UNIT 4 BUSINESS SOFTWARE

EMBRACING CHANGE

Business World UK Product

# Server Process Toolkit for Interfaces

## Configuration Guide

**Version 3.4**

CONTENTS

# 1 Introduction

This document provides a description of the various methods available in the Server Process Toolkit for Interfaces (ITK) and on how to configure those methods on a UNIT4 Business World (Business World) system.

This manual is for ITK version 3.4 or better (ITKv3). ITKv3 has been developed in C#[1] and ACT[2] for use on Business World Milestone 4 or higher. ITKv3 is currently a work in progress because a small number of the ITKv2 functions have yet to be developed as ITKv3 methods. These methods are greyed out in this document as follows:

**Method that has not yet been implemented.**

See section 1.2 for more details on the differences between ITKv3 and ITKv2.

How to install ITK is detailed in a separate Installation Guide.

**WARNING**

If you are still using ITK version 2 (ITKv2) which was developed in VB6[3] and AgrComSrv[4], then you must use the version of the Configuration Guide that accompanied that version of the software.

## 1.1 Overview

Business World has an extensive set of data import and data export facilities which can cater for many of a customer's needs without any programming being required. These facilities include:

- Standard import routines with 'query' facilities (CS15, GL07, LG04, etc.).
- ASQL[5] queries used to import data (COPY IN, COPY FROM).
- ASQL queries used to export data (COPY TO).
- ARW[6] reports to export data in more complex formats.

Eight shortcomings of the above are:

- The ASQL COPY commands can only handle two types of file:
  - One fixed length record per line where each field is in the same position on each record and there are no separators between fields,
  - One variable length record per line with each field separated by a delimiting character, usually a 'Tab' (ASCII[7] code 9), but can also be a "@", "£", "$", "%", "&", "." Or ";" character.
- When an interface uses a non-standard import file that file is not automatically deleted at the end of the run, so it is possible for a user to process the same file twice.

[1] *A Microsoft developed .NET programming language, pronounced "See-sharp".*
[2] *Agresso Customisation Tool: a library used to interact with any object in recent* Business World *versions.*
[3] *Microsoft Visual Basic 6.*
[4] *Agresso COM Server: a library used to interact with server processes in older* Business World *versions.*
[5] *Agresso Structured Query Language.*
[6] *Agresso Report Writer.*
[7] *See section 5.2.*

- The update of existing rows in the “Master file” import routines CS15 and PR43 does not allow you to blank out columns or remove addresses or relation values (where available).
- ASQL’s “COPY TO” command will fail if the file it is trying to write to already exists.
- Whilst an ARW report can create extremely complex output files, including CSV and XML the file is usually placed in the ‘Report Results’ folder with a name indistinguishable from that of any normal report. There is an ARW command that overrides the default name of an output file but this is not very flexible.
- An ARW can only create one output file and many standard server processes do not give you the ability to run additional reports or run a query.
- If an ARW is exporting suppliers (say) as XML and the receiving systems schema can only accept one supplier per file then ARW’s ability to only creating one file is a problem.
- It is not possible to access a process’s temporary tables (known as “help-tables”) from an embedded AG16 query because the names of those tables vary on each run of the process.

ITK addresses these issues by providing the following methods:

- A start-of-process method that will convert “Character Separated Value” (CSV) files to Tab Separated Value (TSV) files that can then be read by ASQL’s “COPY IN” command in the process’ query.
- A start-of-process method that will convert worksheets in Excel workbooks to Tab Separated Value (TSV) files that can then be read by ASQL’s “COPY IN” command in the process’ query.
- A start-of-process method that will convert “Counted Fields per Record” (CFR) files to Tab Separated Value (TSV) files that can then be read by ASQL’s “COPY IN” command in the process’ query.
  See the introduction of ‘CFR Import File Conversion (ConvCfr)’ for a description of this file format.
- A start-of-process method that will convert “Form-feed Separated Record” (FSR) files to Tab Separated Value (TSV) files that can then be read by ASQL’s “COPY IN” command in the process’ query.
  See the introduction of ‘FSR Import File Conversion (ConvFsr)’ for a description of this file format.
- A start-of-process method that will delete export files.
- An end-of-process method that will delete (or move to the “scratch” folder) non-standard import files
- An end-of-process method that will blank columns that have been changed to a particular value and also remove addresses and/or relation values that have been set to the same value.

- An end-of-process method that copies or moves output files created by an ARW (for instance) out of the Report Results folder and give them a new name.
- An end-of-process method that runs additional reports.
- A start-of-process method that runs an AG16 query.
- An end-of-process method that runs an AG16 query.
- **A start-of-process method that calls a command-line tool with parameters. The command-line tool can be a script (bat, cmd, vbs, etc) or a program (in-house, UNIT4 or third party).**
- **An end-of-process method that calls a command-line tool with parameters. The command-line tool can be a script (bat, cmd, vbs, etc) or a program (in-house, UNIT4 or third party).**
- A number of methods can split XML output into multiple files at a defined element and also make the payload "XML safe".
- A mid-process method that can capture the name of a help-table as it is created and put it in a process parameter that can then be accessed by an additional report or an end-of-process AG16 query.

## 1.2 Differences between ITKv3 and ITKv2

ITKv3 is different to ITKv2 in a number of ways:

1. As mentioned in the introduction ITKv3 is a work in progress because two (at version 3.4.0) of the ITKv2 functions have not been implemented as ITKv3 methods yet.
2. The old 'load_bi', 'load_rep' and 'load_be' process parameters were Business World's mechanism for calling VB6 server exits and:
    a. Therefore these parameters are not used in ITKv3 and **MUST** be removed from any report variants or user defined reports that previously used ITKv2.
    b. Are replaced with new process parameters: 'itk_start', 'itk_rep' and 'itk_stop' respectively. Note that in the current version of ITKv3 there no methods that can be called using the 'itk_rep' parameter.
3. The rewrite from VB6/AgrComSrv to C#/ACT has resulted in a number of benefits:
    a. You are no longer restricted to running a process on a 32bit server queue in order to use ITK.
    b. There is no registration of DLLs on the Business World Business-Server included in the installation of ITKv3, and so the entire installation, apart from the optional "INI file" for 'FldBlnk', can be carried out from within the Desktop client.
    c. It is now possible to link methods to those points in a process where help-tables are created. Methods that use these points are defined using the 'itk_crtab' process parameter.
    d. Because ITKv3 is no longer reliant on the "special" 'load_*xx*' parameters (see point 2 above) it has been possible to:
        i. Validate many of each method's parameters at the beginning of the process rather at the point when they carry out their action.

ii. Allow multiple values of the 'itk_*yyyy*' parameters in the same variant, see section 2.1 for more details.

4. A new 'Excel Workbook Conversion (ConvXls)' method has been developed.
5. The 'FSR Import File Conversion (ConvFsr)' method now accepts 'num_headers' and 'num_footers' parameters.
6. The 'Blank Columns and Remove Addresses and Relation Values (FldBlnk)' method:
   a. No longer has the complexity of having two different ways of being set up and so is consistent across both supported processes: CS15 and PR43.
   b. The processing has been improved by looking for changes since the job started rather than since a fixed number of minutes ago.
   c. **At this release support for DataLoad has not been included.**
7. In ITKv2 only the 'Copy or Move Output Files as a Process Ends (Export)' method would replace file name tags (see section 4.3). In ITKv3 this feature has been extended to include:
   a. The 'Remove Output Files as a Process Starts (DelFile)' method,
   b. The 'Run a Query as a Process Starts (BiQuery)' method,
   c. The 'Run a Query as a Process Ends (BeQuery)' method.
8. There is a new "<i>" file name tag that is replaced by the value of the 'AGRESSO_IMPORT' environment variable.
9. **The 'Run a Query as a Process Ends (BeQuery)' method does not have ITKv2's 'be_xparam_*n*' parameters.** These are no longer needed because the 'BeQuery' method automatically makes **ALL** of the process's parameters available to the query.
10. A new "Save Help-Table Name (TabName) method has been developed.

## 1.3 Differences in a UNIT4 SaaS Environment

If your Business World system is running in a UNIT4 SaaS[8] environment that environment has some restrictions that require certain ITK functions to either operate differently or not be available.

The running of AG16 queries is generally not allowed[9] and so methods 'BiQuery' and 'BeQuery' are not available.

Access to the Business World Business-Server's file system is restricted to the following folders:

| Folders | Access Level | Environment Variable |
|---|---|---|
| Customized Reports | Read | AGRESSO_CUSTOM |
| Data Export | Read/Write | AGRESSO_EXPORT |
| Data Import | Read/Write | AGRESSO_IMPORT |
| Report Results | Read | AGRESSO_PRINT |
| Server Logging | Read | AGRESSO_LOG |

This results in the following restrictions:

- The 'Conv*xxx*' methods can only delete import files not move them to "scratch".

[8] Software as a Service.

[9] If you have migrated to SaaS from a traditionally hosted Business World system you may be allowed to continue using queries that were developed prior to that migration, but you will not normally be allowed to develop new queries.

- Method ‘MovFile’ is not available because there is no access to the ‘AGRESSO_SCRATCH’ folder or the Windows\Temp folder.
- Method ‘FldBlnk’ cannot use an “INI File” because it has no access to the AGRESSO_EXE Bin folder, so the new “use system parameters” option must be used.
- In method ‘Export’ the file being exported can only be written to the ‘Data Export’ folder or a sub-folder of the ‘Data Export’ folder.

# 2 Configuration

In order to make use of an ITK method on a standard server process you must create a variant of that process that has some additional process parameters. In order to make use of an ITK method on a user defined report you must either add these additional process parameters to its definition or to a variant of that process using the '**Tools | Create report template**' command. In general these parameters vary from method to method, but they all have one thing in common:

- If you wish to use a start-of-process method then you must create a parameter with its 'ParamID' set to "itk_start" and its 'Default value' must be the name of the method you wish to use.
- If you wish to use an on-help-table-created method then you must create a parameter with its 'ParamID' set to "itk_crtab" and its 'Default value' must be the name of the method you wish to use.
- If you wish to use an end-of-process method then you must create a parameter with its 'ParamID' set to "itk_stop" and its 'Default value' must be the name of the method you wish to use.

As you can see from careful inspection of the screenshots in this manual the 'Param name' that you enter when creating an 'itk_...' parameter is purely for you own convenience, ITK does not use this value at all.

A server process variant or user defined report can have 'itk_start', 'itk_crtab' and 'itk_stop' parameters or any combination of the three.

**WARNING**

If you are modifying a report variant or user defined report to use ITKv3 instead of ITKv2 then you **must** remove the existing 'load_bi', 'load_rep' and 'load_be' process parameters from that variant and replace:

- The 'load_bi' parameter with the corresponding 'itk_start',
- The 'load_be' parameter with the corresponding 'itk_stop'.

Please note that there are no 'itk_rep' ('load_rep' in ITKv2) methods in the current release of ITKv3.

**WARNING**

Due to a restriction of ACT, ITK methods can only be used on server processes, therefore if you wish to use any of its methods on a simple report then you must convert that report into a server process. See section 4.1 for details of how to do this.

## 2.1 Multiple ITK… Parameters

As mentioned above an "itk_start", 'itk_crtab' or "itk_stop" parameter must be used to run an ITKv3 method. A significant enhancement to this is that it is now possible to run multiple methods on the same event. This has only been explicitly implemented for some of the ITK methods, but the intention is to make this possible for as many of the existing methods as possible in future releases. At the current release the only methods that can be called more than once in the same process are:

| Method | Description |
|---|---|
| BiQuery | Run a query as a process starts |
| BeQuery | Run a query as a process ends |
| Extra | Run an additional report as a process ends |
| FldBlnk | Blank columns and remove addresses and relation values |
| TabName | Save a help-table's name as it is created |

For the first ITK method on an event the corresponding 'itk_*xxxx*' parameter should be used (where "*xxxx*" is either "crtab", "start" or "stop"), for the second method a parameter 'itk_*xxxx*_2' should be used, for the third method a parameter 'itk_*xxxx*_3', and so on up to a maximum of 20.

When one of the methods in the above table is defined by an "itk…" parameter with a numeric suffix then all of its additional parameters must also have the same numeric suffix.

**WARNING**

The parameters of the 'Export' method also have numeric suffices but 'Export' can only be called once in a process. The number used in the ID of 'Export's parameters is used to identify which report output file(s) should be renamed and/or moved and must not be confused with the mechanism for allowing multiple 'itk_...' parameters to be defined (see section 3.8 for an explanation).

It is, however, entirely safe to call 'Export' from an 'itk_stop' parameter with a numeric suffix, but the numeric suffix of its additional parameters will not be the same (except by coincidence) as the numeric suffix of the 'itk_stop' parameter.

For example if you wish to run two extra reports on a process and then export one of the report files then you might define the following parameters:

| ParamID | Param name | Default value |
|---|---|---|
| itk_stop | First On Stop action | Extra |
| x_rep | Second report to run | MYREP1 |
| itk_stop_2 | Second On Stop action | Extra |
| x_rep_2 | Third report to run | MYREP2 |
| x_test_2 | Report help-table | helptab0 |
| itk_stop_3 | Third On Stop action | Export |
| move_1 | Move the second file to | c:\temp\... |

# 3 The Methods

## 3.1 CSV Import File Conversion (ConvCsv)

### 3.1.1 Introduction

For many years there has been a de-facto standard for data transfer files known as character separated value (CSV). Traditionally the character used to separate the values was a comma and in some people's minds CSV still stands for "comma separated value", but other characters such as the pipe "|" and tilde "~" are not uncommon.

Unfortunately only a very limited selection of separator-characters are supported by ASQL's "COPY IN" command and a comma is not one of them. This omission has been a minor implementation issue on a number of occasions. See section 5.1 for an overview of using queries to import data.

Other complications with the processing of CSV files are:

- The first record in the file may contain the names of the fields rather than data to be imported.
- If the value of a field contains the separator-character then that field's value will be enclosed in either apostrophes or speech marks, in some files this may be done only when necessary but in others the values of all "text" fields are "enclosed" regardless of their content.
- If a value that is enclosed in apostrophes or speech marks contains any of those characters then these will be "doubled up".

All of the above complications can be seen in the following:

```
EmpNo,Forenames,Surname,DoB
12,'Thomas, Patrick','O''Connor',31/10/1939
```

So the first record of the file contains the names of the fields and should not be treated as data to be imported and the values of the first data record are:

| | |
|---|---|
| **EmpNo** | 12 |
| **Forenames** | Thomas, Patrick |
| **Surname** | O'Connor |
| **DoB** | 31/10/1939 |

'ConvCsv' will take one or more existing CSV files in the folder defined by the "AGRESSO_IMPORT" environment variable and translate them into new files that have tab separated values (TSV), which can be read by the "COPY IN" command, the field values will have enclosing apostrophes or speech marks removed along with any doubled apostrophes or speech marks within them. It then changes the value of the "file_name" parameters so that the SQL query can process the TSV versions of the CSV files and finally it removes the original CSV files from the "AGRESSO_IMPORT" folder:

- In a SaaS environment they are simply deleted.
- In other environments they are moved to the folder defined by the "AGRESSO_SCRATCH" environment variable, or the folder defined by the 'TEMP' Windows environment variable if AGRESSO_SCRATCH is not set.

The new TSV files will have a “tsv” extension and will be present in the AGRESSO_IMPORT folder.

If a file has header records and/or footer records then an additional field will be inserted at the beginning of each record of the TSV file, this field will be of the form:

“*Xn*”

Where *X* is

- “H” on a header record,
- “D” on a detail record
- “F” on a footer record

And *n* is a sequence number which starts from one for each record type. For example:

- “H1” is the first header record,
- “D1” is the first detail record,
- “D235” is the two hundredth and thirty fifth detail record
- “F1” is the first footer record

If a file has neither header records nor footer records then this extra field is not created.

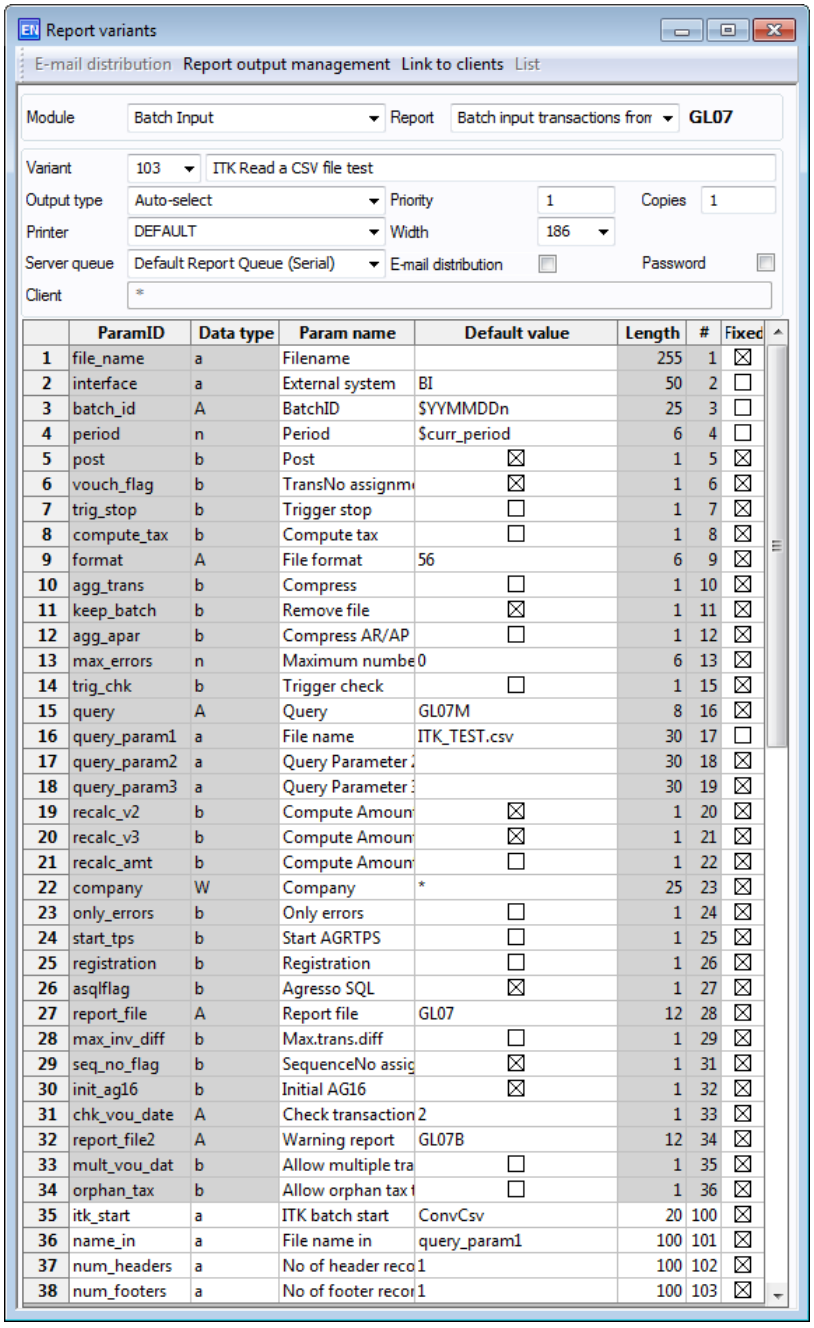
**WARNING**

The contents of a header or footer record will be put into the first data field of the output record with an appropriate number of trailing tab characters. So when you use the “COPY IN” command it is important that the table column you are copying the first data field to is large enough to hold any possible header or footer data. For example in the query referred to in section 3.1.3 the ‘account’ column is defined as “SPACE(25)” so that it is large enough to hold the date from the header (10 characters would have been enough for this) or the total debit amount from the footer (the line amount is 15 characters so 18 or 20 should have been enough but you can’t be too careful!).

Similarly the table column that the additional field that the program generates is going to be copied into (‘rec_type’ in the query below) must be big enough to hold at least the expected number of detail records, if you could have a million detail records then the column must be at least eight characters long to hold “D1000000”.

