" tab does not fill up and the graphical interface is "blocked" for the time of execution. Once the processing is done, a LOG screen appears and shows the result.

SYNCHRO-QUERY - Transcoded for Target

When you have written a query for a synchro. job, you may want to test how well the Source query transcoding is working on the Target. This menu lets you see the query as it will be performed in the Target.

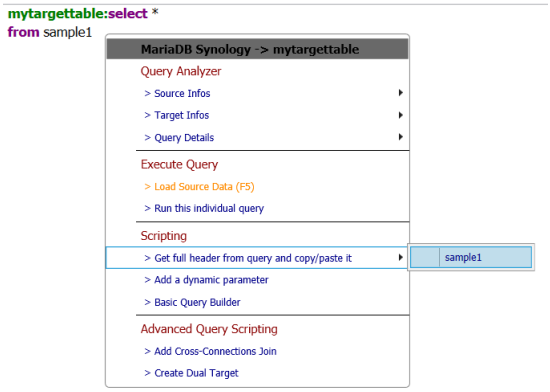


SYNCHRO-QUERY - Validity check for Synchro Query

This option simply checks the validity of the synchro. query.

SCRIPTING - Get Full Header Query and copy/paste it

It can be tedious to manually enter the entire header of an SQL table or file (if you want, for example, not to do a SELECT * but a SELECT with the name of the fields). By clicking here, Fuzible will retrieve all the fields from the Source and if it has joins, you can choose from which table/file/webservice... you want the header back:

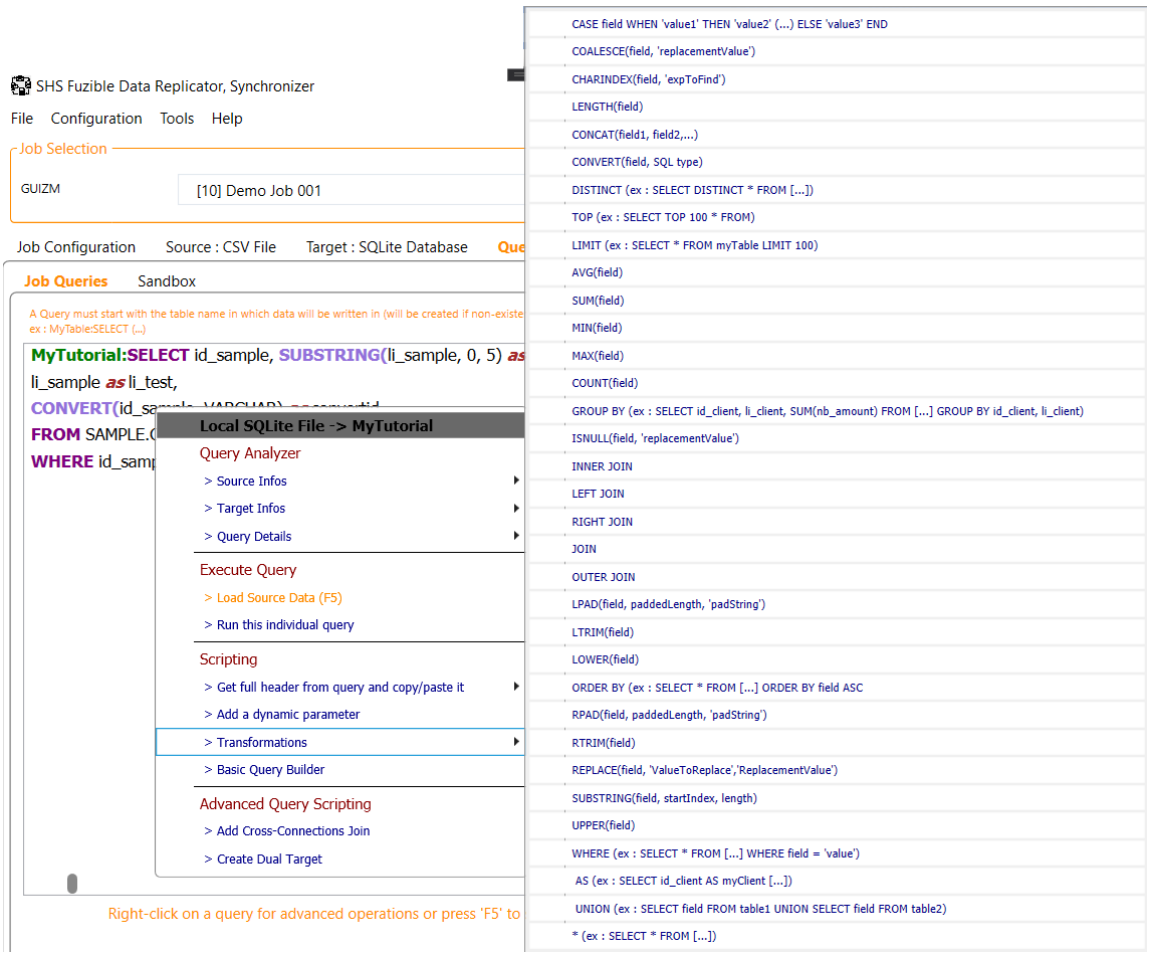


Once the header is retrieved, the software copies the header in the clipboard, you are then asked to paste it in your query:

```
mytargettable:select id_sample,li_sample,dt_random_date,nb_random_number,li_random_string,id_group,id_ssgroup
from sample1
```

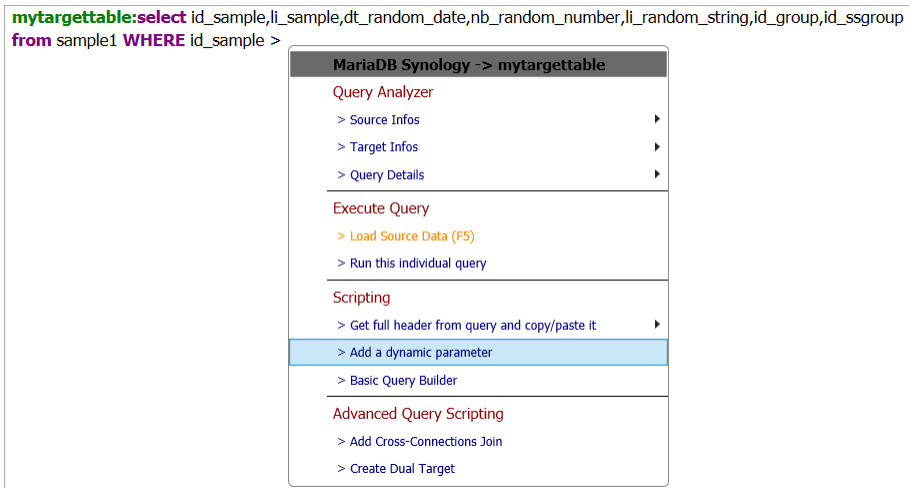
SCRIPTING - Transformations

In case your Source connection is not a database, the Transformations menu reminds you of all the SQL syntax available to manipulate the data. This is obviously also available during writing : the assistant makes suggestions based on what you type.