### 3.1.2 Set Up

In order to process CSV files in a data import routine you must create a variant with some of its standard parameters changed and some new parameters added. The example is for GL07 but the same principle applies to any other data import routine.




| | ParamID | Data type | Param name | Default value | Length | # | Fixed |
|---|---|---|---|---|---|---|---|
| 1 | file_name | a | Filename | | 255 | 1 | ☒ |
| 2 | interface | a | External system | BI | 50 | 2 | ☐ |
| 3 | batch_id | A | BatchID | $YYMMDDn | 25 | 3 | ☐ |
| 4 | period | n | Period | $curr_period | 6 | 4 | ☐ |
| 5 | post | b | Post | ☒ | 1 | 5 | ☒ |
| 6 | vouch_flag | b | TransNo assignm | ☒ | 1 | 6 | ☒ |
| 7 | trig_stop | b | Trigger stop | ☐ | 1 | 7 | ☒ |
| 8 | compute_tax | b | Compute tax | ☐ | 1 | 8 | ☒ |
| 9 | format | A | File format | 56 | 6 | 9 | ☒ |
| 10 | agg_trans | b | Compress | ☐ | 1 | 10 | ☒ |
| 11 | keep_batch | b | Remove file | ☒ | 1 | 11 | ☒ |
| 12 | agg_apar | b | Compress AR/AP | ☐ | 1 | 12 | ☒ |
| 13 | max_errors | n | Maximum numbe | 0 | 6 | 13 | ☒ |
| 14 | trig_chk | b | Trigger check | ☐ | 1 | 15 | ☒ |
| 15 | query | A | Query | GL07M | 8 | 16 | ☒ |
| 16 | query_param1 | a | File name | ITK_TEST.csv | 30 | 17 | ☐ |
| 17 | query_param2 | a | Query Parameter | | 30 | 18 | ☒ |
| 18 | query_param3 | a | Query Parameter | | 30 | 19 | ☒ |
| 19 | recalc_v2 | b | Compute Amoun | ☒ | 1 | 20 | ☒ |
| 20 | recalc_v3 | b | Compute Amoun | ☒ | 1 | 21 | ☒ |
| 21 | recalc_amt | b | Compute Amoun | ☐ | 1 | 22 | ☒ |
| 22 | company | W | Company | * | 25 | 23 | ☒ |
| 23 | only_errors | b | Only errors | ☐ | 1 | 24 | ☒ |
| 24 | start_tps | b | Start AGRTPS | ☐ | 1 | 25 | ☒ |
| 25 | registration | b | Registration | ☐ | 1 | 26 | ☒ |
| 26 | asqlflag | b | Agresso SQL | ☒ | 1 | 27 | ☒ |
| 27 | report_file | A | Report file | GL07 | 12 | 28 | ☒ |
| 28 | max_inv_diff | b | Max.trans.diff | ☐ | 1 | 29 | ☒ |
| 29 | seq_no_flag | b | SequenceNo assig | ☒ | 1 | 31 | ☒ |
| 30 | init_ag16 | b | Initial AG16 | ☒ | 1 | 32 | ☒ |
| 31 | chk_vou_date | A | Check transaction | 2 | 1 | 33 | ☒ |
| 32 | report_file2 | A | Warning report | GL07B | 12 | 34 | ☒ |
| 33 | mult_vou_dat | b | Allow multiple tra | ☐ | 1 | 35 | ☒ |
| 34 | orphan_tax | b | Allow orphan tax | ☐ | 1 | 36 | ☒ |
| 35 | itk_start | a | ITK batch start | ConvCsv | 20 | 100 | ☒ |
| 36 | name_in | a | File name in | query_param1 | 100 | 101 | ☒ |
| 37 | num_headers | a | No of header reco | 1 | 100 | 102 | ☒ |
| 38 | num_footers | a | No of footer recor | 1 | 100 | 103 | ☒ |

## Parameter ‘Filename (file_name)’

The ‘Default value’ must be blank and ‘Fixed’ must be checked so that GL07 does not try to read the file as a standard GL07 file.

## Parameter ‘Query (query)’

The ‘Default value’ must be set to the appropriate query’s name.

## Parameter ‘Query parameter 1 (query_param1)’ (and similar)

Change the ‘Param name’ to “File name” and unfix it.

## Parameter ‘Agresso SQL (asqlflag)’

If your query is written in ASQL then change the ‘Default value’ to a 1.

## New Parameter 'ITK batch start (itk_start)'

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | "ConvCsv" |
| **Length** | "30" |
| **#** | *On a report variant this must be greater than or equal to "100", on a user defined report this can be the next available number.* |
| **Fixed** | ☒ |

## New Parameter 'File names in (name_in)'

This parameter is used to define the number and names of the CSV files that are to be converted into TSV files.

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | *A comma separated list of the names of the parameters containing the file names to convert ("query_param1" in the example). Spaces on either side of each comma are ignored.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'No of header records (num_headers)'

This parameter is used to specify whether the CSV files have any header records and if they do how many.

This parameter is optional so if none of the files have header records it can either be omitted or it can be included and have its value set to zero. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | • *If all files in 'name_in' have the same number of header records then set 'Default value' to that single number.*<br>• *Otherwise set 'Default value' to a comma separated list of numbers of header records, each value corresponds to the file in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |

| Column | Value |
|---|---|
| **Fixed** | ☒ |

## New Parameter 'No of footer records (num_footers)'

This parameter is used to specify whether the CSV files have any footer records and if they do how many.

This parameter is optional so if none of the files have footer records it can either be omitted or it can be included and have its value set to zero. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | • *If all files in 'name_in' have the same number of footer records then set 'Default value' to that single number.*<br>• *Otherwise set 'Default value' to a comma separated list of numbers of footer records, each value corresponds to the file in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Field separator char (separator)'

By default 'ConvCSV' assumes that the fields in the CSV files are separated by commas, this parameter can be used to override this assumption.

This parameter is optional so if all of the files use a comma as their field separator then it can either be omitted or it can be included and have its value set to a comma. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | • *If all files in 'name_in' have the same field separator character then set 'Default value' to that single character.*<br>• *Otherwise set 'Default value' to a comma separated list of single characters, each value corresponds to the file in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

**WARNING**

If you wish to explicitly define a comma as the field separator character then you must enclose it in apostrophes e.g.

' , '

Failure to do this will result in 'ConvCSV' miscounting the number of values in this parameter.

### New Parameter 'Remove non-ASCII chars (asc_only)'

This parameter is used to control whether any non-ASCII characters in the CSV files should be transferred to the TSV files or not.

This parameter is optional so if all of the files whose names are in the 'name_in' parameter can only contain characters in the ASCII character set (see section 5.2) or there could be non-ASCII characters and you do not want them removed then this parameter can either be omitted or it can be included and have its value set to "N". If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "A" |
| **Default value** | • *If all the files in 'name_in' could contain non-ASCII characters and you want them removed then set 'Default value' to "*Y*".*<br>• *If all of the files in 'name_in' can only contain ASCII characters or they could contain non-ASCII characters but you don't want them removed then set 'Default value' to "*N*".*<br>• *Otherwise set 'Default value' to a comma separated list of "*Y*"s and "*N*"s, , each value corresponds to the file in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## 3.1.3 Example File and Query

As an example here is one of the files and queries used during testing; it posts a simple GL journal and relates to the report variant screen shot above. The file header contains a date to be used as the transaction date and the footer contains a total-debit figure that the query will "throw" an error on if it is incorrect

### The File

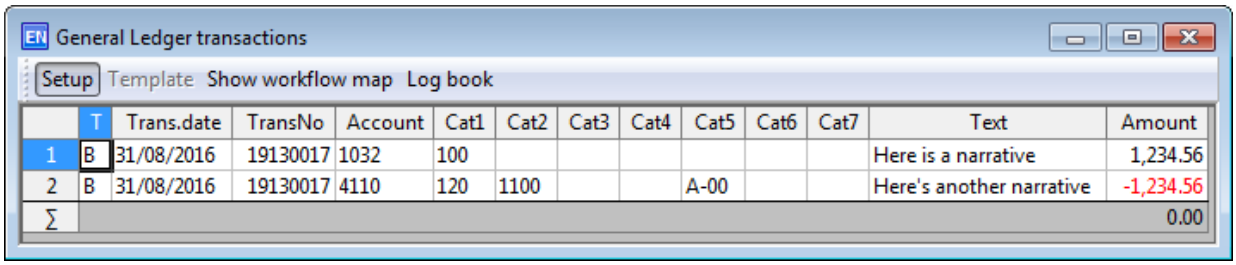
| Line | Contents |
|---|---|
| 1 | 31/08/2016 |
| 2 | 1032,100,'','','','','',1234.56,'Here is a narrative' |
| 3 | '4110',120,'1100',,,'A-00',,,-1234.56,'Here''s another narrative' |

| Line | Contents |
|---|---|
| 4 | 1234.56 |

## The SQL Query

See section 4.2.

## The Results




| | T | Trans.date | TransNo | Account | Cat1 | Cat2 | Cat3 | Cat4 | Cat5 | Cat6 | Cat7 | Text | Amount |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | B | 31/08/2016 | 19130017 | 1032 | 100 | | | | | | | Here is a narrative | 1,234.56 |
| 2 | B | 31/08/2016 | 19130017 | 4110 | 120 | 1100 | | | A-00 | | | Here's another narrative | -1,234.56 |
| Σ | | | | | | | | | | | | | 0.00 |

## 3.2 Excel Workbook Conversion (ConvXls)

### 3.2.1 Introduction

Some systems (including Business World) provide the facility to export data in the form of an Excel workbook, either in the old "xls" format or in the more recent "xlsx" format.

Unfortunately these files cannot be read by ASQL's "COPY IN" command. This fact has been a minor implementation issue on a number of occasions. See section 5.1 for an overview of using queries to import data.

'ConvXls' will take one or more worksheets in one or more Excel workbooks in the folder defined by the "AGRESSO_IMPORT" environment variable and translate them into new files that have tab separated values (TSV), which can be read by the "COPY IN" command. It then changes the value of the "file_name" parameters so that the SQL query can process the TSV versions of the worksheets and finally it removes the original workbooks from the "AGRESSO_IMPORT" folder:

- In a SaaS environment they are simply deleted.
- In other environments they are moved to the folder defined by the "AGRESSO_SCRATCH" environment variable, or the folder defined by the 'TEMP' Windows environment variable if AGRESSO_SCRATCH is not set.

The new TSV files will have a "tsv" extension and will be present in the AGRESSO_IMPORT folder.

Worksheets must be tabular in form and may or may not have column names on row one. If there are column names then 'ConvXls' should be told to ignore them by using the 'col_names' parameter otherwise all the columns will be treated as strings which means that dates will end up in American format in the TSV file. Rows immediately after the column names (if any) can be treated as "header" records by using the 'num_headers' parameter and the final rows in the worksheet can treated as "footer" records by using the 'num_footers' parameter. The header and footer rows do not have to have a value in the same number of columns as the detail rows.

If a worksheet has header records and/or footer rows then an additional field will be inserted at the beginning of each record of the TSV file, this field will be of the form:

"*Xn*"

Where *X* is

- "H" on a header record,
- "D" on a detail record
- "F" on a footer record

And *n* is a sequence number which starts from one for each record type. For example:

- "H1" is the first header record,
- "D1" is the first detail record,
- "D235" is the two hundredth and thirty fifth detail record
- "F1" is the first footer record

If a file has neither header records nor footer records then this extra field is not created.

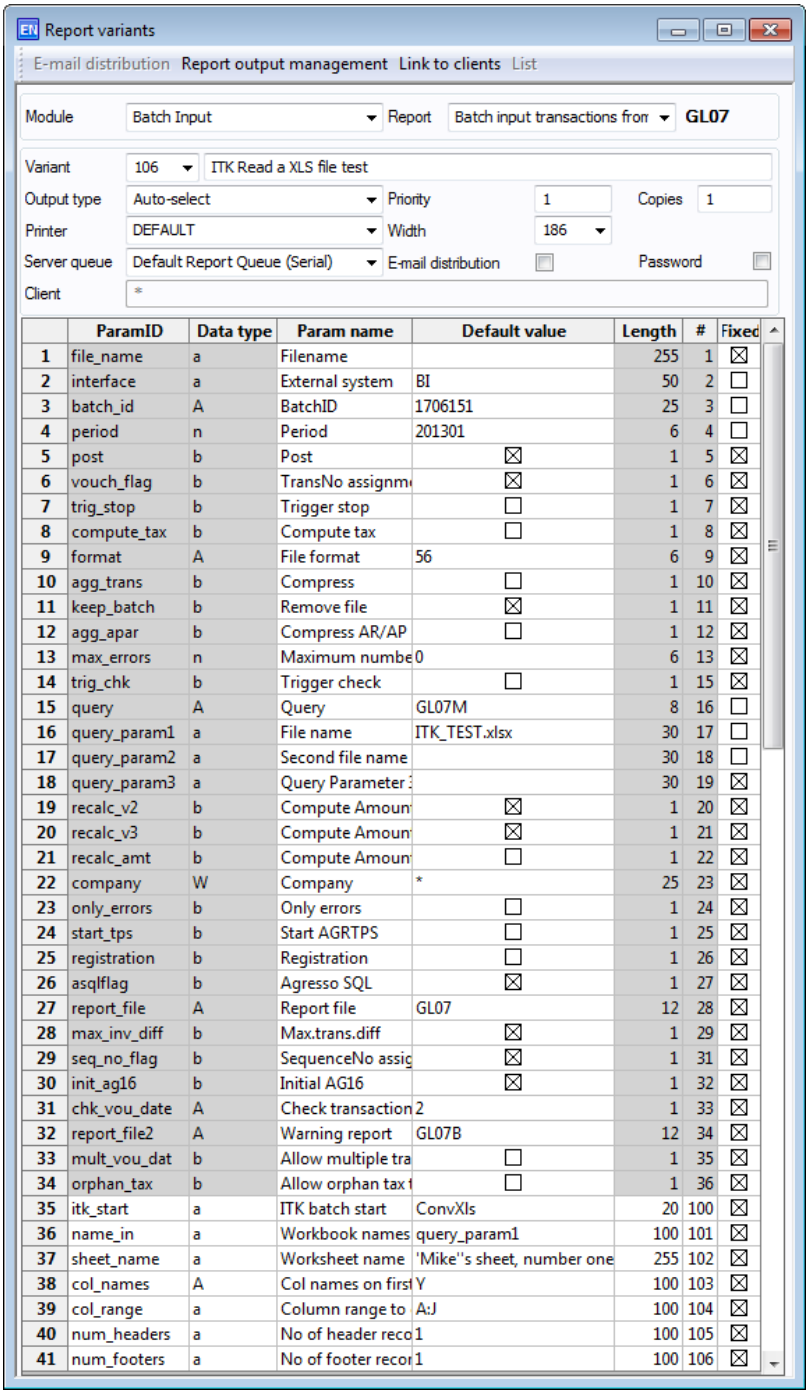
**WARNING**

The contents of a header or footer record will be put into the first data field of the output record with an appropriate number of trailing tab characters. So when you use the "COPY IN" command it is important that the table column you are copying the first data field to is large enough to hold any possible header or footer data. For example in the query referred to in section 3.2.3 the 'account' column is defined as "SPACE(25)" so that it is large enough to hold the date from the header (10 characters would have been enough for this) or the total debit amount from the footer (the line amount is 15 characters so 18 or 20 should have been enough but you can't be too careful!).

Similarly the table column that the additional field that the program generates is going to be copied into ('rec_type' in the query below) must be big enough to hold at least the expected number of detail records, if you could have a million detail records then the column must be at least eight characters long to hold "D1000000".

### 3.2.2 Set Up

In order to process Excel worksheets in a data import routine you must create a variant with some of its standard parameters changed and some new parameters added. The example is for GL07 but the same principle applies to any other data import routine.




| | ParamID | Data type | Param name | Default value | Length | # | Fixed |
|---|---|---|---|---|---|---|---|
| 1 | file_name | a | Filename | | 255 | 1 | ☒ |
| 2 | interface | a | External system | BI | 50 | 2 | ☐ |
| 3 | batch_id | A | BatchID | 1706151 | 25 | 3 | ☐ |
| 4 | period | n | Period | 201301 | 6 | 4 | ☐ |
| 5 | post | b | Post | ☒ | 1 | 5 | ☒ |
| 6 | vouch_flag | b | TransNo assignm | ☒ | 1 | 6 | ☒ |
| 7 | trig_stop | b | Trigger stop | ☐ | 1 | 7 | ☒ |
| 8 | compute_tax | b | Compute tax | ☐ | 1 | 8 | ☒ |
| 9 | format | A | File format | 56 | 6 | 9 | ☒ |
| 10 | agg_trans | b | Compress | ☐ | 1 | 10 | ☒ |
| 11 | keep_batch | b | Remove file | ☒ | 1 | 11 | ☒ |
| 12 | agg_apar | b | Compress AR/AP | ☐ | 1 | 12 | ☒ |
| 13 | max_errors | n | Maximum numbe | 0 | 6 | 13 | ☒ |
| 14 | trig_chk | b | Trigger check | ☐ | 1 | 15 | ☒ |
| 15 | query | A | Query | GL07M | 8 | 16 | ☐ |
| 16 | query_param1 | a | File name | ITK_TEST.xlsx | 30 | 17 | ☐ |
| 17 | query_param2 | a | Second file name | | 30 | 18 | ☐ |
| 18 | query_param3 | a | Query Parameter | | 30 | 19 | ☒ |
| 19 | recalc_v2 | b | Compute Amoun | ☒ | 1 | 20 | ☒ |
| 20 | recalc_v3 | b | Compute Amoun | ☒ | 1 | 21 | ☒ |
| 21 | recalc_amt | b | Compute Amoun | ☐ | 1 | 22 | ☒ |
| 22 | company | W | Company | * | 25 | 23 | ☒ |
| 23 | only_errors | b | Only errors | ☐ | 1 | 24 | ☒ |
| 24 | start_tps | b | Start AGRTPS | ☐ | 1 | 25 | ☒ |
| 25 | registration | b | Registration | ☐ | 1 | 26 | ☒ |
| 26 | asqlflag | b | Agresso SQL | ☒ | 1 | 27 | ☒ |
| 27 | report_file | A | Report file | GL07 | 12 | 28 | ☒ |
| 28 | max_inv_diff | b | Max.trans.diff | ☒ | 1 | 29 | ☒ |
| 29 | seq_no_flag | b | SequenceNo assig | ☒ | 1 | 31 | ☒ |
| 30 | init_ag16 | b | Initial AG16 | ☒ | 1 | 32 | ☒ |
| 31 | chk_vou_date | A | Check transaction | 2 | 1 | 33 | ☒ |
| 32 | report_file2 | A | Warning report | GL07B | 12 | 34 | ☒ |
| 33 | mult_vou_dat | b | Allow multiple tra | ☐ | 1 | 35 | ☒ |
| 34 | orphan_tax | b | Allow orphan tax t | ☐ | 1 | 36 | ☒ |
| 35 | itk_start | a | ITK batch start | ConvXls | 20 | 100 | ☒ |
| 36 | name_in | a | Workbook names | query_param1 | 100 | 101 | ☒ |
| 37 | sheet_name | a | Worksheet name | 'Mike''s sheet, number one | 255 | 102 | ☒ |
| 38 | col_names | A | Col names on first | Y | 100 | 103 | ☒ |
| 39 | col_range | a | Column range to | A:J | 100 | 104 | ☒ |
| 40 | num_headers | a | No of header reco | 1 | 100 | 105 | ☒ |
| 41 | num_footers | a | No of footer recor | 1 | 100 | 106 | ☒ |

## Parameter ‘Filename (file_name)’

The ‘Default value’ must be blank and ‘Fixed’ must be checked so that GL07 does not try to read the file as a standard GL07 file.

## Parameter ‘Query (query)’

The ‘Default value’ must be set to the appropriate query’s name.

## Parameter ‘Query parameter 1 (query_param1)’ (and similar)

Change the ‘Param name’ to “File name” and unfix it.

## Parameter 'Agresso SQL (asqlflag)'

If your query is written in ASQL then change the 'Default value' to a 1.

## New Parameter 'ITK batch start (itk_start)'

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | "ConvXls" |
| **Length** | "30" |
| **#** | *On a report variant this must be greater than or equal to "100", on a user defined report this can be the next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Workbook names in (name_in)'

This parameter is used to define the number and names of the Excel workbooks that are to have worksheets converted into TSV files.

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | *A comma separated list of the names of the parameters containing the workbook names that contain the worksheets to convert ("query_param1" in the example). Spaces on either side of each comma are ignored.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Worksheet name (sheet_name)'

By default 'ConvXls' assumes that each of the worksheets that you wish to convert is called "Sheet1", this parameter can be used to override this assumption.

This parameter is optional so if all the worksheets are called "Sheet1" then it can either be omitted or can be included and have its value set to "Sheet1". If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |

| Column | Value |
|---|---|
| **Default value** | • *If all workbooks in 'name_in' have the same worksheet name then set 'Default value' to that single worksheet name.*<br>• *Otherwise set 'Default value' to a comma separated list of worksheet names, each value corresponds to the workbook in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

### 💣 WARNING

If a worksheet name contains one or more commas then you must enclose it in apostrophes e.g.

'My, Sheet 3'.

If you have had to enclose the worksheet name in apostrophes and that name also contains apostrophes then each of these must be doubled up e.g.

'Mike''s difficult, sheet name'.

Failure to comply with the preceding rules will result in 'ConvXls' miscounting the number of values in this parameter.

## New Parameter 'Col names on first row (col_names)'

By default 'ConvXls' assumes that each of the worksheets that you wish to convert has column names on row one and you do not wish these to be transferred to the TSV files, if the worksheet does not have column names on row one then this assumption means that the first row of data will be ignored. This parameter can be used to override this behaviour to avoid this problem.