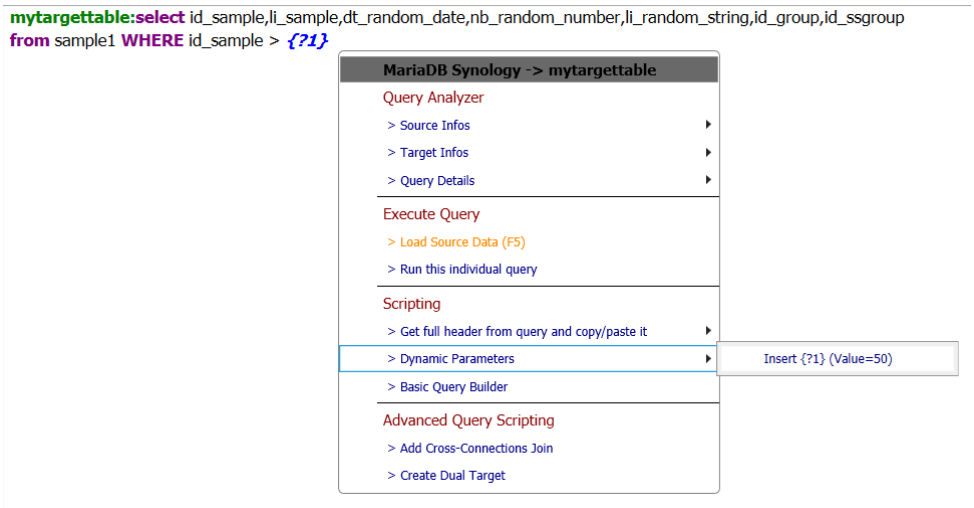


SCRIPTING - Add a Dynamic Parameter

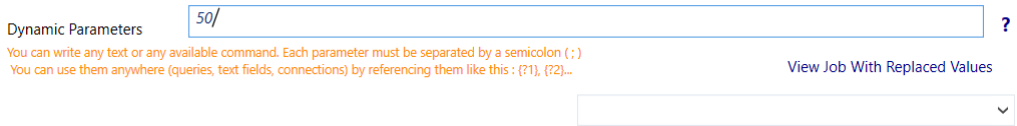
You can add a Dynamic Parameter to your query. For example, I want to make the "WHERE id_sample" filter dynamic :



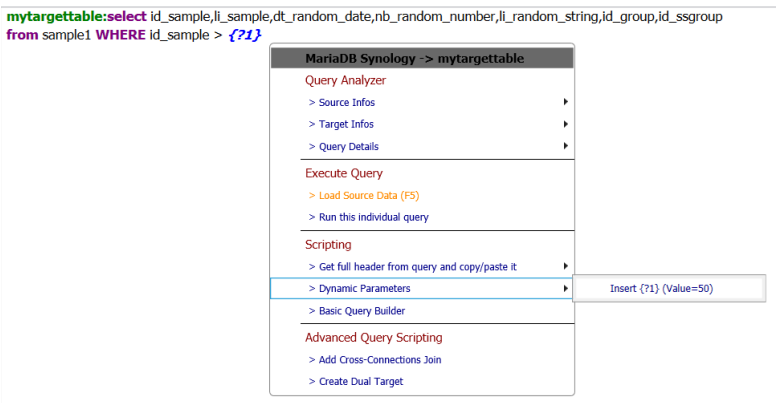
After entering the desired value, Fuzible will add the dynamic setting in the query:



... then add the parameter in the “Job Parameters” menu :



If Dynamic Parameters have already been set, the menu will be as follows:



SCRIPTING - Basic Query Builder

A simple assistant to create a query. It is understood that this mode does not allow for advanced queries.

ADVANCED QUERY SCRIPTING - Add Cross-Connections Join

This is where you can open the menu for a cross-query.

ADVANCED QUERY SCRIPTING - Create Dual Target

This is where you can open the menu for a multi-target query.

Log Viewer tab

Depending on the level of LOG chosen in the "Job Configuration" tab, you will see more or less detail in the Log Viewer.

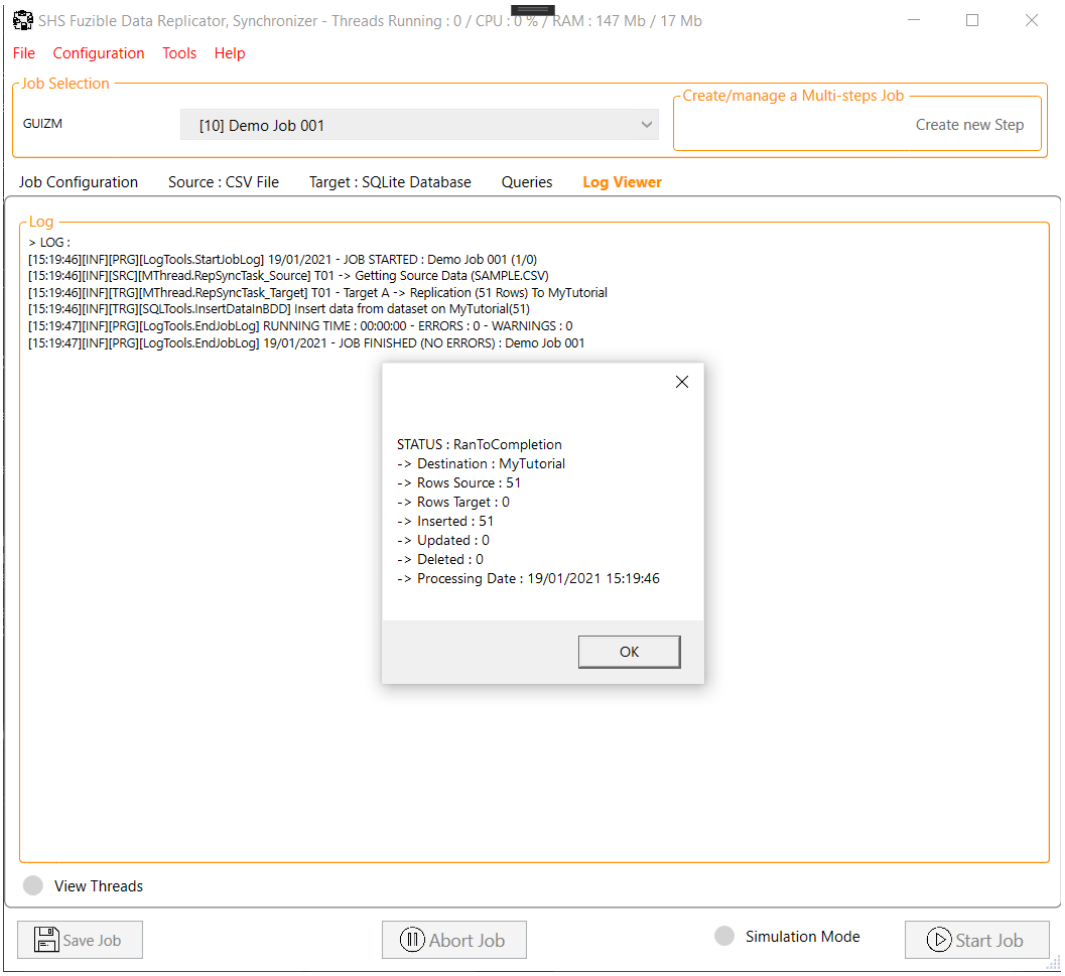
What you see is systematically referred to log files that are produced by the software.

At the end of a Job, a message will appear on the screen indicating its status.

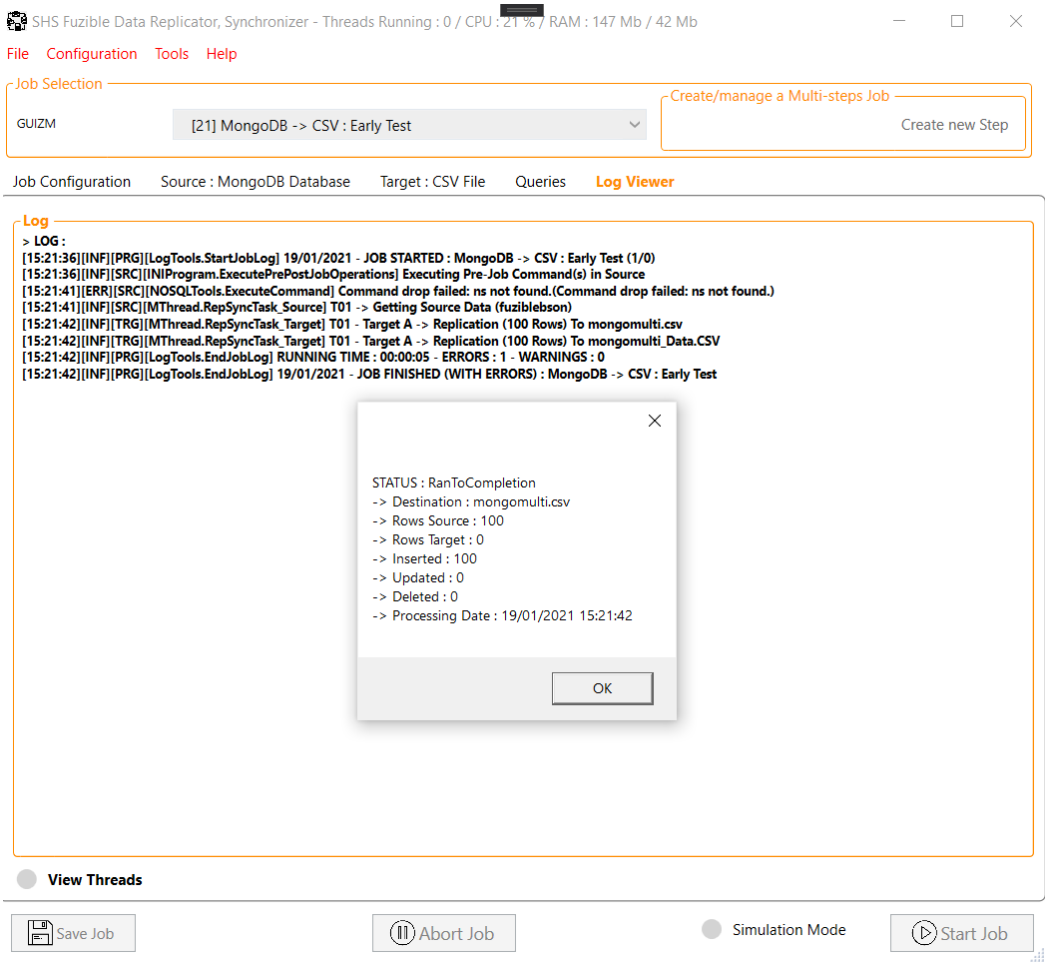
A round-up of debug possibilities:

- The Query Analyzer and the "Show Source data" screen (contextual query menu)
- The "SIMULATION" mode, which executes the Job without performing any operation in the Target, it merely displays everything it will do there (see below)
- The general LOG: quite verbose, it can help you understand a problem during Job execution, you can also set it up in "Detailed" to get as much information as possible.
  Check out the files produced by the Job, you have 3:
  1) The LOG file, which shows everything you see in the "LOG Viewer" tab
  2) The "QUERIES" file that shows all the queries that have failed.
  3) The "DEBUG" file that provides a higher level of information from any error.