This parameter is optional so if you want to discard row one in all of the worksheets then it can either be omitted or it can be included and have its value set to "Y". If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "A" |

| Column | Value |
|---|---|
| **Default value** | • *If you wish to discard row one in all of the worksheets defined by 'name_in' and 'sheet_name' then set 'Default value' to "Y".*<br>• *If you wish to retain row one in all of the worksheets defined by 'name_in' and 'sheet_name' then set 'Default value' to "N".*<br>• *Otherwise set 'Default value' to a comma separated list of "Y"s and "N"s, each value corresponds to the workbook in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Column range to import (col_range)'

By default 'ConvXls' assumes that you wish to convert the whole of each of the worksheets, this parameter can be used to override this assumption.

This parameter is optional so if you wish to convert the entire content of each worksheet then it can either be omitted or it can be included and have its value left blank. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "A" |
| **Default value** | • *If you wish to convert the entire content of each of the worksheets defined by 'name_in' and 'sheet_name' then set 'Default value' to blank.*<br>• *If you wish to convert the same range of columns in each of the worksheets defined by 'name_in' and 'sheet_name' then set 'Default value' to that range e.g. "A:J" means include columns "A" to "J" inclusive.*<br>• *Otherwise set 'Default value' to a comma separated list of ranges, each value corresponds to the workbook in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'No of header records (num_headers)'

This parameter is used to specify whether the worksheets have any header records and if they do how many.

This parameter is optional so if none of the worksheets have header records it can either be omitted or it can be included and have its value set to zero. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | • *If all of the worksheets defined by 'name_in' and 'sheet_name' have the same number of header records then set 'Default value' to that single number.*<br>• *Otherwise set 'Default value' to a comma separated list of numbers of header records, each value corresponds to the file in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'No of footer records (num_footers)'

This parameter is used to specify whether the worksheets have any footer records and if they do how many.

This parameter is optional so if none of the worksheets have footer records it can either be omitted or it can be included and have its value set to zero. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | • *If all of the worksheets defined by 'name_in' and 'sheet_name' have the same number of footer records then set 'Default value' to that single number.*<br>• *Otherwise set 'Default value' to a comma separated list of numbers of footer records, each value corresponds to the file in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Remove workbook (del_book)'

By default 'ConvXls' removes each workbook from the folder defined by the "AGRESSO_IMPORT" environment variable after it has been converted into a TSV file, if you wish to wish to convert another worksheet in the same workbook then the removal of the workbook after converting the first worksheet will cause the conversion of the second worksheet to fail. This parameter can be used to override this behaviour.

This parameter is optional so if you wish each workbook to be removed from the "AGRESSO_IMPORT" folder after it has been converted then the parameter can either be omitted or it can be included and have its value set to "Y". If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "A" |
| **Default value** | • *If you wish all the workbooks in 'name_in' to be removed from the "AGRESSO_IMPORT" folder then set its 'Default value' to "Y".*<br>• *If you wish none of the workbooks in 'name_in' to be removed from the "AGRESSO_IMPORT" folder then set 'Default value' to "N", this should be used with caution!*<br>• *Otherwise set 'Default value' to a comma separated list of "Y"s and "N"s, each value corresponds to the workbook in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.*<br>***N.B.** Only the final mention of each workbook should have its value set to "Y".* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

### 3.2.3 Example Workbook and Query

As an example here is one of the files and queries used during testing; it posts a simple GL journal and relates to the report variant screen shot above. The file header contains a date to be used as the transaction date and the footer contains a total-debit figure that the query will "throw" an error on if it is incorrect

## The Workbook

| A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|
| account | dim_1 | dim_2 | dim_3 | dim_4 | dim_5 | dim_6 | dim_7 | cur_amount | description |
| 31/08/2016 | | | | | | | | | |
| 1032 | 100 | | | | | | | 1234.56 | Here is a narrative |
| 4110 | 120 | 1100 | | | A-00 | | | -1234.56 | Here's another narrative |
| 1234.56 | | | | | | | | | |

**WARNING**

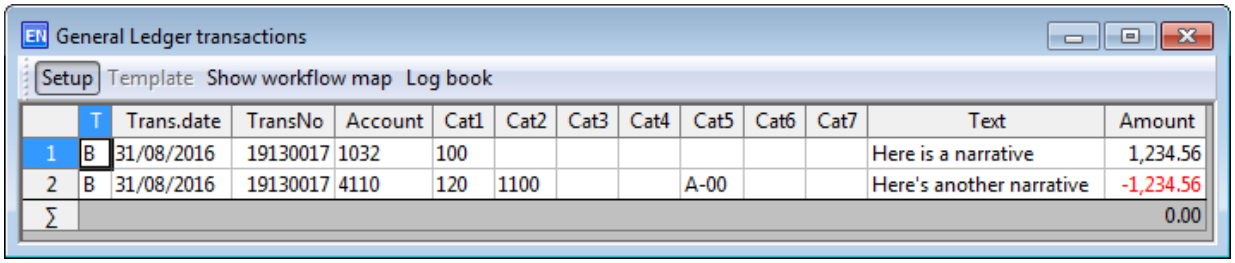
Because column A contains a mixture of dates, integers and amounts its data type is "String", so if the date on row 2 is not entered as a string (i.e. you prefix it with a single apostrophe) then its value will be converted to "3/1/2005" (i.e. American format) and this is what will end up in the TSV file.

Columns that contain only dates will not suffer from this problem.

### The SQL Query

See section 4.2.

### The Results




| | T | Trans.date | TransNo | Account | Cat1 | Cat2 | Cat3 | Cat4 | Cat5 | Cat6 | Cat7 | Text | Amount |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | B | 31/08/2016 | 19130017 | 1032 | 100 | | | | | | | Here is a narrative | 1,234.56 |
| 2 | B | 31/08/2016 | 19130017 | 4110 | 120 | 1100 | | | A-00 | | | Here's another narrative | -1,234.56 |
| ∑ | | | | | | | | | | | | | 0.00 |

## 3.2.4 Error Messages

### 3.2.4.1 The '*nnnnnnn*' provider is not registered on the local machine.

Where *nnnnnnn* is either:

- "Microsoft.ACE.OLEDB.12.0" if you trying to convert an "xlsx" workbook
- "Microsoft.Jet.OLEDB.4.0" if you trying to convert an "xls" workbook

The 'ConvXls' method uses OLE DB[10] to read the data from the Excel workbook; so to be able to extract the data from the workbook the appropriate OLE DB provider must be installed on the Business World Business-Server. If you get this message then either:

- The provider named in the error message is not installed
- The provider named in the error message is incorrectly installed
- You do not have providers for the architecture of the server queue that you are running the process on, e.g. you only have 32bit providers and you're running this process on a 64 bit server queue.

In order to correct this error you need to:

- Determine the name of the installed provider for this type of workbook (ask your system administrator if you do not know how to do this yourself) and override the default provider by setting one of two optional parameters that are not documented in section 3.2.2.
- Ensure that the providers are correctly installed on the Business-Server.
- Ensure that the installed providers match the architecture of the server queue you are running the process on.

[10] *Object Linking and Embedding, Database, an API designed by Microsoft, which allows accessing data from a variety of sources in a uniform manner.*
From WIKIPEDIA.

**WARNING**

UNIT4 has only tested 'ConvXls' with the default OLE DB providers named above on a 32bit server queue. It is probable that it would also work with other providers from Microsoft, such as the 64bit ones, but less likely with providers from other sources.

If you are using providers other than the defaults then it is possible that the wording of the error messages in section 3.2.4 will be different.

## New Parameter 'OLE DB provider for xls (prov_xls)'

This parameter is used to override the default OLE DB provider for reading from an "xls" workbook.

This parameter is optional so if the installed OLE DB provider is "Microsoft.Jet.OLEDB.4.0" then it should be omitted or it can be included and have its value left blank. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | *The name of the installed OLE DB provider for "xls" files.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

If you get a similar error message when you rerun the process then you have either mistyped the 'Default value' of this parameter or you have been misinformed about the name of the provider to use.

## New Parameter 'OLE DB provider for xlsx (prov_xlsx)'

This parameter is used to override the default OLE DB provider for reading from an "xlsx" workbook.

This parameter is optional so if the installed OLE DB provider is "Microsoft.ACE.OLEDB.12.0" then it should be omitted or it can be included and have its value left blank. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | *The name of the installed OLE DB provider for "xlsx" files.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

If you get a similar error message when you rerun the process then you have either mistyped the 'Default value' of this parameter or you have been misinformed about the name of the provider to use.

#### 3.2.4.2 External table is not in the expected format.

This error means either that:

- The file you are trying to convert is not a workbook.
- The wrong OLE DB provider is being used for this type of workbook, see the ‘prov_xls’ and ‘prov_xlsx’ process parameters in section 3.2.4.1.

#### 3.2.4.3 '*nnnnnn*$' is not a valid name. Make sure that it does not include invalid characters or punctuation and that it is not too long.

Where *nnnnnn* is the name of the worksheet you are trying to convert.

This error means that there is no worksheet in the specified workbook with this name.

## 3.3 CFR Import File Conversion (ConvCfr)

### 3.3.1 Introduction

Some main-frame systems generate files that have a fixed number of fields per record and each field is on a separate line; there is no delimiting character at the end of each record, so the only way to know when you have reached the end of a record is to "count the fields". I have coined the acronym CFR (Counted Fields per Record) to describe this type of file.

Unfortunately this format cannot be read by ASQL's "COPY IN" command. This fact has been a minor implementation issue on a number of occasions. See section 5.1 for an overview of using queries to import data.

'ConvCfr' will take one or more existing CFR files in the folder defined by the "AGRESSO_IMPORT" environment variable and translate them into new files that have tab separated values (TSV), which can be read by the "COPY IN" command. It then changes the value of the "file_name" parameters so that the SQL query can process the TSV versions of the CSV and finally it removes the original CFR files from the "AGRESSO_IMPORT" folder:

- In a SaaS environment they are simply deleted.
- In other environments they are moved to the folder defined by the "AGRESSO_SCRATCH" environment variable, or the folder defined by the 'TEMP' Windows environment variable if AGRESSO_SCRATCH is not set.

The new TSV files will have a "tsv" extension and will be present in the AGRESSO_IMPORT folder.

The content of each line on the input file is not inspected by 'ConvCfr; and so could contain more than one data item; if this is the case then your SQL query must break the field into its constituent data items using substring operations ("SUBSTR(…)" in ASQL).

If a file has header records and/or footer records then an additional field will be inserted at the beginning of each record of the TSV file, this field will be of the form:

"*Xn*"

Where *X* is

- "H" on a header record,
- "D" on a detail record
- "F" on a footer record

And *n* is a sequence number which starts from one for each record type. For example:

- "H1" is the first header record,
- "D1" is the first detail record,
- "D235" is the two hundredth and thirty fifth detail record
- "F1" is the first footer record

If a file has neither header records nor footer records then this extra field is not created.

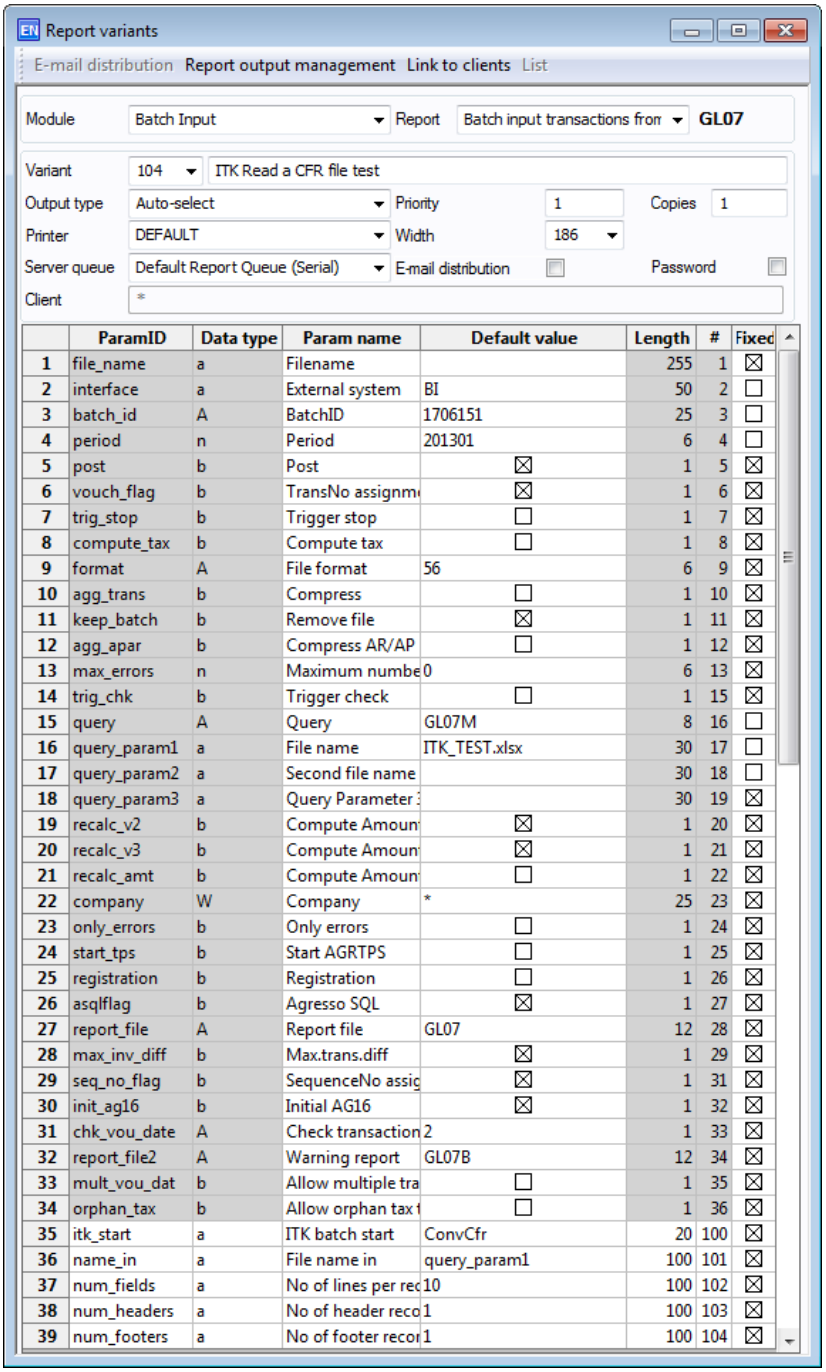
**WARNING**

The contents of a header or footer record will be put into the first data field of the output record with an appropriate number of trailing tab characters. So when you use the "COPY IN" command it is important that the table column you are copying the first data field to is large enough to hold any possible header or footer data. For example in the query referred to in section 3.3.3 the 'account' column is defined as "SPACE(25)" so that it is large enough to hold the date from the header (10 characters would have been enough for this) or the total debit amount from the footer (the line amount is 15 characters so 18 or 20 should have been enough but you can't be too careful!).

Similarly the table column that the additional field that the program generates is going to be copied into ('rec_type' in the query below) must be big enough to hold at least the expected number of detail records, if you could have a million detail records then the column must be at least eight characters long to hold "D1000000".

### 3.3.2 Set Up

In order to process CFR files in a data import routine you must create a variant with some of its standard parameters changed and some new parameters added. The example is for GL07 but the same principle applies to any other data import routine.




| | ParamID | Data type | Param name | Default value | Length | # | Fixed |
|---|---|---|---|---|---|---|---|
| 1 | file_name | a | Filename | | 255 | 1 | ☒ |
| 2 | interface | a | External system | BI | 50 | 2 | ☐ |
| 3 | batch_id | A | BatchID | 1706151 | 25 | 3 | ☐ |
| 4 | period | n | Period | 201301 | 6 | 4 | ☐ |
| 5 | post | b | Post | ☒ | 1 | 5 | ☒ |
| 6 | vouch_flag | b | TransNo assignm | ☒ | 1 | 6 | ☒ |
| 7 | trig_stop | b | Trigger stop | ☐ | 1 | 7 | ☒ |
| 8 | compute_tax | b | Compute tax | ☐ | 1 | 8 | ☒ |
| 9 | format | A | File format | 56 | 6 | 9 | ☒ |
| 10 | agg_trans | b | Compress | ☐ | 1 | 10 | ☒ |
| 11 | keep_batch | b | Remove file | ☒ | 1 | 11 | ☒ |
| 12 | agg_apar | b | Compress AR/AP | ☐ | 1 | 12 | ☒ |
| 13 | max_errors | n | Maximum numbe | 0 | 6 | 13 | ☒ |
| 14 | trig_chk | b | Trigger check | ☐ | 1 | 15 | ☒ |
| 15 | query | A | Query | GL07M | 8 | 16 | ☐ |
| 16 | query_param1 | a | File name | ITK_TEST.xlsx | 30 | 17 | ☐ |
| 17 | query_param2 | a | Second file name | | 30 | 18 | ☐ |
| 18 | query_param3 | a | Query Parameter | | 30 | 19 | ☒ |
| 19 | recalc_v2 | b | Compute Amoun | ☒ | 1 | 20 | ☒ |
| 20 | recalc_v3 | b | Compute Amoun | ☒ | 1 | 21 | ☒ |
| 21 | recalc_amt | b | Compute Amoun | ☐ | 1 | 22 | ☒ |
| 22 | company | W | Company | * | 25 | 23 | ☒ |
| 23 | only_errors | b | Only errors | ☐ | 1 | 24 | ☒ |
| 24 | start_tps | b | Start AGRTPS | ☐ | 1 | 25 | ☒ |
| 25 | registration | b | Registration | ☐ | 1 | 26 | ☒ |
| 26 | asqlflag | b | Agresso SQL | ☒ | 1 | 27 | ☒ |
| 27 | report_file | A | Report file | GL07 | 12 | 28 | ☒ |
| 28 | max_inv_diff | b | Max.trans.diff | ☒ | 1 | 29 | ☒ |
| 29 | seq_no_flag | b | SequenceNo assig | ☒ | 1 | 31 | ☒ |
| 30 | init_ag16 | b | Initial AG16 | ☒ | 1 | 32 | ☒ |
| 31 | chk_vou_date | A | Check transaction | 2 | 1 | 33 | ☒ |
| 32 | report_file2 | A | Warning report | GL07B | 12 | 34 | ☒ |
| 33 | mult_vou_dat | b | Allow multiple tra | ☐ | 1 | 35 | ☒ |
| 34 | orphan_tax | b | Allow orphan tax t | ☐ | 1 | 36 | ☒ |
| 35 | itk_start | a | ITK batch start | ConvCfr | 20 | 100 | ☒ |
| 36 | name_in | a | File name in | query_param1 | 100 | 101 | ☒ |
| 37 | num_fields | a | No of lines per rec | 10 | 100 | 102 | ☒ |
| 38 | num_headers | a | No of header reco | 1 | 100 | 103 | ☒ |
| 39 | num_footers | a | No of footer recor | 1 | 100 | 104 | ☒ |

## Parameter 'Filename (file_name)'

The 'Default value' must be blank and 'Fixed' must be checked so that GL07 does not try to read the file as a standard GL07 file.

## Parameter 'Query (query)'

The 'Default value' must be set to the appropriate query's name.

## Parameter 'Query parameter 1 (query_param1)' (and similar)

Change the 'Param name' to "File name" and unfix it.

## Parameter 'Agresso SQL (asqlflag)'

If your query is written in ASQL then change the 'Default value' to a 1.

## New Parameter 'ITK batch start (itk_start)'

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | "ConvCfr" |
| **Length** | "30" |
| **#** | *On a report variant this must be greater than or equal to "100", on a user defined report this can be the next available number.* |
| **Fixed** | ☒ |

## New Parameter 'File names in (name_in)'

This parameter is used to define the number and names of the CFR files that are to be converted into TSV files.

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | *A comma separated list of the names of the parameters containing the file names to convert ("query_param1" in the example). Spaces on either side of each comma are ignored.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Number of lines per record (num_fields)'

This parameter is used to define the number of lines in the CFR file that make up one logical record in the TSV file.

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | • *If all files in 'name_in' have the same number of fields per record then set 'Default value' to that single number.*<br>• *Otherwise set 'Default value' to a comma separated list of numbers of fields, each value corresponds to the file in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |

| Column | Value |
|---|---|
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'No of header records (num_headers)'

This parameter is used to specify whether the CFR files have any header records and if they do how many.

This parameter is optional so if none of the files have header records it can either be omitted or it can be included and have its value set to zero. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | • *If all files in 'name_in' have the same number of header records then set 'Default value' to that single number.*<br>• *Otherwise set 'Default value' to a comma separated list of numbers of header records, each value corresponds to the file in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'No of footer records (num_footers)'

This parameter is used to specify whether the CFR files have any footer records and if they do how many.

This parameter is optional so if none of the files have footer records it can either be omitted or it can be included and have its value set to zero. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | • *If all files in 'name_in' have the same number of footer records then set 'Default value' to that single number.*<br>• *Otherwise set 'Default value' to a comma separated list of numbers of footer records, each value corresponds to the file in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Remove non-ASCII chars (asc_only)'

This parameter is used to control whether any non-ASCII characters in the CFR files should be transferred to the TSV files or not.