Example of a successful execution:



Example of a failed execution; THE LOG is displayed in bold:



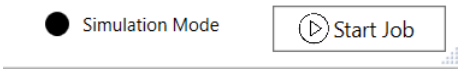
Running a Job

Once the Job is set up, saved, and tested, you can execute it and interrupt it if necessary.

These buttons are at the bottom of the interface. Also, you will find that on the "Job Configuration" tab, you have a little "Simulation Mode" button just above. This will write in the LOG any “write” operation that is going to be executed on the Target, without performing it.

This LOG has the advantage of not actually executing the Job, so not to take any risks and possibly compromise the Target.

In addition, this mode writes all SQL queries, which can be very useful in the case of replication of SGBD data to SGBD: allows you to get all SQL code: "INSERT," "DELETE," "UPDATE" statements.



Example of "Simulation" output (Job that copies data from a file to a database)

```
[SIMULATION MODE] WILL POST-WORK INPUT FILE(S) -RIEN : C:\Users\guizm\source\Workspaces\Fuzible\Fuzible\bin\x64\Debug\FILES\SAMPLE.CSV
[SIMULATION MODE] WILL POST-WORK INPUT FILE(S) ON (S)FTP - RIEN : SAMPLE.CSV
[SIMULATION MODE] SELECT COUNT(*) FROM sqlite_master WHERE type = 'table' AND tbl_name = 'MyTutorial';
[SIMULATION MODE] SELECT il.cid, il.name, il.name, il.type, '0', il."notnull", il.name, il.pk FROM sqlite_master AS m, pragma_table_info(m.name) AS il WHERE m.type='table' AND m.name = 'MyTutorial' AND pk =
[SIMULATION MODE] SELECT tbl.cid, tbl.name, il."from" || '_' || il."to" as name_constraint, 0, tbl."notnull", il.seq, il.seq, il."table", il."to" FROM sqlite_master AS m INNER JOIN pragma_foreign_key_list('
[SIMULATION MODE] SELECT il.name, il.type, '0', il."notnull", il."notnull", il.type, '0', il.dflt_value, 'NO' as UniqueKey, 'NO' as IsKey FROM sqlite_master AS m, pragma_table_info(m.name) AS il WHERE m.type
[SIMULATION MODE] DELETE FROM "MyTutorial";
[SIMULATION MODE] SELECT COUNT(*) FROM "MyTutorial";
[SIMULATION MODE] INSERT INTO "MyTutorial" ("id_sample","li_sample","RenommedColumn","li_test") VALUES (50,'five-','000','five-');INSERT INTO "MyTutorial" ("id_sample","li_sample","RenommedColumn","li_test")
[SIMULATION MODE] INSERT INTO "MyTutorial" ("id_sample","li_sample","RenommedColumn","li_test") VALUES (82,'eight','002','eight');INSERT INTO "MyTutorial" ("id_sample","li_sample","RenommedColumn","li_test")
```

Also, if you check the associated LOG "Queries" file, you will get all the queries in plain sight, including INSERT. This scenario is especially useful if you've installed the app locally, and you want to send data to a locally inaccessible database.

You can only generate queries through the "simulation" mode, and then connect to the remote server to integrate the data via INSERT code Fuzible produced.

"Service" Application

The software comes with a background service application: This service works in harmony with the “Client” application (using SQL mode), as well as the Job Orchestrator.

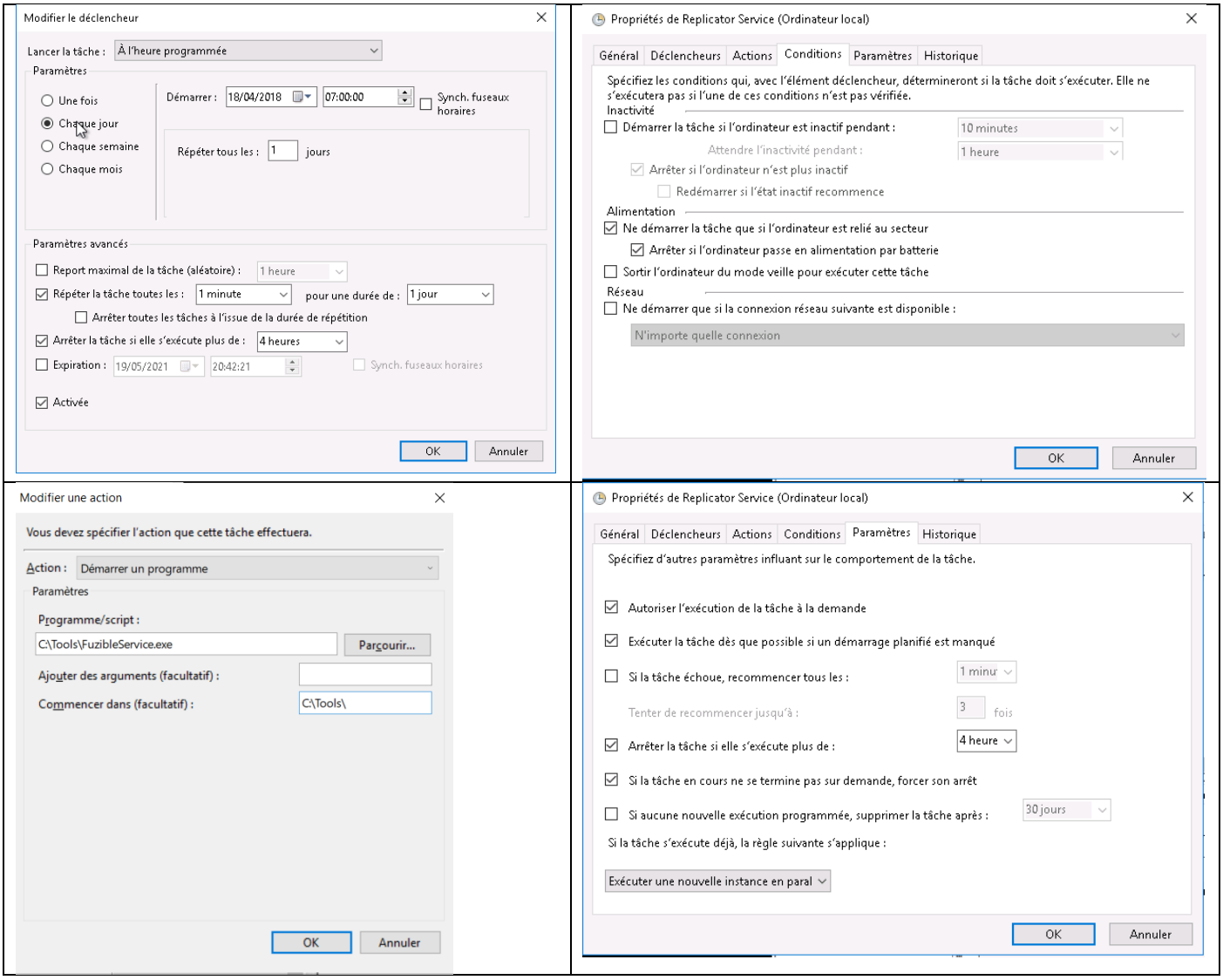
The app automatically creates and purges the SQL table that is used for its operation. As a "console" application, its LOG is written in the « SERVICE_YYYYMMDD_LOG.TXT » file.

The application keeps only one file, it systematically erases the one from the day before.

It retrieves the list of Jobs invoked (by the "Client" application or by the Planification) and executes them one after the other (it can run several in parallel, this setting being managed in the main application, configuration menu)

Setting up the Windows Task Manager

Note: This setting can be done automatically by Fuzible's configuration menu. However, you may need to manually edit/create it on somewhat tricky points, such as the execution account.



With each start, the app checks the stack of requested Jobs to run and:

- Checks if the number of Jobs being processed does not exceed the max. value from the settings.
- Sorts out the list of Jobs to be launched according to the priority assigned to it (between 1 and 3)
- Executes the requested Job(s)
- Follows the progress of the job and get its output ; updates the SQL table accordingly so that the user, from the “Client” application, can see the progress (by clicking "Job Status")

The service must be installed on the same machine and path as the Fuzible application.

Setting up an external job (excluding Fuzible) with the "Service" app

The idea of this option allows you to take advantage of the "client/service" system to perform any other task.

This is quite feasible, and simply requires you to manually enter data into the SQL table "client_jobs" (which is used by the "Service" application and which is located on its SQL instance) the external Job information that one wishes to be able to trigger (basically, the execution of a BAT file performing certain operations is well-advised)

Field	Description
User_jobs	Use any username (for example, the person who creates this job)
Job_id	A Job number, for example "001"
Job_name	Job name: this is what will be displayed to the user
Job_description	Description: A few more words to describe the Job
Job_params	Default Dynamic Parameters (optional)
Job_queries	You can maybe write a more in-depth description of the Job ?
Job_haschildren	0
Job_password	The password that allows the user to launch the Job. Fuzible passwords are encrypted, but for those external commands, you have to enter it "as it" in the table
Job_priority	Execution priority (1,2,3)
Application_name	The name of the app to be launched (ex : c:\Tools\MyFile.bat)
Job_category	Job categorization to optimize user view

For the user, this is the representation of an external Job in the list of Jobs that will be proposed to him in the “Client” application:

And here is the database representation:

	user_jobs	job_id	job_name	job_description	job_params	job_queries	job_haschildren	job_password	job_priority	application_name	job_category
1	CMD	001	Clôture Comete	Préparation à la cl...	202001	NULL	0	wolf	1	c:\Tools\omnis_comete_cloture.bat	External Batch
2	SVCSCO	[14]	BLEUSHARE -> BLEU : Véhicules : Imp...	Traitement des fic...	150320	IMPORT_LEASEPLAN_C...	0	3sdigA8JNkuO...	1	ReplicatorApp.exe	File -> Database
3	SVCSCO	[15]	BLEUSHARE -> BLEU : Véhicules : Imp...	Traitement des fic...	202004	IMPORT_LEASEPLAN:s...	0	y/9f5nSoe2Dy...	1	ReplicatorApp.exe	File -> Database
4	SVCSCO	[16]	BLEUSHARE -> BLEU : Véhicules : Imp...	Traitement des fic...	20200331	IMPORT_TOTAL_SERIS...	0	i+CKpDuXSFM...	1	ReplicatorApp.exe	File -> Database
5	SVCSCO	[53]	DWH -> FICHIER : Extraction du Grand ...	Extraction du Gra...	>=;%YYYY%MM<...	v_grand_livre_comptable....	0	lyMQTIDiKO+u...	1	ReplicatorApp.exe	Database -> File
6	SVCSCO	[62]	MOSAIC -> BLEUSHARE : Extraction d...	Réalise l'extractio...	20200515	INT_acomptes.CSV:selec...	0	57P6RHhtq8X...	1	ReplicatorApp.exe	Database -> File
7	SVCSCO	[63]	BLEUSHARE -> PANAMA : Import RUB...	Import de l'OD de ...	120520_130125_...	RUBPAI_RHPI_import:sel...	0	PAEzQGzkzVD...	1	ReplicatorApp.exe	File -> Database
8	SVCSCO	[65]	BLEUSHARE -> PANAMA : Import des ...	Import des heures ...	2020Avril_202005...	COMETE_heures:select *...	0	4WnbSiyvugdz...	1	ReplicatorApp.exe	File -> Database
9	SVCSCO	[72]	MOSAIC -> RHPI : Webservice Modifier...	Exécution des We...	id_matricule	webService=dossier.contr...	1	S5MO8Gtp0/G...	1	ReplicatorApp.exe	Database -> Webservice
10	SVCSCO	[75]	MOSAIC -> RHPI : Webservices Person...	Exécution des We...	modifier;ric;id_matr...	webService=dossier.{?2}....	0	HNq7kAZtyKn...	1	ReplicatorApp.exe	Database -> Webservice
11	SVCSCO	[81]	MOSAIC -> RHPI : Préparation aux modi...	Extractions à partir...		01_CONTRAT_MODIFIE...	0	O7JbyuFDrblq1...	1	ReplicatorApp.exe	Database -> File
12	SVCSCO	[85]	MOSAIC-PP -> RHPI : Webservice Prim...	Exécution du We...		program=SPS010RB:SEL...	0	pmx5BF4cXE6...	1	ReplicatorApp.exe	Database -> Webservice

"Client" Application

A lightweight client is provided and allows you to run jobs remotely from his own workstation.