This parameter is optional so if all of the files whose names are in the 'name_in' parameter can only contain characters in the ASCII character set (see section 5.2) or there could be non-ASCII characters and you do not want them removed then this parameter can either be omitted or it can be included and have its value set to "N". If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "A" |
| **Default value** | • *If all the files in 'name_in' could contain non-ASCII characters and you want them removed then set 'Default value' to "Y".*<br>• *If all of the files in 'name_in' can only contain ASCII characters or they could contain non-ASCII characters but you don't want them removed then set 'Default value' to "N".*<br>• *Otherwise set 'Default value' to a comma separated list of "Y"s and "N"s, , each value corresponds to the file in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

### 3.3.3 Example File and Query

As an example here is one of the files and queries used during testing; it posts a simple GL journal and relates to the report variant screen shot above. The file header contains a date to be used as the transaction date and the footer contains a total-debit figure that the query will "throw" an error on if it is incorrect

## The File

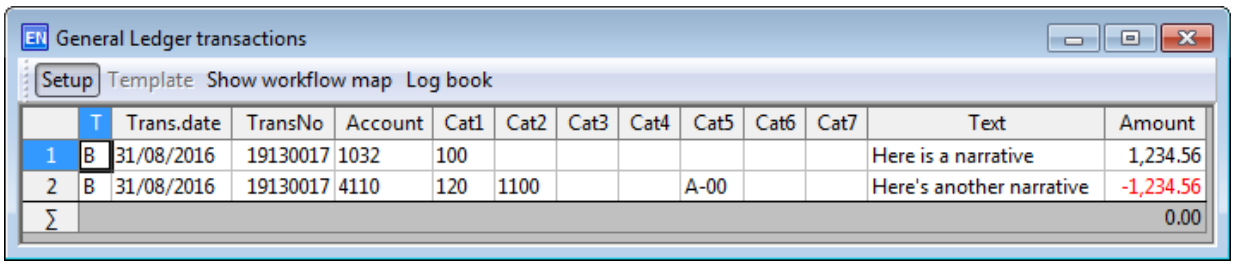
| Line | Contents |
|---|---|
| 1 | 31/08/2016 |
| 2 | 1032 |
| 3 | 100 |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | 1234.56 |

| Line | Contents |
|---|---|
| 11 | Here is a narrative |
| 12 | 4100 |
| 13 | 120 |
| 14 | 1100 |
| 15 | |
| 16 | |
| 17 | A-00 |
| 18 | |
| 19 | |
| 20 | -1234.56 |
| 21 | Here's another narrative |
| 22 | 1234.56 |

## The SQL Query

See section 4.2.

## The Results




| | T | Trans.date | TransNo | Account | Cat1 | Cat2 | Cat3 | Cat4 | Cat5 | Cat6 | Cat7 | Text | Amount |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | B | 31/08/2016 | 19130017 | 1032 | 100 | | | | | | | Here is a narrative | 1,234.56 |
| 2 | B | 31/08/2016 | 19130017 | 4110 | 120 | 1100 | | | A-00 | | | Here's another narrative | -1,234.56 |
| ∑ | | | | | | | | | | | | | 0.00 |

## 3.4 FSR Import File Conversion (ConvFsr)

### 3.4.1 Introduction

A popular way of exporting data from some systems is to use a non-graphical report writer. Some report writers do not have the ability as ARW does to suppress page throws and so a transfer file generated from such a report writer consists of:

- One page per record,
- Each record consisting of one or more lines with a fixed width, usually 80 or 132 characters
- Each line consists of either:
    - one field
    - multiple fields with the same field on each record being in the same character position in the same line on each record

Multiple lines appear in each record either because there is one field per record or because the number of fields required will not fit into the maximum allowed line width.

Because each record starts with a form-feed character (ASCII[11] code 12) I have coined the acronym FSR (Form-feed Separated Record) to describe this type of file.

Depending partly on the report writer used to create the file the form-feed character at the beginning of each record may always be on a line on its own or it may simply be the first character on the first line of the record.

Unfortunately this format cannot be read by ASQL's "COPY IN" command. This fact has been a minor implementation issue on a number of occasions. See section 5.1 for an overview of using queries to import data.

'ConvFsr' will take one or more existing FSR files in the folder defined by the "AGRESSO_IMPORT" environment variable and translate them into new files that have tab separated values (TSV), which can be read by the "COPY IN" command. It then changes the value of the "file_name" parameters so that the SQL query can process the TSV versions of the CSV files and finally it removes the original FSR files from the "AGRESSO_IMPORT" folder:

- In a SaaS environment they are simply deleted.
- In other environments they are moved to the folder defined by the "AGRESSO_SCRATCH" environment variable, or the folder defined by the 'TEMP' Windows environment variable if AGRESSO_SCRATCH is not set.

The new TSV files will have a "tsv" extension and will be present in the AGRESSO_IMPORT folder.

The content of each line on the input file is not inspected by 'ConvFsr' and so could contain more than one data item (see above); if this is the case then your SQL query must break the field into its constituent data items using substring operations ("SUBSTR(…)" in ASQL).

If a file has header records and/or footer records then an additional field will be inserted at the beginning of each record of the TSV file, this field will be of the form:

"*Xn*"

[11] *See section 5.2.*

Where *X* is

- "H" on a header record,
- "D" on a detail record
- "F" on a footer record

And *n* is a sequence number which starts from one for each record type. For example:

- "H1" is the first header record,
- "D1" is the first detail record,
- "D235" is the two hundredth and thirty fifth detail record
- "F1" is the first footer record

If a file has neither header records nor footer records then this extra field is not created.

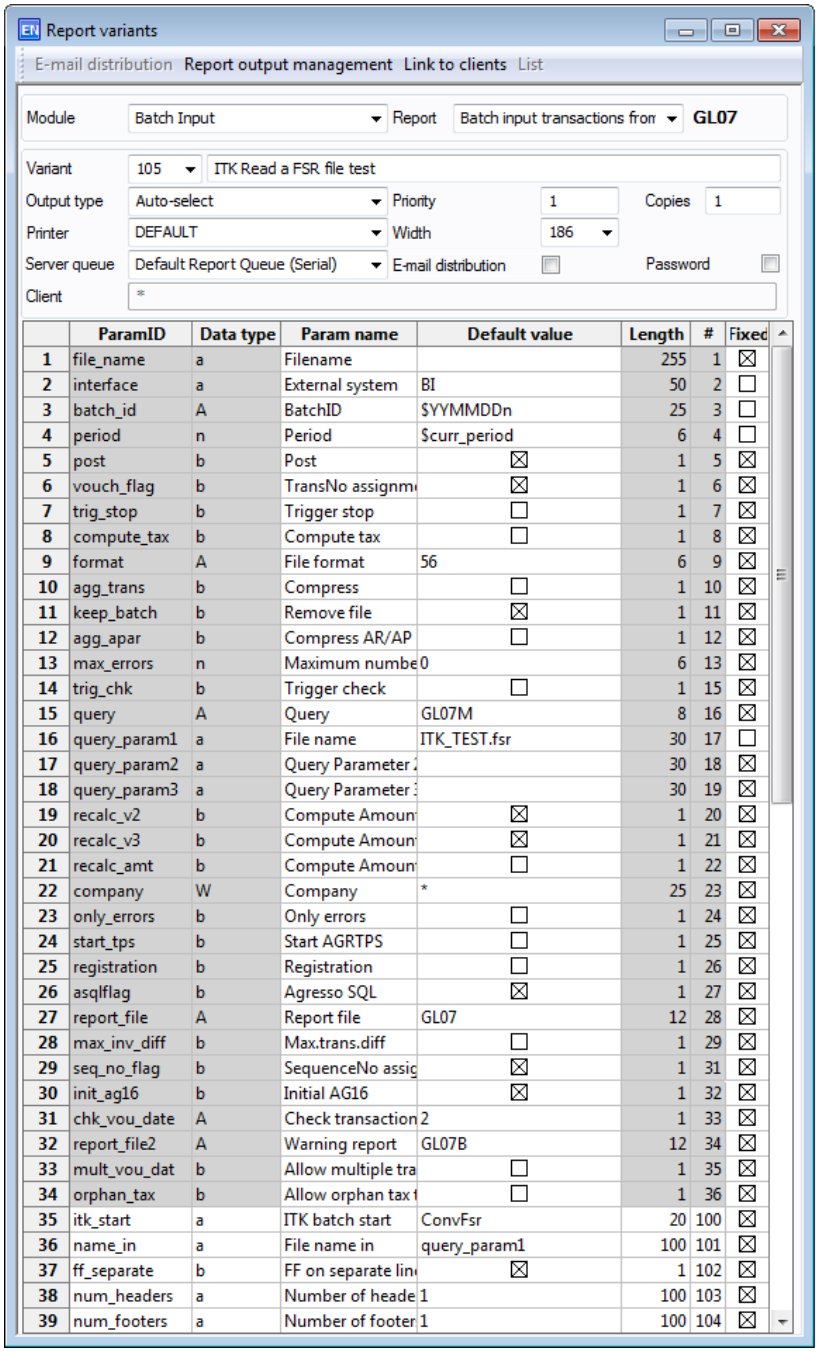
**WARNING**

The contents of a header or footer record will be put into the first data field of the output record with an appropriate number of trailing tab characters. So when you use the "COPY IN" command it is important that the table column you are copying the first data field to is large enough to hold any possible header or footer data. For example in the query referenced in section 3.4.3 the 'account' column is defined as "SPACE(25)" so that it is large enough to hold the date from the header (10 characters would have been enough for this) or the total debit amount from the footer (the line amount is 15 characters so 18 or 20 should have been enough but you can't be too careful!).

Similarly the table column that the additional field that the program generates is going to be copied into ('rec_type' in the query below) must be big enough to hold at least the expected number of detail records, if you could have a million detail records then the column must be at least eight characters long to hold "D1000000".

### 3.4.2 Set Up

In order to process FSR files in a data import routine you must create a variant with some of its standard parameters changed and some new parameters added. The example is for GL07 but the same principle applies to any other data import routine.




| | ParamID | Data type | Param name | Default value | Length | # | Fixed |
|---|---|---|---|---|---|---|---|
| 1 | file_name | a | Filename | | 255 | 1 | ☒ |
| 2 | interface | a | External system | BI | 50 | 2 | ☐ |
| 3 | batch_id | A | BatchID | $YYMMDDn | 25 | 3 | ☐ |
| 4 | period | n | Period | $curr_period | 6 | 4 | ☐ |
| 5 | post | b | Post | ☒ | 1 | 5 | ☒ |
| 6 | vouch_flag | b | TransNo assignm | ☒ | 1 | 6 | ☒ |
| 7 | trig_stop | b | Trigger stop | ☐ | 1 | 7 | ☒ |
| 8 | compute_tax | b | Compute tax | ☐ | 1 | 8 | ☒ |
| 9 | format | A | File format | 56 | 6 | 9 | ☒ |
| 10 | agg_trans | b | Compress | ☐ | 1 | 10 | ☒ |
| 11 | keep_batch | b | Remove file | ☒ | 1 | 11 | ☒ |
| 12 | agg_apar | b | Compress AR/AP | ☐ | 1 | 12 | ☒ |
| 13 | max_errors | n | Maximum numbe | 0 | 6 | 13 | ☒ |
| 14 | trig_chk | b | Trigger check | ☐ | 1 | 15 | ☒ |
| 15 | query | A | Query | GL07M | 8 | 16 | ☒ |
| 16 | query_param1 | a | File name | ITK_TEST.fsr | 30 | 17 | ☐ |
| 17 | query_param2 | a | Query Parameter | | 30 | 18 | ☒ |
| 18 | query_param3 | a | Query Parameter | | 30 | 19 | ☒ |
| 19 | recalc_v2 | b | Compute Amoun | ☒ | 1 | 20 | ☒ |
| 20 | recalc_v3 | b | Compute Amoun | ☒ | 1 | 21 | ☒ |
| 21 | recalc_amt | b | Compute Amoun | ☐ | 1 | 22 | ☒ |
| 22 | company | W | Company | * | 25 | 23 | ☒ |
| 23 | only_errors | b | Only errors | ☐ | 1 | 24 | ☒ |
| 24 | start_tps | b | Start AGRTPS | ☐ | 1 | 25 | ☒ |
| 25 | registration | b | Registration | ☐ | 1 | 26 | ☒ |
| 26 | asqlflag | b | Agresso SQL | ☒ | 1 | 27 | ☒ |
| 27 | report_file | A | Report file | GL07 | 12 | 28 | ☒ |
| 28 | max_inv_diff | b | Max.trans.diff | ☐ | 1 | 29 | ☒ |
| 29 | seq_no_flag | b | SequenceNo assig | ☒ | 1 | 31 | ☒ |
| 30 | init_ag16 | b | Initial AG16 | ☒ | 1 | 32 | ☒ |
| 31 | chk_vou_date | A | Check transaction | 2 | 1 | 33 | ☒ |
| 32 | report_file2 | A | Warning report | GL07B | 12 | 34 | ☒ |
| 33 | mult_vou_dat | b | Allow multiple tra | ☐ | 1 | 35 | ☒ |
| 34 | orphan_tax | b | Allow orphan tax | ☐ | 1 | 36 | ☒ |
| 35 | itk_start | a | ITK batch start | ConvFsr | 20 | 100 | ☒ |
| 36 | name_in | a | File name in | query_param1 | 100 | 101 | ☒ |
| 37 | ff_separate | b | FF on separate lin | ☒ | 1 | 102 | ☒ |
| 38 | num_headers | a | Number of heade | 1 | 100 | 103 | ☒ |
| 39 | num_footers | a | Number of footer | 1 | 100 | 104 | ☒ |

## Parameter ‘Filename (file_name)’

The ‘Default value’ must be blank and ‘Fixed’ must be checked so that GL07 does not try to read the file as a standard GL07 file.

## Parameter ‘Query (query)’

The ‘Default value’ must be set to the appropriate query’s name.

## Parameter ‘Query parameter 1 (query_param1)’ (and similar)

Change the ‘Param name’ to “File name” and unfix it.

## Parameter ‘Agresso SQL (asqlflag)’

If your query is written in ASQL then change the ‘Default value’ to a 1.

## New Parameter 'ITK batch start (itk_start)'

This parameter is mandatory and must be set as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | "ConvFsr" |
| **Length** | "30" |
| **#** | *On a report variant this must be greater than or equal to "100", on a user defined report this can be the next available number.* |
| **Fixed** | ☒ |

## New Parameter 'File names in (name_in)'

This parameter is used to define the number and names of the FSR files that are to be converted into TSV files.

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | A comma separated list of the names of the parameters containing the file names to convert ("query_param1" in the example). Spaces on either side of each comma are ignored. |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'FF on separate line (ff_separate)'

This parameter is mandatory and a value of "Y" means that the form-feeds in the files being processed are on lines of their own, a value of "N" means that the form-feeds in the files being processed are the first character of lines containing other data. This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | • *If all files in 'name_in' need the same 'ff_separate' value then set 'Default value' to that single value.*<br>• *Otherwise set 'Default value' to a comma separated list of values, each value corresponds to the file in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

Note that in the screen shot this parameter has been defined as Data type “b” so that it appears as a checkbox, this is acceptable because there is only one file being processed.

## New Parameter ‘No of header records (num_headers)’

This parameter is used to specify whether the FSR files have any header records and if they do how many.

This parameter is optional so if none of the files have header records it can either be omitted or it can be included and have its value set to zero. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | “a” |
| **Default value** | • *If all files in ‘name_in’ have the same number of header records then set ‘Default value’ to that single number.*<br>• *Otherwise set ‘Default value’ to a comma separated list of numbers of header records, each value corresponds to the file in the same position in ‘name_in’ and there must be the same number of values as there are in ‘name_in’.* |
| **Length** | *Greater than or equal to the length of the ‘Default value’ up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter ‘No of footer records (num_footers)’

This parameter is used to specify whether the FSR files have any footer records and if they do how many.

This parameter is optional so if none of the files have footer records it can either be omitted or it can be included and have its value set to zero. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | “a” |
| **Default value** | • *If all files in ‘name_in’ have the same number of footer records then set ‘Default value’ to that single number.*<br>• *Otherwise set ‘Default value’ to a comma separated list of numbers of footer records, each value corresponds to the file in the same position in ‘name_in’ and there must be the same number of values as there are in ‘name_in’.* |
| **Length** | *Greater than or equal to the length of the ‘Default value’ up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Remove non-ASCII chars (asc_only)'

This parameter is used to control whether any non-ASCII characters in the FSR files should be transferred to the TSV files or not.

Note that the form-feed character is an ASCII character but will not be transferred to the TSV file.

This parameter is optional so if all of the files whose names are in the 'name_in' parameter can only contain characters in the ASCII character set (see section 5.2) or there could be non-ASCII characters and you do not want them removed then this parameter can either be omitted or it can be included and have its value set to "N". If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "A" |
| **Default value** | • *If all the files in 'name_in' could contain non-ASCII characters and you want them removed then set 'Default value' to "*Y*".*<br>• *If all of the files in 'name_in' can only contain ASCII characters or they could contain non-ASCII characters but you don't want them removed then set 'Default value' to "*N*".*<br>• *Otherwise set 'Default value' to a comma separated list of "*Y*"s and "*N*"s, , each value corresponds to the file in the same position in 'name_in' and there must be the same number of values as there are in 'name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

### 3.4.3 Example File and Query

As an example here is one of the files and queries used during testing; it posts a simple GL journal and relates to the report variant screen shot above. The file header contains a date to be used as the transaction date and the footer contains a total-debit figure that the query will "throw" an error on if it is incorrect

## The File

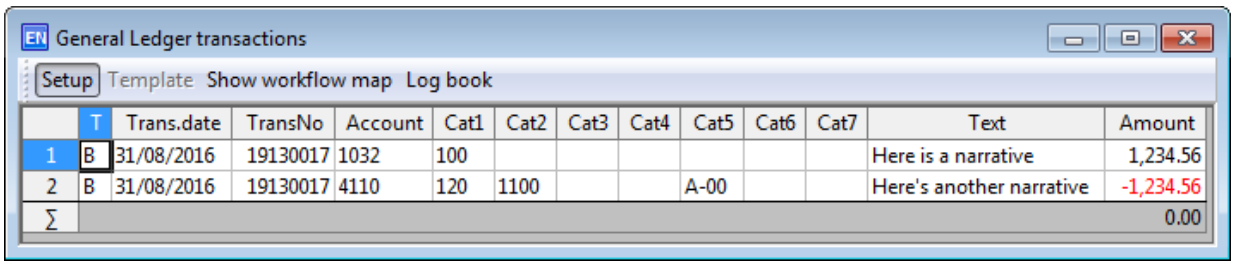
| Line | Contents |
|---|---|
| 1 | 31/08/2016 |
| 2 | $^{F}_{F}$ |
| 3 | 1032 |
| 4 | 100 |
| 5 | |
| 6 | |
| 7 | |
| 8 | |

| Line | Contents |
|---|---|
| 9 | |
| 10 | |
| 11 | 1234.56 |
| 12 | Here is a narrative |
| 13 | $^{F}_{F}$ |
| 14 | 4100 |
| 15 | 120 |
| 16 | 1100 |
| 17 | |
| 18 | |
| 19 | A-00 |
| 20 | |
| 21 | |
| 22 | -1234.56 |
| 23 | Here's another narrative |
| 24 | $^{F}_{F}$ |
| 25 | 1234.56 |

## The SQL Query

See section 4.2.

## The Results




| | T | Trans.date | TransNo | Account | Cat1 | Cat2 | Cat3 | Cat4 | Cat5 | Cat6 | Cat7 | Text | Amount |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | B | 31/08/2016 | 19130017 | 1032 | 100 | | | | | | | Here is a narrative | 1,234.56 |
| 2 | B | 31/08/2016 | 19130017 | 4110 | 120 | 1100 | | | A-00 | | | Here's another narrative | -1,234.56 |
| Σ | | | | | | | | | | | | | 0.00 |

# 3.5 Remove Import Files as a Process Ends (DelFile or MovFile)

## 3.5.1 Introduction

When you are using an SQL query to read a file into an import routine the name of the file(s) are usually held in one or more of the 'query_param*n*' process parameters. These files unlike ones whose names are stored in process parameters such as 'file_name' are not removed from the 'Data Import' folder at the end of the run, so it is possible for a user to process the same file(s) twice.