The value of making the ability to trigger a Job by an user is crucial, for example:

- Let users choose the date/time to launch a Job

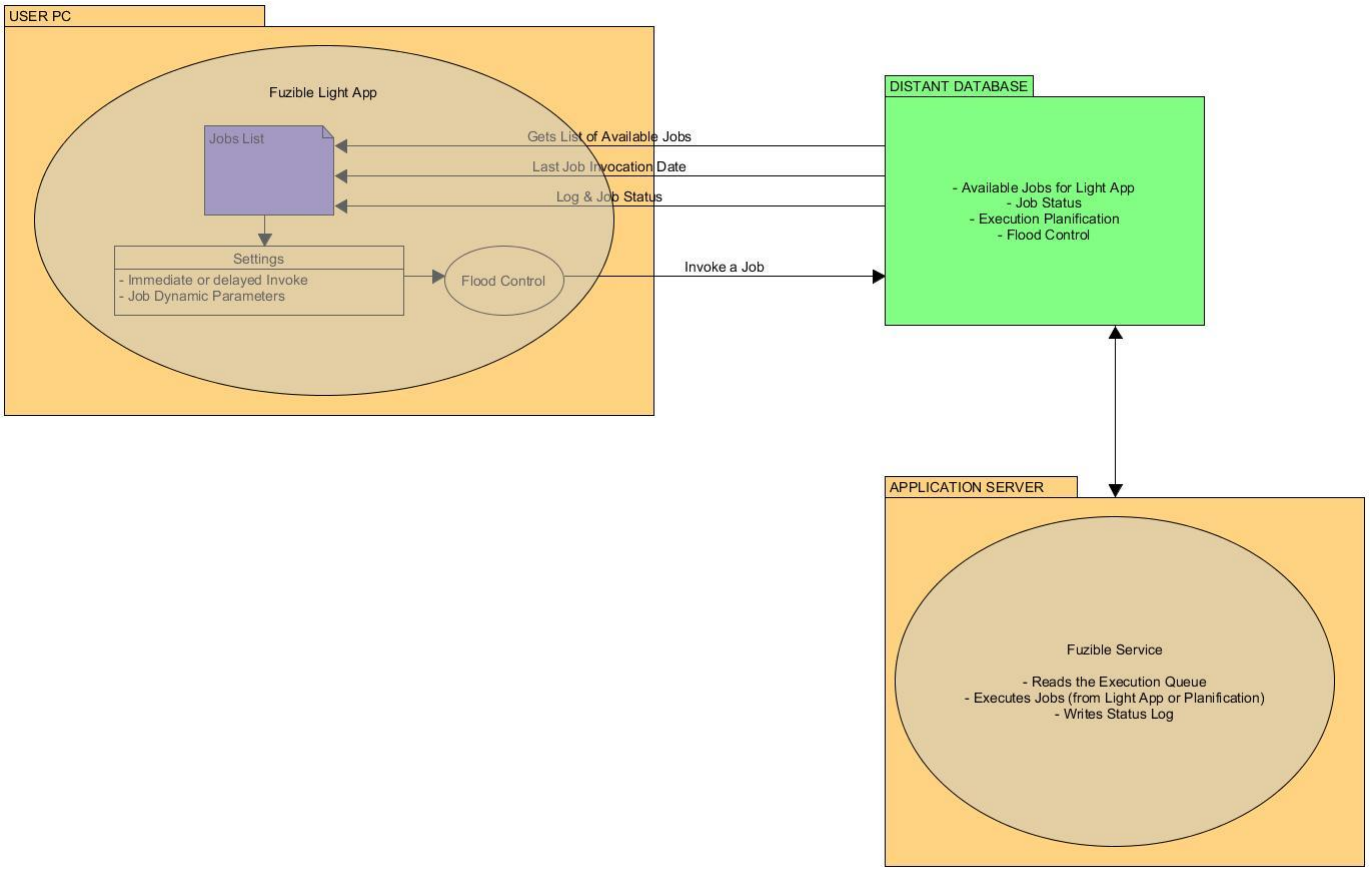
- Produce reports on the fly

Pre-requisite on client workstation:

- Windows OS (7+)
- .NET Core 3.1
- INI file "CLIENTAPP. INI" (available in Fuzible installation path)

Warning: If you change the Client/Service connection string in the app's general settings, you'll need to provide the new "CLIENTAPP. INI" to users because this file contains the connection settings!

Here's a diagram of how « Client » App works :

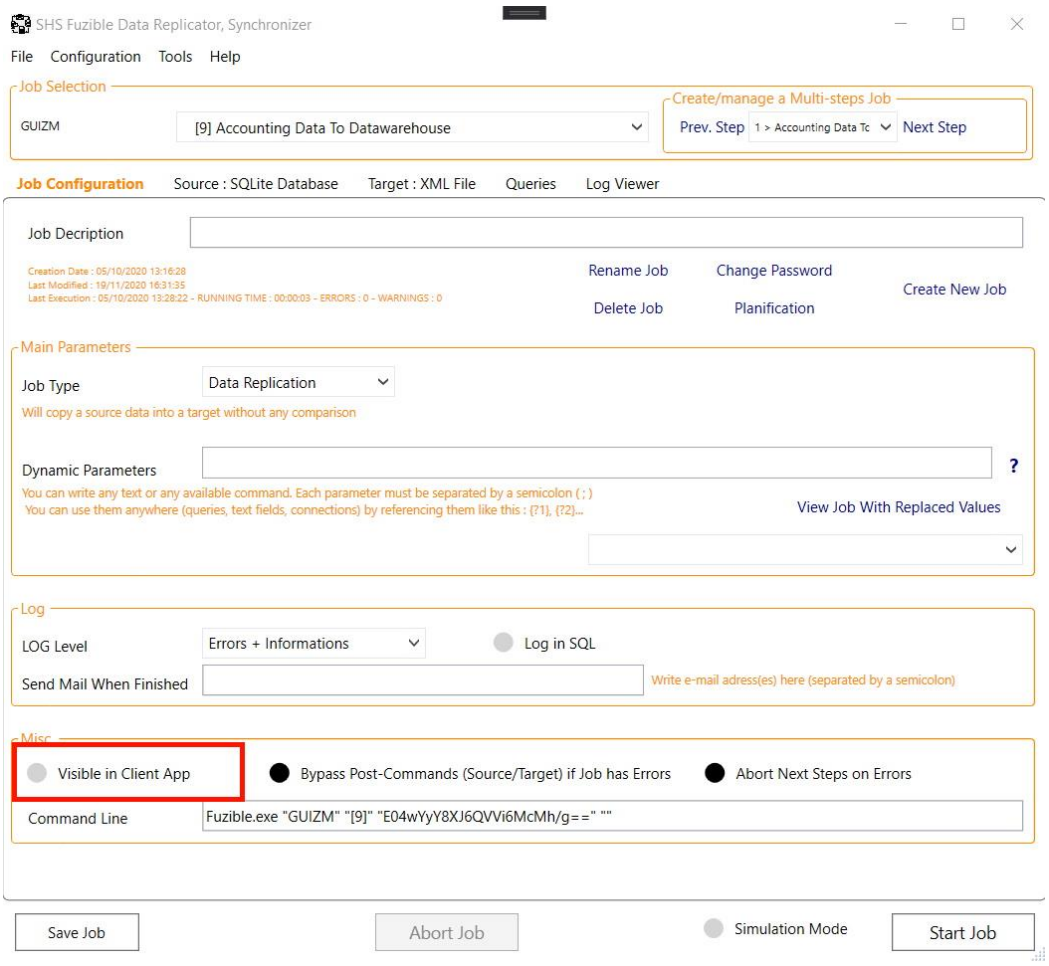


It communicates with the Service through a database that is configured from the Main Application.

The light client app uses a "stack" system. When choosing a job to execute, the app writes an SQL row in the "Service" app database instance. The settings of this connection are present in "CLIENTAPP. INI" and are encrypted for security reasons.

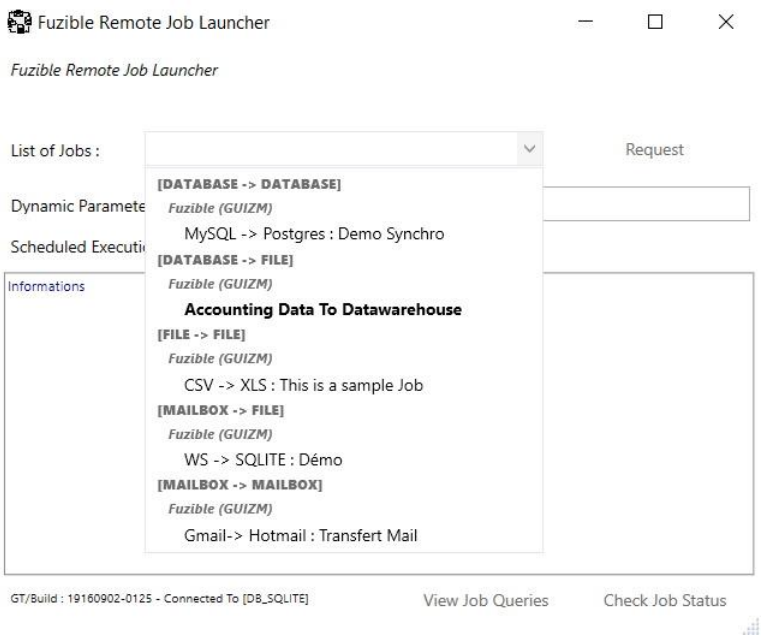
The “Client” software does not execute the Job. It loads it into a stack; it's the "Service" app provided that controls and launches the jobs that are invoked.

A Job must be configured beforehand as visible in the "Client" application:

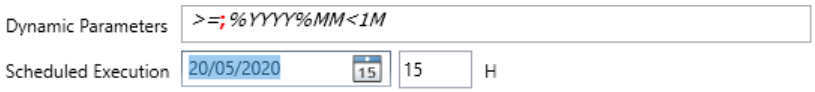


By opening the "Client"app, you can choose one of the Jobs available from the list. Its password will be required; The person who creates the Job will have had the presence of mind to provide it to the person (or persons) granted to launch the Job.

The list of available Jobs:



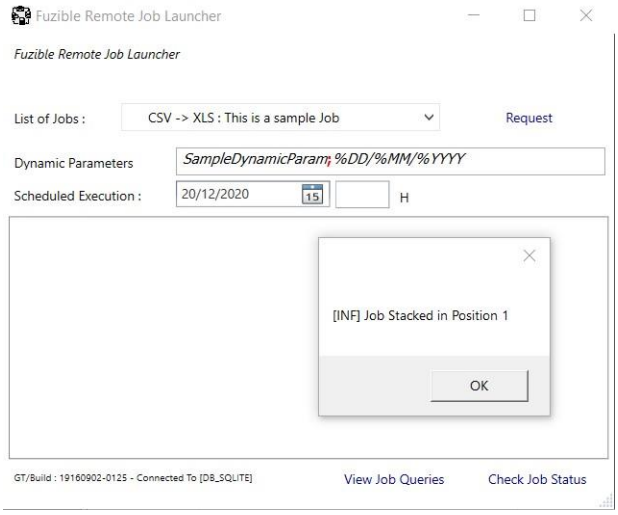
The person handling the client application can change the dynamic parameters of the selected Job: very useful to produce, for example, a period-specific Reporting, setting a filename to import, etc. everything is possible, it all depends on how the Job is configured!