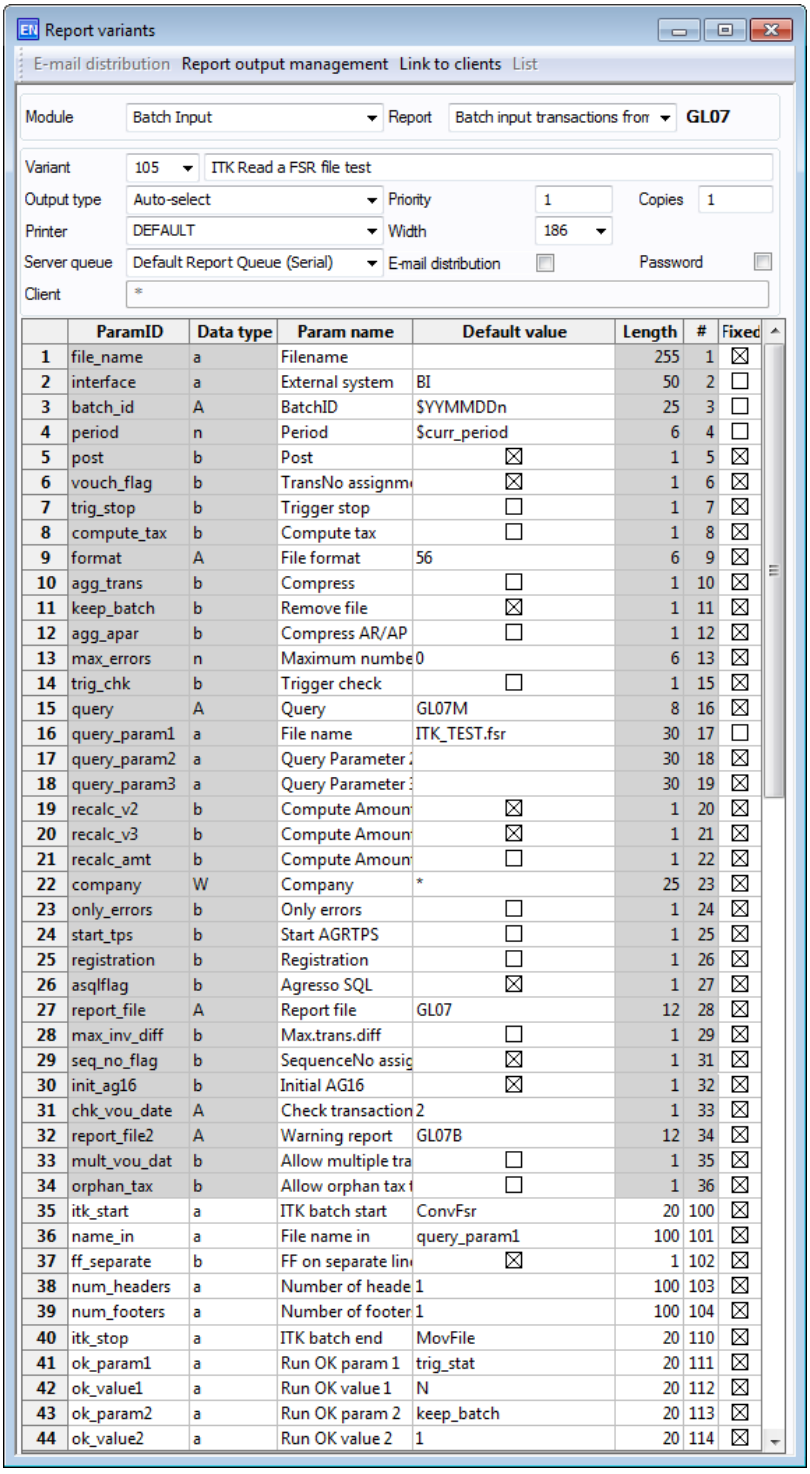
These methods allow you to either delete the file(s) or move them to the Business World "Scratch" folder and are typically used in combination with the 'BiQuery', 'ConvCsv', 'ConvCfr', 'ConvFsr' and 'ConvXls' methods.

**WARNING**

You cannot use the 'MovFile' method in a SaaS environment.

## 3.5.2 Set Up

In order to use one these methods on a process you must create a variant with some new parameters added or add these parameters to your 'User defined report'. The example is for a GL07 variant that also uses 'ConvFsr' method and moves the file to scratch but the same principle applies to deleting the files and to any other server process.




| | ParamID | Data type | Param name | Default value | Length | # | Fixed |
|---|---|---|---|---|---|---|---|
| 1 | file_name | a | Filename | | 255 | 1 | ☒ |
| 2 | interface | a | External system | BI | 50 | 2 | ☐ |
| 3 | batch_id | A | BatchID | $YYMMDDn | 25 | 3 | ☐ |
| 4 | period | n | Period | $curr_period | 6 | 4 | ☐ |
| 5 | post | b | Post | ☒ | 1 | 5 | ☒ |
| 6 | vouch_flag | b | TransNo assignm | ☒ | 1 | 6 | ☒ |
| 7 | trig_stop | b | Trigger stop | ☐ | 1 | 7 | ☒ |
| 8 | compute_tax | b | Compute tax | ☐ | 1 | 8 | ☒ |
| 9 | format | A | File format | 56 | 6 | 9 | ☒ |
| 10 | agg_trans | b | Compress | ☐ | 1 | 10 | ☒ |
| 11 | keep_batch | b | Remove file | ☒ | 1 | 11 | ☒ |
| 12 | agg_apar | b | Compress AR/AP | ☐ | 1 | 12 | ☒ |
| 13 | max_errors | n | Maximum numbe | 0 | 6 | 13 | ☒ |
| 14 | trig_chk | b | Trigger check | ☐ | 1 | 15 | ☒ |
| 15 | query | A | Query | GL07M | 8 | 16 | ☒ |
| 16 | query_param1 | a | File name | ITK_TEST.fsr | 30 | 17 | ☐ |
| 17 | query_param2 | a | Query Parameter | | 30 | 18 | ☒ |
| 18 | query_param3 | a | Query Parameter | | 30 | 19 | ☒ |
| 19 | recalc_v2 | b | Compute Amoun | ☒ | 1 | 20 | ☒ |
| 20 | recalc_v3 | b | Compute Amoun | ☒ | 1 | 21 | ☒ |
| 21 | recalc_amt | b | Compute Amoun | ☐ | 1 | 22 | ☒ |
| 22 | company | W | Company | * | 25 | 23 | ☒ |
| 23 | only_errors | b | Only errors | ☐ | 1 | 24 | ☒ |
| 24 | start_tps | b | Start AGRTPS | ☐ | 1 | 25 | ☒ |
| 25 | registration | b | Registration | ☐ | 1 | 26 | ☒ |
| 26 | asqlflag | b | Agresso SQL | ☒ | 1 | 27 | ☒ |
| 27 | report_file | A | Report file | GL07 | 12 | 28 | ☒ |
| 28 | max_inv_diff | b | Max.trans.diff | ☐ | 1 | 29 | ☒ |
| 29 | seq_no_flag | b | SequenceNo assig | ☒ | 1 | 31 | ☒ |
| 30 | init_ag16 | b | Initial AG16 | ☒ | 1 | 32 | ☒ |
| 31 | chk_vou_date | A | Check transaction | 2 | 1 | 33 | ☒ |
| 32 | report_file2 | A | Warning report | GL07B | 12 | 34 | ☒ |
| 33 | mult_vou_dat | b | Allow multiple tra | ☐ | 1 | 35 | ☒ |
| 34 | orphan_tax | b | Allow orphan tax t | ☐ | 1 | 36 | ☒ |
| 35 | itk_start | a | ITK batch start | ConvFsr | 20 | 100 | ☒ |
| 36 | name_in | a | File name in | query_param1 | 100 | 101 | ☒ |
| 37 | ff_separate | b | FF on separate lin | ☒ | 1 | 102 | ☒ |
| 38 | num_headers | a | Number of heade | 1 | 100 | 103 | ☒ |
| 39 | num_footers | a | Number of footer | 1 | 100 | 104 | ☒ |
| 40 | itk_stop | a | ITK batch end | MovFile | 20 | 110 | ☒ |
| 41 | ok_param1 | a | Run OK param 1 | trig_stat | 20 | 111 | ☒ |
| 42 | ok_value1 | a | Run OK value 1 | N | 20 | 112 | ☒ |
| 43 | ok_param2 | a | Run OK param 2 | keep_batch | 20 | 113 | ☒ |
| 44 | ok_value2 | a | Run OK value 2 | 1 | 20 | 114 | ☒ |

## New Parameter 'ITK batch end (itk_stop)'

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | "DelFile" or "MovFile" |
| **Length** | "30" |

| Column | Value |
|---|---|
| **#** | *On a report variant this must be greater than or equal to "100", on a user defined report this can be the next available number.* |
| **Fixed** | ☒ |

## New Parameter 'File names in (name_in)'

This parameter is used to define the number and names of the files that are to be deleted ("DelFile") or moved to "scratch" ("MovFile").

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | *A comma separated list of the names of the parameters containing the file names to convert ("query_param1" in the example). Spaces on either side of each comma are ignored.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *Normally the next available number. But see below.* |
| **Fixed** | ☒ |

Note that in the above screenshot the 'name_in' parameter is being shared with the 'itk_start' "ConvFsr" method so:

- The 'MovFile' does not have its own copy of this parameter.
- The file that will be moved to scratch is the TSV file created by the "ConvFsr" method.

## New Parameters 'Run OK param *n* (ok_param*n*)' and 'Run OK value *n* (ok_value*n*)'

By default 'DelFile' will delete the files and 'MovFile' move the files to "scratch" regardless of whether or not the process to which the method is linked runs without errors. If the process is a data import and it fails and the input file will be deleted then you may not be able to re-process the data in that file. Some server processes set a one or more parameters to particular values to indicate that no errors have occurred, these parameters allow you to use these parameters to modify the behaviour of 'DelFile' and 'MovFile'.

These parameters are optional so if all of the files whose names are in the 'name_in' parameter should be deleted or moved to scratch regardless of whether the process works or not then omit all of these parameters. If these parameters are present they must be defined in pairs as follows:

Parameter one of two:

| Column | Value |
|---|---|
| **Param ID** | "ok_param"*n* |
| **Data type** | "a" |
| **Default value** | *The name of the parameter whose value is to be tested against the 'ok_valuen' parameter with the same number.* |

| Column | Value |
|---|---|
| **Length** | "20" |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

Parameter two of two:

| Column | Value |
|---|---|
| **Param ID** | "ok_value"*n* |
| **Data type** | "a" |
| **Default value** | *The value of the parameter whose name is in the 'ok_paramn' parameter with the same number that indicates that the run was successful.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

Up to five pairs of parameters can be defined: 'ok_param1' and 'ok_value1' to 'ok_param5' and 'ok_value5'. All the defined parameters must have their specified values in order for the files to be deleted or moved.

Processes that set "run OK" type parameters include:

| Process | ok_param1 | ok_value1 | ok_param2 | ok_value2 |
|---|---|---|---|---|
| GL07 | trig_stat | N | keep_batch | 1 |
| SRP01 | error# | 0 (zero) | only_apar | N |

# 3.6 Blank Columns and Remove Addresses and Relation Values (FldBlnk)

## 3.6.1 Introduction

Some of the standard "master file" import routines allow you to add and change address and relation values but do not allow you to remove them. In addition some of these same routines will not let you blank out columns on existing rows.

'FldBlnk' will blank out specified columns (see "Table and column configuration" below) if they have been set to a particular character that means "delete me" and remove relation values set to the same character and addresses whose 'address' column has been set to that value.

'FldBlnk' is currently available for use on the following server processes:

| Process | Standard Menu Text |
|---|---|
| CS15 | Batch input customer and supplier information |
| PR43 | Update employees/resources |

**WARNING**

**Method 'FldBlnk' does not support process 'DataLoad' in the current version of ITK.**

**WARNING**

If you wish to blank or remove a column that is validated (such as an attribute value) then you must ensure that the "delete me" character is a valid value in that column; for example to remove a relation value the "delete me" character must be defined as a valid value of that attribute.

## 3.6.2 Set Up

### 3.6.2.1 Table and Column Configuration, Introduction

When amending existing data both CS15 and PR43 interpret a blank value in the import file to mean "Do not change the value of this column", so neither process contains logic to stop mandatory columns from being blanked out. Because the list of mandatory columns can vary from release to release of Business World it would not be advisable to hardwire such logic into 'FldBlnk' therefore a user-definable way of controlling which columns it is valid to blank out is needed.

This control is provided either by a set of system parameters or by an "INI File". In a SaaS environment only the system parameters can be used, in other environments either mechanism can be used; and a process parameter defines which one to use.

UNIT4 recommends that the process parameter mechanism is used because:

- If you decide at some future date to migrate to SaaS then you will be forced to use the process parameters at that time anyway.
- The changing of the system parameters only has to be done once no matter how many server queues you have.

Both mechanisms share several concepts:

- Primary Table
  The main table that the import routine is going to use:
- Primary Column Setting
  A column on the primary table that can be left blank with an indication of whether you wish 'FldBlnk' to blank out this column's value when it is set to the "delete me" character.
- Secondary Table
  A table that can also be updated by the process that has rows that are linked to a row on the primary table, for instance a customer's address.
- Secondary Table Key
  The column on the primary and secondary tables that create a link from one to the other
- Secondary Column Setting
  A column on the secondary table that can be left blank with an indication of whether you wish 'FldBlnk' to blank out this column's value when it is set to the "delete me" character.

The supported primary and secondary tables are:

| Process | Primary table | Secondary Tables |
|---|---|---|
| CS15 | acsheaderinput | agladdress |
| | | aglrelvalue |
| PR43 | ahsresources | asuheader |
| | | agladdress |

### 3.6.2.2 Table and Column Configuration Using the System Parameters

In the system parameters the concepts are represented by parameters with particular names:

| Concept | Parameter Name | On/off |
|---|---|---|
| Primary Table | ITK_FB_P*pp*_0_NAME | Must be checked |
| Primary Column Setting | ITK_FB_P*pp*_2_C*cc* | Check this column if can be blanked out. |
| Secondary Table | ITK_FB_P*pp*_S*ss*_0_NAME | Must be checked |
| Secondary Table Key | ITK_FB_P*pp*_S*ss*_1_KEYS | Must be checked |
| Secondary Column Setting | ITK_FB_P*pp*_S*ss*_2_C*cc* | Check this column if can be blanked out. |

These parameters should be modified using the (**Agresso )Common ▶ System setup ▶ System parameters** screen (AG07). The somewhat contrived naming scheme for these parameters has been adopted so that they appear in a sensible order in the AG07 screen.

**ITK_FB_P*pp*_0_NAME**

Where "***pp***" is a two digit number that identifies the primary table, i.e. "acsheaderinput" is "01" and "ahsresources" is "02".

The value of this parameter is the name of the primary table followed by the number of 'ITK_FB_P*pp*_2_C*cc*' parameters that belong to it, separated by a hyphen, e.g. "acsheaderinput-16".

**ITK_FB_P*pp*_2_C*cc***

Where "***pp***" is the same as in the 'ITK_FB_P*pp*_0_NAME' parameter and "***cc***" is a two digit column number starting at "01" and continuing to the number in the second part of the 'ITK_FB_P*pp*_NAME' parameter's value, e.g. 'ITK_P01_2_C01' to 'ITK_P01_2_C16'.

The value of this parameter is the name of the column on the primary table.

If it is checked then this column's value can be blanked out, if it must not be blanked out then leave it unchecked.

**ITK_FB_P*pp*_S*ss*_0_NAME**

Where "***pp***" is the same as in the 'ITK_FB_P*pp*_0_NAME' parameter and "***ss***" is a number identifying the secondary table.

The value of this parameter is the name of the secondary table followed by the number of 'ITK_FB_P*pp*_S*ss*_2_C*cc*' parameters that belong to it, separated by a hyphen, e.g. "agladdress-15".

**ITK_FB_P*pp*_S*ss*_1_KEYS**

Where "***pp***" and "**ss**" are the same as in the 'ITK_FB_P*pp*_S*ss*_0_NAME' parameter.
The value of this parameter is either:

- A single column name if the secondary table's key column is also a column on the primary table and holds the same value.
- If the column names are different then the name of the secondary table's key column, a hyphen and then name of the column on the primary table that holds the same value, e.g. "dim_value-apar_id".

**ITK_FB_P*pp*_S*ss*_2_C*cc***

Where "***pp***" and "**ss**" are the same as in the 'ITK_FB_P*pp*_S*ss*_0_NAME' parameter and "***cc***" a two digit column number starting at "01" and continuing to the number in the second part of the 'ITK_FB_P*pp*_S*ss*_NAME' parameter's value, e.g. 'ITK_P01_S01_2_C01' to 'ITK_P01_S01_2_C15'.

The value of this parameter is the name of the column on the primary table.

If it is checked then this column's value can be blanked out, if it must not be blanked out then leave it unchecked.

An example follows:

| Name | Mod | Max Length | Value | On/off | Sys. setup | Client level |
|---|---|---|---|---|---|---|
| ITK_FB_P01_0_NAME | 00 | 25 | acsheaderinput-16 | ☒ | ☒ | ☐ |
| ITK_FB_P01_2_C01 | 00 | 25 | apar_id_ref | ☐ | ☒ | ☐ |
| ITK_FB_P01_2_C02 | 00 | 25 | clearing_code | ☒ | ☒ | ☒ |
| | | | ... | | | |
| ITK_FB_P01_2_C15 | 00 | 25 | tax_system | ☐ | ☒ | ☐ |
| ITK_FB_P01_2_C16 | 00 | 25 | vat_reg_no | ☐ | ☒ | ☐ |
| ITK_FB_P01_S01_0_NAME | 00 | 25 | agladdress-15 | ☐ | ☒ | ☐ |
| ITK_FB_P01_S01_1_KEYS | 00 | 25 | dim_value-apar_id | ☐ | ☒ | ☐ |
| ITK_FB_P01_S01_2_C01 | 00 | 25 | e_mail | ☐ | ☒ | ☐ |

| Name | Mod | Max Length | Value | On/off | Sys. setup | Client level |
|---|---|---|---|---|---|---|
| ITK_FB_P01_S01_2_C02 | 00 | 25 | e_mail_cc | ☐ | ☒ | ☐ |
| … | | | | | | |
| ITK_FB_P01_S01_2_C14 | 00 | 25 | url_path | ☐ | ☒ | ☐ |
| ITK_FB_P01_S01_2_C15 | 00 | 25 | zip_code | ☐ | ☒ | ☐ |
| ITK_FB_P01_S02_0_NAME | 00 | 25 | aglrelvalue-1 | ☒ | ☒ | ☐ |
| ITK_FB_P01_S02_1_KEYS | 00 | 25 | att_value-apar_id | ☒ | ☒ | ☐ |
| ITK_FB_P01_S02_2_C01 | 00 | 25 | rel_value | ☒ | ☒ | ☒ |
| ITK_FB_P02_0_NAME | 00 | 25 | ahsresources-30 | ☐ | ☒ | ☐ |
| ITK_FB_P02_2_C01 | 00 | 25 | Citizenship | ☐ | ☒ | ☐ |
| … | | | | | | |

These settings would mean that either 'clearing_code' (a.k.a. 'Sort code') or 'rel_value' (a.k.a. 'Relation value') will be blanked if their value is set to the "delete me" character.

#### 3.6.2.3 Table and Column Configuration Using the "INI" File

This is the 'InterfaceTools.ini' file which can be found in the "*BW Bin*" folder on the Business-Server.

The path of the "*BW Bin*" folder will vary depending on which version of Business World you are running and on versions where it is provided whether the processes will be running on a 32bit or 64bit server queue.

| Version Name | Version No | Queue | Path |
|---|---|---|---|
| Milestone 3 | 5.6.3 | 32bit | *x*:\Program Files (x86)\Agresso Route 66\Bin |
| Milestone 4 | 5.7.1 | 32bit | *x*:\Program Files (x86)\Agresso 5.7.1\Bin |
| Milestone 5 | 5.7.2 | 32bit | *x*:\Program Files (x86)\Agresso 5.7.2\Bin |
| | | 64bit | *x*:\Program Files\Agresso 5.7.2\Bin |
| Milestone 6 | 6.0.0 | 32bit | *x*:\Program Files (x86)\UNIT4 Business World M6\Bin |
| | | 64bit | *x*:\Program Files\UNIT4 Business World M6\Bin |

Where *x* is the letter of the drive where Business World has been installed (usually the "C" drive).

In the unlikely event that your Business-Server is not running a 64bit version of Microsoft Windows then:

- 64bit server queues are not possible.
- There will not be a "Program Files (x86)" folder, only a "Program Files" folder.

**N.B.** If you wish to run 'FldBlnk' on both 32bit and 64bit server queues then you will need to edit 'InterfaceTools.ini' to both Bin folders.

Each primary table has a section in the file: e.g. "[acsheaderinput]" and "[ahsresources]".

Each primary column setting is a line in the section, the line consists of the name of the column followed by either:

- "=Y" which means that FldBlnk should look at this column and blank it if its value is the "delete me" character

- "=N" which means that FldBlnk should not look at this column.

For example:

```
[acsheaderinput]
apar_id_ref=N
bonus_gr=N
clearing_code=N
comp_reg_no=N
control=N
description=N
disc_code=N
factor_short=N
foreign_acc=Y
invoice_code=N
message_text=Y
pay_temp_id=N
reference_1=N
swift=N
tax_system=N
vat_reg_no=N
```

These settings would mean that either 'foreign_acc' (a.k.a. 'Account name') or 'message_text' (a.k.a. 'Message') will be blanked if their value is set to the "delete me" character.

After all of the primary column settings a number of repeating groups can appear. Each group represents one of the secondary tables and consists of a key line followed by one or more secondary column setting lines which have the same format and purpose as the primary column settings, except that the name of the column has a prefix consisting of the secondary table name followed by a full-stop.