It is also possible to delay the launch of the Job, by default, by clicking "Request", the Job will be stripped as quickly as possible by the "Service" application, but it can also be set to a particular date/time (for example to launch "heavy" tasks in the middle of the night without having to wake up in the middle of the night to trigger the launch)

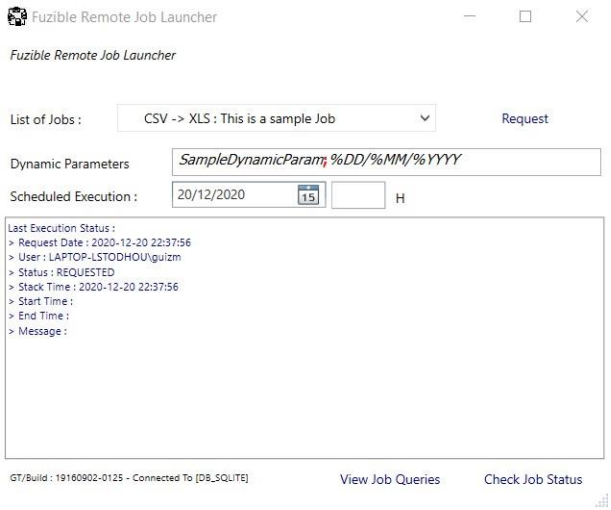
"View Queries" allows you to see the queries associated with the Job, and their translation with dynamic settings. Rather reserved for users with some knowledge of SQL!

After clicking "Request", the software shows the position in the stack:

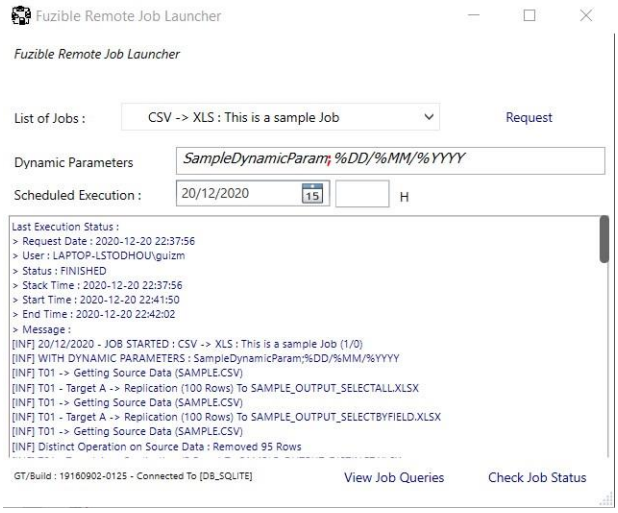


The user can follow the status by clicking on "Check Job Status."

The Job has been invoked but not yet handled by the "Service" application:



Job executed and completed:



Note: If the user clicks "Check Job Status" before requesting the execution, he will see the LOG of the last run to date, if it has not yet been purged (the retention time is defined in the general parameters of the software)

Fuzible SQL: Glossary

Supported functions

Here is a list of the SQL functions supported by Fuzible's engine for querying a non-SQL Source. For use, internet is your friend (SQL standard). The Query Assistant will also show you how to use them.

TRANSFORMATION	
SUBSTRING	Extraire un morceau de chaîne dans une chaîne
CONCAT	Concaténer des champs ou des caractères
CASE field WHEN ... THEN ... ELSE ... END	Piloter une valeur en fonction d'une autre
CONVERT	Force la conversion d'un type de données pour un autre
LTRIM, RTRIM	Effacement des espaces avant/après une chaîne
ISNULL, COALESCE	Remplacer une valeur vide par autre chose
LPAD, RPAD	Compléter une valeur par une chaîne à droite ou gauche
LENGTH	Longueur d'une chaîne
CHARINDEX	Position d'une chaîne dans une chaîne
LOWER, UPPER	Mettre une valeur en majuscules ou minuscules
REPLACE	Remplacer une valeur par une autre
ANONYMIZE	Randomizes values to simulate an « anonymization » feature
AGGREGATION	
SUM	Somme d'un ensemble
MAX, MIN	Maximum ou minimum d'un ensemble
AVG	Moyenne d'un ensemble
COUNT	Quantité d'un ensemble
FONCTIONS ESSENTIELLES	
SELECT DISTINCT	Supprimer les doublons d'un résultat
SELECT TABLE x	Propre au SQL de Replicator : permet de définir la table sur laquelle on aliase les champs (rappel : un webservice peut par exemple renvoyer plusieurs tables)
SELECT TABLE x ONLY	Propre au SQL de Replicator : permet de ne renvoyer qu'une table choisie dans un ensemble (cas des webservices par exemple)
LIMIT, TOP	Limite les résultats retournés **Ex** : SELECT TOP 100 * FROM monfichier **Ex 2** : SELECT * FROM monfichier.csv LIMIT 100
JOIN (LEFT, OUTER, INNER, RIGHT)	Jointures entre sources
WHERE	Filtres (= < > != IN NOT IN) ainsi que les « SELECT » imbriqués **Ex :** WHERE LENGTH(li_test) > 0 **Ex 2 :** WHERE li_test in (SELECT li_data FROM matable)
ORDER BY	Organisation des résultats
GROUP BY	Regroupements d'aggrégations
UNION	Merge de plusieurs résultats aux schémas identiques
FONCTIONS AVANCEES	
Math. Operations in functions	Additions et soustractions aux fonctions traitant de nombres entiers (charindex, length, substring) **Ex :** SELECT SUBSTRING(li_test, CHARINDEX(li_test, "-") + 1, 10) FROM monfichier.csv
Sub-queries	**Ex :** SELECT * FROM (select * FROM monfichier.csv) as subQ **Ex 2 :** SELECT * FROM monfichier.csv WHERE id_test NOT IN (SELECT id FROM monautrefichier.csv)

Unsupported

HAVING	Filtrer des fonctions d'aggrégation
« null »	Le « NULL » au sens d'une base de données n'est pas compris **Ex :** CASE WHEN x IS NULL THEN doit être saisi ainsi : CASE WHEN x = '' THEN
« GETDATE » or « CURRENT_TIMESTAMP »	De manière plus générale, la saisie dynamique du datetime actuel. En revanche vous pouvez utiliser les paramètres dynamiques du Job pour contourner cette limitation **Ex :** SELECT * FROM monfichier WHERE annee > {%YYYY}
Fields as sub-queries	**Ex :** SELECT (select id FROM monfichier.csv) as id FROM monautrefichier.csv
Math. operations on aggregated data	**Ex** : SELECT COUNT(*) + 10 FROM monfichier