Key lines have the following format:

"**key.***table_name***=***key_details*"

Where *table_name* is a secondary table (e.g. "agladdress"),

*key_details* is either:

- A single column name if the secondary table's key column is also a column on the primary table and holds the same value
- If the column names are different then the name of the secondary table's key column, a hyphen and then name of the column on the primary table that holds the same value.

For example:

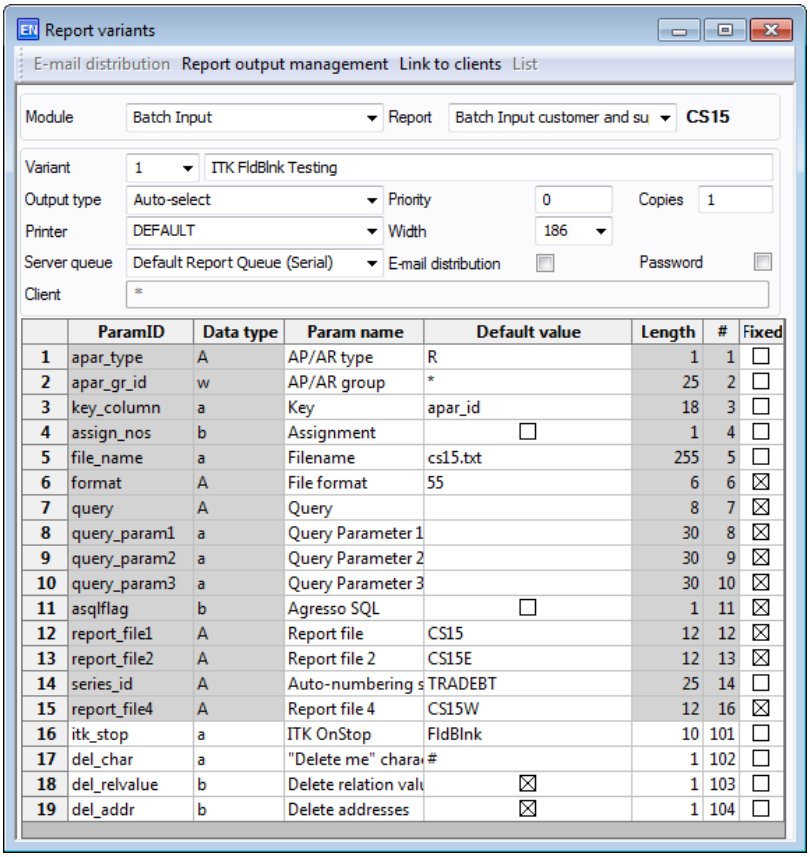
```
keys.agladdress=dim_value-apar_id
agladdress.e_mail=Y
agladdress.e_mail_cc=Y
agladdress.ean=Y
agladdress.place=N
agladdress.pos_title=Y
agladdress.province=N
agladdress.telephone_1=N
agladdress.telephone_2=Y
agladdress.telephone_3=Y
agladdress.telephone_4=Y
agladdress.telephone_5=Y
agladdress.telephone_6=Y
agladdress.telephone_7=Y
agladdress.url_path=Y
agladdress.zip_code=N
```

The settings in this example mean that the 'apar_id' (CUST.ID or SUPP.ID) is the key to the address and that all fields can be blanked except 'place', 'province', 'telephone_1' and 'zip_code'.

In order to customise the "INI file" for your installation use a text editor such as 'Notepad' to change the "=N" to ="Y" on all appropriate fields, DO NOT change any of the lines that start "keys…".

#### 3.6.2.4 The Process Parameters

In order to use 'FldBlnk' on a process you must create a variant with some new parameters added or add these parameters to your 'User defined report'. The example is for the 'AGRESSO Financials ► Batch input' process "Batch input customer and supplier information" but the same principle applies for PR43.




| | ParamID | Data type | Param name | Default value | Length | # | Fixed |
|---|---|---|---|---|---|---|---|
| 1 | apar_type | A | AP/AR type | R | 1 | 1 | ☐ |
| 2 | apar_gr_id | w | AP/AR group | * | 25 | 2 | ☐ |
| 3 | key_column | a | Key | apar_id | 18 | 3 | ☐ |
| 4 | assign_nos | b | Assignment | ☐ | 1 | 4 | ☐ |
| 5 | file_name | a | Filename | cs15.txt | 255 | 5 | ☐ |
| 6 | format | A | File format | 55 | 6 | 6 | ☒ |
| 7 | query | A | Query | | 8 | 7 | ☒ |
| 8 | query_param1 | a | Query Parameter 1 | | 30 | 8 | ☒ |
| 9 | query_param2 | a | Query Parameter 2 | | 30 | 9 | ☒ |
| 10 | query_param3 | a | Query Parameter 3 | | 30 | 10 | ☒ |
| 11 | asqlflag | b | Agresso SQL | ☐ | 1 | 11 | ☒ |
| 12 | report_file1 | A | Report file | CS15 | 12 | 12 | ☒ |
| 13 | report_file2 | A | Report file 2 | CS15E | 12 | 13 | ☒ |
| 14 | series_id | A | Auto-numbering s | TRADEBT | 25 | 14 | ☐ |
| 15 | report_file4 | A | Report file 4 | CS15W | 12 | 16 | ☒ |
| 16 | itk_stop | a | ITK OnStop | FldBlnk | 10 | 101 | ☐ |
| 17 | del_char | a | "Delete me" chara | # | 1 | 102 | ☐ |
| 18 | del_relvalue | b | Delete relation val | ☒ | 1 | 103 | ☐ |
| 19 | del_addr | b | Delete addresses | ☒ | 1 | 104 | ☐ |

**WARNING**

When using 'FldBlnk' on PR43:

- Make sure that you modify a variant of "Update employees/resources" not "Batch input of employees/resources", because both processes are "PR43".
- If the 'Start (func_name)' parameter is set to "CS15" then
  - Modify the 'Variant (variant)' parameter to use a CS15 variant that also has 'FldBlnk' linked to it.
  - The CS15 variant's 'del_char' parameter must have the same value as the PR43 variant's 'del_char' parameter.

## New Parameter 'ITK OnStop (itk_stop{_*n*})'

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | "FldBlnk" |
| **Length** | "30" |
| **#** | *On a report variant this must be greater than or equal to "100", on a user defined report this can be the next available number.* |
| **Fixed** | ☒ |

## New Parameter '"Delete me" character (del_char{_*n*})'

This parameter defines the character that you have set column values to in order to indicate that you wish the value to be blanked or the relation or address to be deleted.

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | *The "delete me" character that you are setting columns, relation values and addresses to.* |
| **Length** | "1" |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Delete relation values (del_relvalue or del_rval_*n*)'

This parameter defines whether relations values should be blanked or deleted if they are set to the "delete me" character.

On CS15 this parameter is optional, if you wish relation values to be blanked then then this parameter can either be omitted or it can be included and have its value set to unchecked. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "b" |
| **Default value** | ☒ *On CS15 if you want to delete relation values whose 'rel_value' column is set to the "delete me" character.*<br>☐ *Otherwise.* |
| **Length** | "1" |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

**WARNING**

PR43 does not allow the maintenance of relation values and so on that process this parameter must be omitted or left unchecked.

## New Parameter 'Delete addresses (del_addr{_n})'

- This parameter defines whether relations values should be blanked or deleted if they are set to the "delete me" character.
- This parameter is optional, if you wish address columns to be blanked then then this parameter can either be omitted or it can be included and have its value set to unchecked and you must also ensure that the appropriate 'agladdress' lines in the "INI file" meet your requirements. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "b" |
| **Default value** | ☒ *If you want to delete addresses whose 'address' column is set to the "delete me" character.*<br>☐ *Otherwise.* |
| **Length** | "1" |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

### New Parameter 'Get config from sysparams (conf_par{_*n*})'

This parameter defines whether the "Table and column configuration" should be retrieved from the system parameters or the INI file.

This parameter is optional,

- In a SaaS environment it should be omitted and 'FldBlnk' will attempt to use the system parameters.
- If other environments if it omitted it will be treated as unchecked and 'FldBlnk' will attempt to use the INI file.

If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "b" |
| **Default value** | ☒ *The "Table and column configuration" will be retrieved from the system parameters.*<br>☐ *The "Table and column configuration" will be retrieved from the INI file.* |
| **Length** | "1" |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

**WARNING**

If you include an unchecked 'conf_par' parameter in a SaaS environment then 'FldBlnk' will stop with an error.

## 3.6.3 Summary of the Use of Multiple Parameters

'FldBlnk' fully supports multiple 'itk_stop' parameters, but because of technical restrictions when there is an explicit sequence number some of the parameter names are different as follows:

| itk_stop | itk_stop_2 | itk_stop_3 | … | itk_stop_20 |
|---|---|---|---|---|
| del_char | del_char_2 | del_char_3 | … | del_char_20 |
| del_relvalue | del_rval_2 | del_rval_3 | … | del_rval_20 |
| del_addr | del_addr_2 | del_addr_3 | … | del_addr_20 |
| conf_par | conf_par_2 | conf_par_3 | … | conf_par_20 |

## 3.6.4 Technical Note

Due to limitations of the PR43 process 'FldBlnk' works in a slightly different way in that process to the way it does in CS15:

- In CS15:
  - It scans each of the appropriate tables to see if any rows have been updated in the current client by the user who ordered this process since this run of the process started.
- In PR43:
  - It scans the primary table (ahsresources) to see if any rows have been updated in the current client since this run of the process started and stores the information about these rows in a help-table.

- It then scans the rows in each of the secondary tables that are linked to the help-table.

- In both CS15 and PR43:
  - If it is blanking rows on that table then:
    - On each scanned row it checks for a “delete me” character in each of the columns whose “INI File” entry is set to “Y” and blanks out any such columns found.
  - If it is deleting rows on that table (see the ‘del_relvalue’ or ‘del_addr’ parameters) then:
    - On each scanned row it checks for a “delete me” character in the appropriate column and deletes the row if it finds one.

---

**WARNING**

**The ITKv2 function ‘FldBlnkPR43’ (a version of the ITKv2 ‘FldBlnk’ function that was especially for PR43) will not be implemented in ITKv3.**

# 3.7 Remove Output Files as a Process Starts (DelFile)

## 3.7.1 Introduction

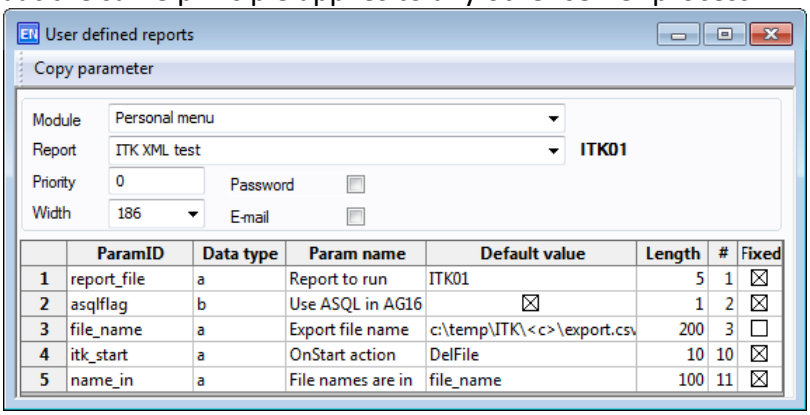
When you are using an SQL query to copy Business World data into one or more export files the query may fail if any of the files already exist in the destination folder.

'DelFile' allows you to delete the file(s) before the query runs:

- The names of the files to be deleted must be held in one or more process parameters.

## 3.7.2 Set Up

In order to use 'DelFile' on a process you must create a variant with some new parameters added or add these parameters to your 'User defined report'. The example is for a bespoke export routine but the same principle applies to any other server process.




| | ParamID | Data type | Param name | Default value | Length | # | Fixed |
|---|---|---|---|---|---|---|---|
| 1 | report_file | a | Report to run | ITK01 | 5 | 1 | ☒ |
| 2 | asqlflag | b | Use ASQL in AG16 | ☒ | 1 | 2 | ☒ |
| 3 | file_name | a | Export file name | c:\temp\ITK\<c>\export.csv | 200 | 3 | ☐ |
| 4 | itk_start | a | OnStart action | DelFile | 10 | 10 | ☒ |
| 5 | name_in | a | File names are in | file_name | 100 | 11 | ☒ |

### New Parameter 'OnStart action (itk_start)'

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | "DelFile" |
| **Length** | "30" |
| **#** | *On a report variant this must be greater than or equal to "100", on a user defined report this can be the next available number.* |
| **Fixed** | ☒ |

### New Parameter 'File names in (name_in)'

This parameter is used to define the number and names of the files that are to be deleted.

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | *A comma separated list of the names of the parameters containing the file names to delete ("query_param1" in the example). Spaces on either side of each comma are ignored.* |

| Column | Value |
|---|---|
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

The 'Default value' of each of the parameters referred to in the 'Default value' of the 'name_in' parameter determines a file path and name that you want to delete:

- The value entered can include fixed text and one or more of the file tags described in section 4.3. The 'Default value' of each of these parameters is then changed so that your query can use the modified name.
- If the value entered is not a file path then 'DelFile' assumes that the file is in the folder defined by the "AGRESSO_EXPORT" environment variable.
- If the value entered is a file path then that path will be used.

## 3.8 Copy or Move Output Files as a Process Ends (Export)

### 3.8.1 Introduction

Files created with ASQL's "COPY TO" command can be placed anywhere on the file system using the "FILE=" parameter, with or without the "EXPORT" modifier. However files created by "COPY TO" have two restrictions:

- All records must have the same format
- The fields in a records must be either fixed width or delimited with one of the following characters:
  - Tab (ASCII[12] code 9)
  - "@" (at sign)
  - "£" (sterling sign)
  - "$" (dollar sign)
  - "%" (percentage symbol)
  - "&" (ampersand)
  - "." (full stop)
  - or ";" (semicolon)

  Using the Tab character is the safest option because this is a non-printing character and is therefore highly unlikely to appear in the data.

  It is possible to produce a CSV file using "COPY TO" as long as none of the text fields contain apostrophes by producing records containing a single field and concatenating all the data into that one field with commas separating each field and apostrophes around all text fields. E.g.

```
COPY TO EXPORT
       FILE = datafile.csv,
       COLSEP = T,
       SELECT CONCAT('''', CONCAT(client, CONCAT(''',', TO_CHAR(amount))))
```

Often we need to create more complex files with multiple record types and the easiest way to do this is to write an ARW to create the file. However the output of an ARW is saved in the 'Report Results' folder and by its name is indistinguishable from an "ordinary" report i.e. it has the form "*xxxxxy_n*.lis" where:

- *xxxxx* is the process code
- *y* is a sequence letter (a for the first report, b for the second, etc.) and
- *n* is the order number of the run

In order to transmit the file to another system it often needs to be stored elsewhere and often with a specific name format. ARW has an '.OUTPUT' command that allows you to specify the file name (and optional path) that your report's output be written to instead of the above "lis" file but you can only use a fixed file name: e.g. "C:\temp\myoutput.txt" with no variable elements, not even the order number of the run.

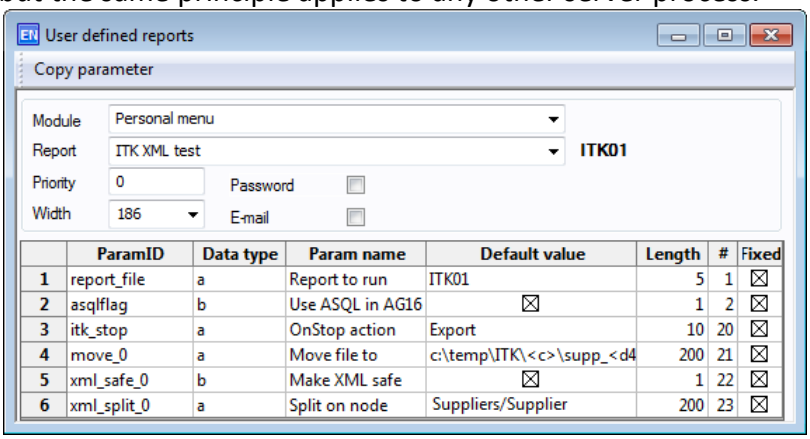
'Export' provides a more flexible solution to the above problem than the ARW '.OUTPUT' command:

[12] *See section 5.2.*

- Variables can be included anywhere in the file path or name
- The file name is a process parameter instead of hard-coded in the report
- If the file name does not include a path then it will be written to the folder defined by the "AGRESSO_EXPORT" environment variable.
- In a SaaS environment the file must be written to the folder defined by the "AGRESSO_EXPORT" environment variable or a folder within that folder.

### 3.8.2 Set Up

In order to use 'Export' on a process you must create a variant with some new parameters added or add these parameters to your 'User defined report'. The example is for bespoke export routine but the same principle applies to any other server process.




| | ParamID | Data type | Param name | Default value | Length | # | Fixed |
|---|---|---|---|---|---|---|---|
| 1 | report_file | a | Report to run | ITK01 | 5 | 1 | ☒ |
| 2 | asqlflag | b | Use ASQL in AG16 | ☒ | 1 | 2 | ☒ |
| 3 | itk_stop | a | OnStop action | Export | 10 | 20 | ☒ |
| 4 | move_0 | a | Move file to | c:\temp\ITK\<c>\supp_<d4 | 200 | 21 | ☒ |
| 5 | xml_safe_0 | b | Make XML safe | ☒ | 1 | 22 | ☒ |
| 6 | xml_split_0 | a | Split on node | Suppliers/Supplier | 200 | 23 | ☒ |

#### New Parameter 'OnStop action (itk_stop)'

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | "Export" |
| **Length** | "30" |
| **#** | *On a report variant this must be greater than or equal to "100", on a user defined report this can be the next available number.* |
| **Fixed** | ☒ |

#### New Parameter 'Copy *n*th file to (copy_*n*)' or 'Move *n*th file to ('move_*n*')

This new parameter is an example of a group of parameters that you can set up to either copy or move one of the process' reports to another location and to give it a new name. All of these parameters have names in the form 'copy_*n*' or 'move_*n*' where *n* is the sequence number of the report file you wish to copy or move. The sequence number of the report file is shown in the '#' column in the **Report printout** screen (CR43). Sequence numbers greater than ten are not currently supported:

- Note that the first report output's sequence number is zero not one.
- If the process has multiple outputs you can leave gaps in the copy_*n* or move_*n* sequence; if you only want to copy the second output file then only define copy_1.
- You can mix 'copy' and 'move' parameters on the same variant as long as they are for different sequence numbers.

At least one of these parameters must be present and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | *The name (or path) of the file name that you wish this report output to copied or moved to.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

- The 'Default value' entered can include fixed text and one or more of the file tags described in section 4.3.
- If the 'Default value' entered is not a file path then the file will be written to the folder defined by the "AGRESSO_EXPORT" environment variable.
- If the 'Default value' entered is a file path then that path will be used the name including the file name. If the folder structure in the file name does not exist then it will be created.
- In order to stop 'Export' from causing serious damage to your system the 'Default value' is not allowed to end with ".dll", ".exe" or ".ocx"

If the expanded file path and name equals the original file path and name then this is only allowed if at least one of the follow in true:

- The corresponding 'asc_only_*n*' is equal to "1"
- The corresponding 'xml_safe_*n*' parameter is equal to "1"
- The corresponding 'xml_split_*n*' parameter is not blank.

**WARNING**

The numeric suffix on this parameter and its linked 'asc_only_*n*', 'blnk_lin_*n*', 'xml_safe_*n*' and 'xml_split_*n*' parameters has nothing to do with the mechanism for defining multiple 'itk_stop' parameters, it is the way you link this action to a particular output of the process.

## New Parameter 'Remove non-ASCII chars (asc_only_*n*)'

This parameter is used to control whether any non-ASCII characters in the output file should be retained in the moved and/or renamed file or not.

This parameter is optional so if the file whose name is in the 'copy_*n*' or 'move_*n*' parameter with the same sequence number can only contain characters in the ASCII character set (see section 5.2) or there could be non-ASCII characters and you do not want them removed then this parameter can either be omitted or it can be included and have its value set to unchecked. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "b" |
| **Default value** | ☒ *If the file whose name is in the 'copy_n' or 'move_n' parameter with the same sequence number could contain non-ASCII characters and you want them removed.*<br>☐ *Otherwise.* |
| **Length** | "1" |

| Column | Value |
|---|---|
| **#** | *The next available number.* |
| **Fixed** | ☒ |

You can have an 'asc_only_*n*' parameter for each 'copy_*n*' or 'move_*n*' that is present and its number must match

## New Parameter 'Allow blank lines (blnk_lin_*n*)'

This parameter is used to control whether any blank lines in the output file should be retained in the moved and/or renamed file or not.

This parameter is optional so if the file whose name is in the 'copy_*n*' or 'move_*n*' parameter with the same sequence number cannot contain blank lines or it can contain them and you wish to retain them then this parameter can either be omitted or have its value set to checked. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "b" |
| **Default value** | ☒ *If the file whose name is in the 'copy_n' or 'move_n' parameter with the same sequence number cannot contain blank lines or it could contain blank lines but you don't want them removed.*<br>☐ *Otherwise.* |
| **Length** | "1" |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

You can have a 'blnk_lin_*n*' parameter for each 'copy_*n*' or 'move_*n*' that is present and its number must match

## New Parameter 'Make XML safe (xml_safe_*n*)'

Business World reports (particularly ARW reports) are sometimes used to create XML files. Some characters are not allowed in XML files because they would cause programs that read the XML to fail. These characters are known as "unsafe" characters and each one must be replaced by a string of characters called an "entity" to avoid this problem. The characters and their corresponding entities are as follows:

| Unsafe Character | Entity |
|---|---|
| & | &amp; |
| < | &lt; |
| > | &gt; |
| ' | &apos; |
| " | &quot; |

This parameter will instruct 'Export' to scan the moved and/or renamed file and replace any unsafe characters with their entity.

This parameter is optional so if the file whose name is in the 'copy_*n*' or 'move_*n*' parameter with the same sequence number is not an XML file or it is but it cannot contain any unsafe

characters then this parameter can either be omitted or set to unchecked. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "b" |
| **Default value** | ☒ *If the file whose name is in the 'copy_n' or 'move_n' parameter with the same sequence number is XML and may contain "unsafe" characters.*<br>☐ *Otherwise.* |
| **Length** | "1" |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

You can have an 'xml_safe_*n*' parameter for each 'copy_*n*' or 'move_*n*' that is present and its number must match

## New Parameter 'Split on node (xml_split_*n*)'

Business World reports (particularly ARW reports) are sometimes used to create XML files. If an XML file contains list of things (e.g. a list of suppliers or GL postings) then these will be represented by a "repeating node" (one node per thing) in the file.

By default 'Export' will keep all of these nodes in a single file just like the report that produced the original output file. It is possible that the system that reads the XML file may not be able to handle files containing "repeating nodes" and so this parameter allows you to override the default behaviour and put each occurrence of the "repeating node" into a separate file.

This parameter is optional so if the file whose name is in the 'copy_*n*' or 'move_*n*' parameter with the same sequence number is not an XML file or it is but you don't need to split it then this parameter can either be omitted or it can be included and its value set to blank. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | • *If the XML file whose name is in the 'copy_n' or 'move_n' parameter with the same sequence number contains a repeating node and you wish each occurrence of that repeating node to be saved in a separate file then set 'Default value' to the "xpath" of the repeating element without the "<" or ">" characters or any of its attributes, e.g. "Suppliers/Supplier". The value of this parameter is case sensitive.*<br>• *Otherwise set 'Default value' to blank.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

You can have an ‘xml_split_*n*’ parameter for each ‘copy_*n*’ or ‘move_*n*’ that is present and its number must match.

The split files will take their name from the ‘copy_*n*’ or ‘move_*n*’ parameter but will have a sequence number starting from one, e.g. if the value of the ‘copy_*n*’ or ‘move_*n*’ parameter is “export.xml” then the files will be called “export_1.xml”, “export_2.xml”, “export_3.xml” and so on.

# 3.9 Run an Additional Report as a Process Ends (Extra)

## 3.9.1 Introduction

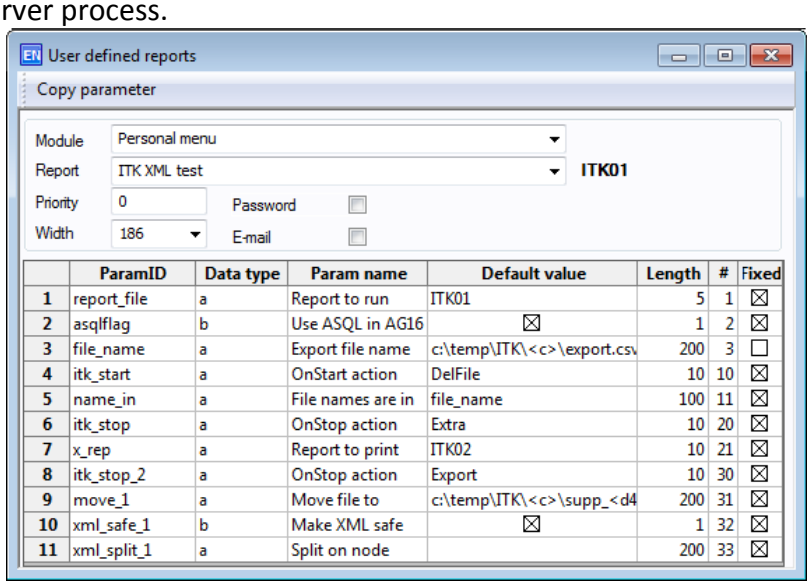
You may wish to add an additional report to a standard server process or user-defined report. This might be simply to produce additional printouts or it may be to add an export file to the process.

The ITKv3 implementation of 'Extra' is different to that in ITKv2:

1. The ITKv2 version of 'Extra' allowed you to run multiple reports from the single call; in ITKv3 you must define a separate 'itk_stop' parameter for each additional report that you want to run (see section 2.1 and below for more details).
2. The ITKv2 version of 'Extra' gave you the ability to copy or move the files out of the 'Report Results' folder as well as to run additional report files; in ITKv3 you use the 'Extra' method to run the reports and the 'Export' method to copy or move the files, again by defining multiple 'itk_stop' parameters.

## 3.9.2 Set Up

In order to use 'Extra' on a process you must create a variant with some new parameters added or add these parameters to your 'User defined report'. The example is for a user defined report that also uses the 'DelFile' and 'Export' methods but the same principle applies to any other server process.




| | ParamID | Data type | Param name | Default value | Length | # | Fixed |
|---|---|---|---|---|---|---|---|
| 1 | report_file | a | Report to run | ITK01 | 5 | 1 | ☒ |
| 2 | asqlflag | b | Use ASQL in AG16 | ☒ | 1 | 2 | ☒ |
| 3 | file_name | a | Export file name | c:\temp\ITK\<c>\export.csv | 200 | 3 | ☐ |
| 4 | itk_start | a | OnStart action | DelFile | 10 | 10 | ☒ |
| 5 | name_in | a | File names are in | file_name | 100 | 11 | ☒ |
| 6 | itk_stop | a | OnStop action | Extra | 10 | 20 | ☒ |
| 7 | x_rep | a | Report to print | ITK02 | 10 | 21 | ☒ |
| 8 | itk_stop_2 | a | OnStop action | Export | 10 | 30 | ☒ |
| 9 | move_1 | a | Move file to | c:\temp\ITK\<c>\supp_<d4 | 200 | 31 | ☒ |
| 10 | xml_safe_1 | b | Make XML safe | ☒ | 1 | 32 | ☒ |
| 11 | xml_split_1 | a | Split on node | | 200 | 33 | ☒ |

### New Parameter 'OnStop action (itk_stop{_*n*})'

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | "Extra" |
| **Length** | "30" |
| **#** | *On a report variant this must be greater than or equal to "100", on a user defined report this can be the next available number.* |

| Column | Value |
|---|---|
| **Fixed** | ☒ |

## New Parameter 'Report to print (x_rep{_*n*})'

This parameter defines the name of the report to be run

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "A" |
| **Default value** | *The name of the report to run, not including its file type suffix (".ARW", ".CRW", ".RPT", etc.)* |
| **Length** | "100" |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Check help-table in (x_test{_*n*})'

Some reports print data from one or more of the help-tables created by the process that they are linked to.

Some processes will store the names of the help tables used in reports to parameters created by the process (these standard parameters usually have names like "hlptab", "helptable", "helptab01", etc.). If the process does not store the names of the help-tables you need in parameter then you can use the 'TabName' method (see section 3.14) to do this for you.

The parameters (standard or otherwise) are used in the body of the additional report; If the process fails to create the help tables then it will not have assigned the table names to the parameters and when your report runs "`SELECT … FROM $hlptab t WHERE …`" (or whatever) it will become "`SELECT … FROM  t WHERE …`" and you will get a SQL error (in this case something along the lines of "table 't' does not exist").

By default 'Extra' will run the additional report regardless of whether or not the process ran without errors, this parameter can be used to modify that behaviour by instructing 'Extra' to only run the additional report if the parameters containing help-table names are not blank and the help-tables exist.

This parameter is optional so if the additional report in the 'x_rep(_*n*)' parameter does not use help-tables or you wish the report to run regardless of whether or not the process ran without errors then omit this parameter. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | *A comma separated list of the process parameters that contain the names of the help-tables used by the report.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

All of the specified parameters must not be blank and the named help-tables must exist in order for the report to be run.

The example screenshot for 'Extra' shows a complex usage and so deserves further explanation:

| ParamID | Default value | Explanation |
|---|---|---|
| itk_start | DelFile | Remove an output file. |
| name_in | file_name | Replace the file tags in the 'file_name' parameter and delete the file if it exists. The process's query will use the modified value of the 'file_name' parameter. |
| itk_stop | Extra | Run an additional report. |
| x_rep | ITK02 | Run the ITK02 report as well as the main ITK01 report. |
| | | The absence of an 'x_test' parameter means that the ITK02 will run unconditionally. |
| itk_stop_2 | Export | Export one or more report files. |
| move_1 | c:\temp\… | Move the output of ITK02 (the second report) to the file path in 'Default value', replacing the file tags. |
| xml_safe_1 | ☒ | Make the content of the exported file XML-safe. |
| xml_split_1 | | Don't split up the XML file. |

### 3.9.3 Summary of the Use of Multiple Parameters

As previously mentioned 'Extra' fully supports multiple 'itk_stop' parameters so that more than one additional report can be run, the parameter names should be defined as follows:

| itk_stop | itk_stop_2 | itk_stop_3 | … | itk_stop_20 |
|---|---|---|---|---|
| x_rep (or 'x_rep_1') | x_rep_2 | x_rep_3 | … | x_rep_20 |
| x_test (or 'x_test_1') | x_test_2 | x_test_3 | … | x_test_20 |

# 3.10 Run a Query as a Process Starts (BiQuery)

## 3.10.1 Introduction

Sometimes when configuring a system you need to be able to run a database update before a standard server process or report is run. Even if that process has a query parameter it may be that this query happens too late for your database update or you may wish to run more than one query.

If your query creates any files then you may wish to include variable elements in the names of those files or manipulate their content in a similar manner to the 'Export' method.

**WARNING**

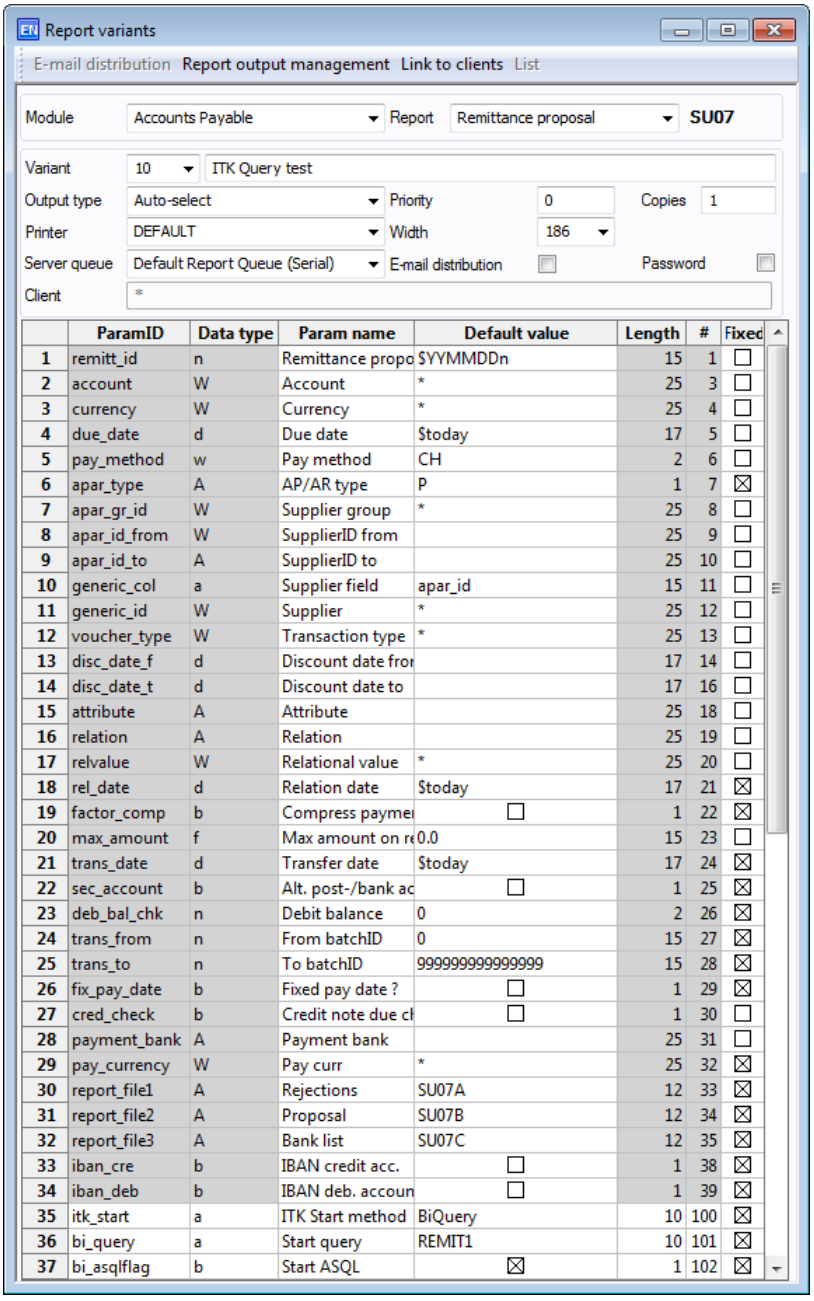
**The 'RunQuery' method that was available in ITKv2 has been removed.**

**WARNING**

You cannot use the 'BiQuery' method in a SaaS environment.

## 3.10.2 Set Up

In order to use 'BiQuery' on a process you must create a variant with some new parameters added or add these parameters to your 'User defined report'. The example is for the Remittance Proposal but the same principle applies to any other server process.




| | ParamID | Data type | Param name | Default value | Length | # | Fixed |
|---|---|---|---|---|---|---|---|
| 1 | remitt_id | n | Remittance propo | $YYMMDDn | 15 | 1 | ☐ |
| 2 | account | W | Account | * | 25 | 3 | ☐ |
| 3 | currency | W | Currency | * | 25 | 4 | ☐ |
| 4 | due_date | d | Due date | $today | 17 | 5 | ☐ |
| 5 | pay_method | w | Pay method | CH | 2 | 6 | ☐ |
| 6 | apar_type | A | AP/AR type | P | 1 | 7 | ☒ |
| 7 | apar_gr_id | W | Supplier group | * | 25 | 8 | ☐ |
| 8 | apar_id_from | W | SupplierID from | | 25 | 9 | ☐ |
| 9 | apar_id_to | A | SupplierID to | | 25 | 10 | ☐ |
| 10 | generic_col | a | Supplier field | apar_id | 15 | 11 | ☐ |
| 11 | generic_id | W | Supplier | * | 25 | 12 | ☐ |
| 12 | voucher_type | W | Transaction type | * | 25 | 13 | ☐ |
| 13 | disc_date_f | d | Discount date fro | | 17 | 14 | ☐ |
| 14 | disc_date_t | d | Discount date to | | 17 | 16 | ☐ |
| 15 | attribute | A | Attribute | | 25 | 18 | ☐ |
| 16 | relation | A | Relation | | 25 | 19 | ☐ |
| 17 | relvalue | W | Relational value | * | 25 | 20 | ☐ |
| 18 | rel_date | d | Relation date | $today | 17 | 21 | ☒ |
| 19 | factor_comp | b | Compress payme | ☐ | 1 | 22 | ☒ |
| 20 | max_amount | f | Max amount on r | 0.0 | 15 | 23 | ☐ |
| 21 | trans_date | d | Transfer date | $today | 17 | 24 | ☒ |
| 22 | sec_account | b | Alt. post-/bank ac | ☐ | 1 | 25 | ☒ |
| 23 | deb_bal_chk | n | Debit balance | 0 | 2 | 26 | ☒ |
| 24 | trans_from | n | From batchID | 0 | 15 | 27 | ☒ |
| 25 | trans_to | n | To batchID | 999999999999999 | 15 | 28 | ☒ |
| 26 | fix_pay_date | b | Fixed pay date ? | ☐ | 1 | 29 | ☒ |
| 27 | cred_check | b | Credit note due cl | ☐ | 1 | 30 | ☐ |
| 28 | payment_bank | A | Payment bank | | 25 | 31 | ☐ |
| 29 | pay_currency | W | Pay curr | * | 25 | 32 | ☒ |
| 30 | report_file1 | A | Rejections | SU07A | 12 | 33 | ☒ |
| 31 | report_file2 | A | Proposal | SU07B | 12 | 34 | ☒ |
| 32 | report_file3 | A | Bank list | SU07C | 12 | 35 | ☒ |
| 33 | iban_cre | b | IBAN credit acc. | ☐ | 1 | 38 | ☒ |
| 34 | iban_deb | b | IBAN deb. accoun | ☐ | 1 | 39 | ☒ |
| 35 | itk_start | a | ITK Start method | BiQuery | 10 | 100 | ☒ |
| 36 | bi_query | a | Start query | REMIT1 | 10 | 101 | ☒ |
| 37 | bi_asqlflag | b | Start ASQL | ☒ | 1 | 102 | ☒ |

## New Parameter 'ITK Start method (itk_start{_*n*})'

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | "BiQuery" |
| **Length** | "30" |
| **#** | *On a report variant this must be greater than or equal to "100", on a user defined report this can be the next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Start query (bi_query{_*n*})'

This parameter defines the name of the AG16 query to run. This query must have been defined in the **Settings** ▶ **System administration** ▶ **Reports** ▶ **SQL Queries** ▶ **Query definition** screen (AG14).

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | *The name of the AG16 query that you want to be run.* |
| **Length** | "30" |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Start ASQL flag (bi_asqlflag or bi_asqlfl_*n*)'

This parameter defines whether or not the AG16 query named in 'bi_query' is written in ASQL or the database's native syntax.

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "b" |
| **Default value** | ☒ *If the AG16 query is written in ASQL.*<br>☐ *If the AG16 query is written in native SQL.* |
| **Length** | "1" |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'File names in (bi_name_in or bi_nam_in_*n*)'

This parameter is used to inform 'BiQuery' that the AG16 query to be run creates one or more files and you wish to:

- Include file name tags in the names of some or all of them.
- Manipulate the content of some or all of them using 'bi_asc_only', 'bi_blnk_lin', 'bi_xml_safe' or 'bi_xml_split'.

This parameter is optional so if you do not wish to modify the names of the files or their content it should be omitted or have its value left blank. . If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | *A comma separated list of the names of the parameters containing the file names to manipulate. Spaces on either side of each comma are ignored.*<br>*The default value of each of the referenced parameters can contain file name tags, see section 4.3.*<br>*A file name with no path is assumed to be in the folder defined by the "AGRESSO_IMPORT" environment variable.* |

| Column | Value |
|---|---|
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *Next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Remove non-ASCII chars (bi_asc_only or bi_asc_on_*n*)'

This parameter is used to control whether any non-ASCII characters in a file should be retained or not.

This parameter is optional so if all of the files whose names are in the 'bi_name_in' parameter can only contain characters in the ASCII character set (see section 5.2) or there could be non-ASCII characters and you do not want them removed then this parameter can either be omitted or have its value set to "N". If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "A" |
| **Default value** | • *If all the files in 'bi_name_in' could contain non-ASCII characters and you want them removed then set 'Default value' to "*Y*".*<br>• *If all of the files in 'bi_name_in' can only contain ASCII characters or they could contain non-ASCII characters but you don't want them removed then set 'Default value' to "*N*".*<br>• *Otherwise set 'Default value' to a comma separated list of "*Y*"s and "*N*"s, each value corresponds to the file in the same position in 'bi_name_in' and there must be the same number of values as there are in 'bi_name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Allow blank lines (bi_blnk_lin or bi_bnk_ln_*n*)'

This parameter is used to control whether any blank lines in the output file should be retained in the moved and/or renamed file or not.

This parameter is optional so if all of the files whose names are in the 'bi_name_in' parameter cannot contain blank lines or can contain them but you wish to retain them then this parameter can either be omitted or have its value set to "Y". If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "A" |

| Column | Value |
|---|---|
| **Default value** | • *If all the files in 'bi_name_in' cannot contain blank lines or they could contain blank lines but you don't want them removed then set 'Default value' to "*Y*".*<br>• *If all the files in 'bi_name_in' can contain blank lines and you want them removed then set 'Default value' to "*N*".*<br>• *Otherwise set 'Default value' to a comma separated list of "*Y*"s and "*N*"s, each value corresponds to the file in the same position in 'bi_name_in' and there must be the same number of values as there are in 'bi_name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Make XML safe (bi_xml_safe or bi_xml_sa_*n*)'

Business World reports (particularly ARW reports) are sometimes used to create XML files. Some characters are not allowed in XML files because they would cause programs that read the XML to fail. These characters are known as "unsafe" characters and each one must be replaced by a string of characters called an "entity" to avoid this problem. The characters and their corresponding entities are as follows:

| Unsafe Character | Entity |
|---|---|
| & | &amp; |
| < | &lt; |
| > | &gt; |
| ' | &apos; |
| " | &quot; |

This parameter will instruct 'BiQuery' to scan a file and replace any unsafe characters with their entity.

This parameter is optional so if none of the files whose names are in the 'bi_name_in' parameter are XML files or the ones that are cannot contain any unsafe characters then this parameter can either be omitted or set to "N". If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "A" |

| Column | Value |
|---|---|
| **Default value** | • *If all the files in 'bi_name_in' are XML and may contain "unsafe" characters then set 'Default value' to "*Y*".*<br>• *If none of the files in 'bi_name_in' are XML or they are but cannot contain "unsafe" characters then set 'Default value' to "*N*".*<br>• *Otherwise set 'Default value' to a comma separated list of "*Y*"s and "*N*"s, each value corresponds to the file in the same position in 'bi_name_in' and there must be the same number of values as there are in 'bi_name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Split XML files (bi_xml_split or bi_xml_sp_*n*)'

Business World reports (particularly ARW reports) are sometimes used to create XML files. If an XML file contains list of things (e.g. a list of suppliers or GL postings) then these will be represented by a "repeating node" (one node per thing) in the file.

By default 'BiQuery' will keep all of these nodes in a single file just like the report that produced the original output file. It is possible that the system that reads the XML file may not be able to handle files containing "repeating nodes" and so this parameter allows you to override the default behaviour and put each occurrence of the "repeating node" into a separate file.

This parameter is optional so if none of the files whose names are in the 'bi_name_in' parameter are XML files or the ones that are don't need to be split then this parameter can either be omitted or it can be included and its value set to blank. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |

| Column | Value |
|---|---|
| **Default value** | • *If all the files in 'bi_name_in' are XML files and contain the same "repeating node" and you wish each occurrence of that repeating node to be saved in a separate file then set 'Default value' to the "xpath" of the repeating element without the "<" or ">" characters or any of its attributes, e.g. "*Suppliers/Supplier*". The value of this parameter is case sensitive.*<br>• *If none of the files in 'bi_name_in' contain XML or they do but you don't want to split the repeating elements into separate files then set 'Default value' to blank.*<br>• *Otherwise set 'Default value' to a comma separated list of element names, each value corresponds to the file in the same position in 'bi_name_in' and there must be the same number of values as there are in 'bi_name_in'. A blank element in the list should be represented by two consecutive apostrophes thus "'' ".* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

The split files will take their name from the parameter pointed to by the 'bi_name_in' parameter but will have a sequence number starting from one.

**Example**
If the parameter pointed to by 'bi_name_in' contains "C:\temp\BwOut\st_<c><o>.xml" and it is run in client "EN" and its order number is 85 and three files are created then their names will be "st_EN85_1.xml", "st_EN85_2.xml" and "st_EN85_3.xml".

## Additional Parameters

These additional parameters can have any 'ParamID' that you wish that does not clash with any other parameters on the server process they are used to transfer values into the AG16 query that you do not want to hardcode or you wish the user to enter. These parameters can be referenced in the query in the usual way.

### 3.10.3 Summary of the Use of Multiple Parameters

'BiQuery' fully supports multiple 'itk_start' parameters so that more than one AG16 query can be run at the start of the process, but because of technical restrictions when there is an explicit sequence number some of the parameter names are different as follows:

| itk_start | itk_start_2 | itk_start_3 | … | itk_start_20 |
|---|---|---|---|---|
| bi_query | bi_query_2 | bi_query_3 | … | bi_query_20 |
| bi_asqlflag | bi_asqlfl_2 | bi_asqlfl_3 | … | bi_asqlfl_20 |
| bi_name_in | bi_nam_in_2 | bi_nam_in_3 | … | bi_nam_in_20 |

| itk_start | itk_start_2 | itk_start_3 | … | itk_start_20 |
|---|---|---|---|---|
| bi_asc_only | bi_asc_on_2 | bi_asc_on_3 | … | bi_asc_on_20 |
| bi_blnk_lin | bi_bnk_ln_2 | bi_bnk_ln_3 | … | bi_bnk_ln_20 |
| bi_xml_safe | bi_xml_sa_2 | bi_xml_sa_3 | … | bi_xml_sa_20 |
| bi_xml_split | bi_xml_sp_2 | bi_xml_sp_3 | … | bi_xml_sp_20 |

# 3.11 Run a Query as a Process Ends (BeQuery)

## 3.11.1 Introduction

Sometimes when configuring a system you need to be able to run a database update at the end of a standard server process or report. Even if that process has a query parameter it may be that this query happens too early for your database update or you may wish to run more than one query.

If your query creates any files then you may wish to include variable elements in the names of those files or manipulate their content in a similar manner to the 'Export' method.

**WARNING**

**The 'RunQuery' method that was available in ITKv2 has been removed.**

**WARNING**

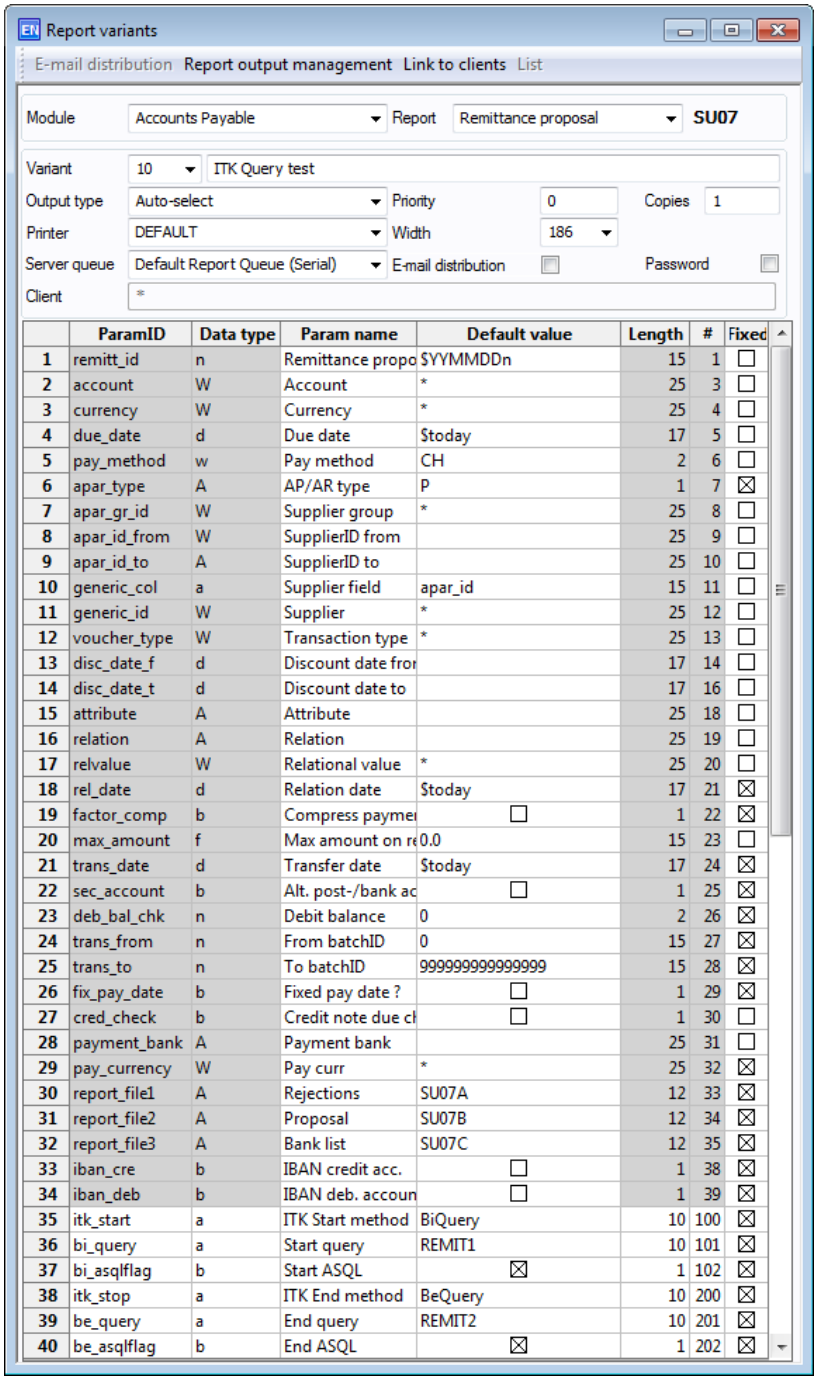
You cannot use the 'BeQuery' method in a SaaS environment.

**WARNING**

**The 'be_xparam_*n*' parameters that were available in the ITKv2 'BeQuery' function have been removed because they are no longer necessary: the ITKv3 method automatically makes ALL of the process's parameters available to your query.**

## 3.11.2 Set Up

In order to use 'BeQuery' on a process you must create a variant with some new parameters added or add these parameters to your 'User defined report'. The example is for a Remittance Proposal variant that is also using the 'BiQuery' method but the same principle applies to any other server process.




| | ParamID | Data type | Param name | Default value | Length | # | Fixed |
|---|---|---|---|---|---|---|---|
| 1 | remitt_id | n | Remittance propo | $YYMMDDn | 15 | 1 | ☐ |
| 2 | account | W | Account | * | 25 | 3 | ☐ |
| 3 | currency | W | Currency | * | 25 | 4 | ☐ |
| 4 | due_date | d | Due date | $today | 17 | 5 | ☐ |
| 5 | pay_method | w | Pay method | CH | 2 | 6 | ☐ |
| 6 | apar_type | A | AP/AR type | P | 1 | 7 | ☒ |
| 7 | apar_gr_id | W | Supplier group | * | 25 | 8 | ☐ |
| 8 | apar_id_from | W | SupplierID from | | 25 | 9 | ☐ |
| 9 | apar_id_to | A | SupplierID to | | 25 | 10 | ☐ |
| 10 | generic_col | a | Supplier field | apar_id | 15 | 11 | ☐ |
| 11 | generic_id | W | Supplier | * | 25 | 12 | ☐ |
| 12 | voucher_type | W | Transaction type | * | 25 | 13 | ☐ |
| 13 | disc_date_f | d | Discount date fro | | 17 | 14 | ☐ |
| 14 | disc_date_t | d | Discount date to | | 17 | 16 | ☐ |
| 15 | attribute | A | Attribute | | 25 | 18 | ☐ |
| 16 | relation | A | Relation | | 25 | 19 | ☐ |
| 17 | relvalue | W | Relational value | * | 25 | 20 | ☐ |
| 18 | rel_date | d | Relation date | $today | 17 | 21 | ☒ |
| 19 | factor_comp | b | Compress payme | ☐ | 1 | 22 | ☒ |
| 20 | max_amount | f | Max amount on r | 0.0 | 15 | 23 | ☐ |
| 21 | trans_date | d | Transfer date | $today | 17 | 24 | ☒ |
| 22 | sec_account | b | Alt. post-/bank ac | ☐ | 1 | 25 | ☒ |
| 23 | deb_bal_chk | n | Debit balance | 0 | 2 | 26 | ☒ |
| 24 | trans_from | n | From batchID | 0 | 15 | 27 | ☒ |
| 25 | trans_to | n | To batchID | 999999999999999 | 15 | 28 | ☒ |
| 26 | fix_pay_date | b | Fixed pay date ? | ☐ | 1 | 29 | ☒ |
| 27 | cred_check | b | Credit note due cl | ☐ | 1 | 30 | ☐ |
| 28 | payment_bank | A | Payment bank | | 25 | 31 | ☐ |
| 29 | pay_currency | W | Pay curr | * | 25 | 32 | ☒ |
| 30 | report_file1 | A | Rejections | SU07A | 12 | 33 | ☒ |
| 31 | report_file2 | A | Proposal | SU07B | 12 | 34 | ☒ |
| 32 | report_file3 | A | Bank list | SU07C | 12 | 35 | ☒ |
| 33 | iban_cre | b | IBAN credit acc. | ☐ | 1 | 38 | ☒ |
| 34 | iban_deb | b | IBAN deb. accoun | ☐ | 1 | 39 | ☒ |
| 35 | itk_start | a | ITK Start method | BiQuery | 10 | 100 | ☒ |
| 36 | bi_query | a | Start query | REMIT1 | 10 | 101 | ☒ |
| 37 | bi_asqlflag | b | Start ASQL | ☒ | 1 | 102 | ☒ |
| 38 | itk_stop | a | ITK End method | BeQuery | 10 | 200 | ☒ |
| 39 | be_query | a | End query | REMIT2 | 10 | 201 | ☒ |
| 40 | be_asqlflag | b | End ASQL | ☒ | 1 | 202 | ☒ |

## New Parameter 'ITK End method (itk_stop{_*n*})'

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | "BeQuery" |
| **Length** | "30" |
| **#** | *On a report variant this must be greater than or equal to "100", on a user defined report this can be the next available number.* |
| **Fixed** | ☒ |

## New Parameter 'End query (be_query{_*n*})'

This parameter defines the name of the AG16 query to run. This query must have been defined in the **Settings** ▶ **System administration** ▶ **Reports** ▶ **SQL Queries** ▶ **Query definition** screen (AG14).

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | *The name of the AG16 query that you want to be run.* |
| **Length** | "30" |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'End ASQL flag (be_asqlflag or be_asqlfl_*n*)'

This parameter defines whether or not the AG16 query named in 'be_query' is written in ASQL or the database's native syntax.

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "b" |
| **Default value** | ☒ *If the AG16 query is written in ASQL*<br>☐ *If the AG16 query is written in native SQL.* |
| **Length** | "1" |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

Due to technical restrictions if one of the parameters is the name of a database table that you wish to access in your AG16 query then the value of your "be_asqlflag" parameter must not be one

## New Parameter 'File names in (be_name_in or be_nam_in_*n*)'

This parameter is used to inform 'BeQuery' that the AG16 query to be run creates one or more files and you wish to:

- Include file name tags in the names of some or all of them.
- Manipulate the content of some or all of them using 'be_asc_only', 'be_blnk_lin', 'be_xml_safe' or 'be_xml_split'.

This parameter is optional so if you do not wish to modify the names of the files or their content it should be omitted or have its value left blank. . If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |

| Column | Value |
|---|---|
| **Default value** | *A comma separated list of the names of the parameters containing the file names to manipulate. Spaces on either side of each comma are ignored.*<br><br>*The default value of each of the referenced parameters can contain file name tags, see section 4.3.*<br><br>*A file name with no path is assumed to be in the folder defined by the "AGRESSO_EXPORT" environment variable.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Remove non-ASCII chars (be_asc_only or be_asc_on_*n*)'

This parameter is used to control whether any non-ASCII characters in a file should be retained or not.

This parameter is optional so if all of the files whose names are in the 'be_name_in' parameter can only contain characters in the ASCII character set (see section 5.2) or there could be non-ASCII characters and you do not want them removed then this parameter can either be omitted or have its value set to "N". If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "A" |
| **Default value** | • *If all the files in 'be_name_in' could contain non-ASCII characters and you want them removed then set 'Default value' to "*Y*".*<br>• *If all of the files in 'be_name_in' can only contain ASCII characters or they could contain non-ASCII characters but you don't want them removed then set 'Default value' to "*N*".*<br>• *Otherwise set 'Default value' to a comma separated list of "*Y*"s and "*N*"s, each value corresponds to the file in the same position in 'be_name_in' and there must be the same number of values as there are in 'be_name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Allow blank lines (be_blnk_lin or be_bnk_ln_*n*)'

This parameter is used to control whether any blank lines in the output file should be retained in the moved and/or renamed file or not.

This parameter is optional so if all of the files whose names are in the 'be_name_in' parameter cannot contain blank lines or can contain them but you wish to retain them then this parameter can either be omitted or have its value set to "Y". If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "A" |
| **Default value** | • *If all the files in 'be_name_in' cannot contain blank lines or they could contain blank lines but you don't want them removed then set 'Default value' to "Y".*<br>• *If all the files in 'be_name_in' can contain blank lines and you want them removed then set 'Default value' to "N".*<br>• *Otherwise set 'Default value' to a comma separated list of "Y"s and "N"s, each value corresponds to the file in the same position in 'be_name_in' and there must be the same number of values as there are in 'be_name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Make XML safe (be_xml_safe or be_xml_sa_*n*)'

Business World reports (particularly ARW reports) are sometimes used to create XML files. Some characters are not allowed in XML files because they would cause programs that read the XML to fail. These characters are known as "unsafe" characters and each one must be replaced by a string of characters called an "entity" to avoid this problem. The characters and their corresponding entities are as follows:

| Unsafe Character | Entity |
|---|---|
| & | &amp; |
| < | &lt; |
| > | &gt; |
| ' | &apos; |
| " | &quot; |

This parameter will instruct 'BeQuery' to scan a file and replace any unsafe characters with their entity.

This parameter is optional so if none of the files whose names are in the 'be_name_in' parameter are XML files or the ones that are cannot contain any unsafe characters then this parameter can either be omitted or set to "N". If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "A" |

| Column | Value |
|---|---|
| **Default value** | • *If all the files in 'be_name_in' are XML and may contain "unsafe" characters then set 'Default value' to "*Y*".*<br>• *If none of the files in 'be_name_in' are XML or they are but cannot contain "unsafe" characters then set 'Default value' to "*N*".*<br>• *Otherwise set 'Default value' to a comma separated list of "*Y*"s and "*N*"s, each value corresponds to the file in the same position in 'be_name_in' and there must be the same number of values as there are in 'be_name_in'.* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## New Parameter 'Split XML files (be_xml_split or be_xml_sp_*n*)'

Business World reports (particularly ARW reports) are sometimes used to create XML files. If an XML file contains list of things (e.g. a list of suppliers or GL postings) then these will be represented by a "repeating node" (one node per thing) in the file.

By default 'BeQuery' will keep all of these nodes in a single file just like the report that produced the original output file. It is possible that the system that reads the XML file may not be able to handle files containing "repeating nodes" and so this parameter allows you to override the default behaviour and put each occurrence of the "repeating node" into a separate file.

This parameter is optional so if none of the files whose names are in the 'be_name_in' parameter are XML files or the ones that are don't need to be split then this parameter can either be omitted or it can be included and its value set to blank. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |

| Column | Value |
|---|---|
| **Default value** | • *If all the files in 'be_name_in' are XML files and contain the same "repeating node" and you wish each occurrence of that repeating node to be saved in a separate file then set 'Default value' to the "xpath" of the repeating element without the "<" or ">" characters or any of its attributes, e.g. "*Suppliers/Supplier*". The value of this parameter is case sensitive.*<br>• *If none of the files in 'be_name_in' contain XML or they do but you don't want to split the repeating elements into separate files then set 'Default value' to blank.*<br>• *Otherwise set 'Default value' to a comma separated list of element names, each value corresponds to the file in the same position in 'be_name_in' and there must be the same number of values as there are in 'be_name_in'. A blank element in the list should be represented by two consecutive apostrophes thus "'' ".* |
| **Length** | *Greater than or equal to the length of the 'Default value' up to a maximum of 255.* |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

The split files will take their name from the parameter pointed to by the 'be_name_in' parameter but will have a sequence number starting from one.

**Example**
If the parameter pointed to by 'be_name_in' contains "C:\temp\BwOut\st_<c><o>.xml" and it is run in client "EN" and its order number is 85 and three files are created then their names will be "st_EN85_1.xml", "st_EN85_2.xml" and "st_EN85_3.xml".

## Additional Parameters

These additional parameters can have any 'ParamID' that you wish that does not clash with any other parameters on the server process they are used to transfer values into the AG16 query that you do not want to hardcode or you wish the user to enter. These parameters can be referenced in the query in the usual way.

### 3.11.3 Summary of the Use of Multiple Parameters

'BeQuery' fully supports multiple 'itk_stop' parameters so that more than one AG16 query can be run at the end of the process, but because of technical restrictions when there is an explicit sequence number some of the parameter names are different as follows:

| itk_stop | itk_stop_2 | itk_stop_3 | … | itk_stop_20 |
|---|---|---|---|---|
| be_query | be_query_2 | be_query_3 | … | be_query_20 |
| be_asqlflag | be_asqlfl_2 | be_asqlfl_3 | … | be_asqlfl_20 |
| be_name_in | be_nam_in_2 | be_nam_in_3 | … | be_nam_in_20 |

| itk_stop | itk_stop_2 | itk_stop_3 | … | itk_stop_20 |
|---|---|---|---|---|
| be_asc_only | be_asc_on_2 | be_asc_on_3 | … | be_asc_on_20 |
| be_blnk_lin | be_bnk_ln_2 | be_bnk_ln_3 | … | be_bnk_ln_20 |
| be_xml_safe | be_xml_sa_2 | be_xml_sa_3 | … | be_xml_sa_20 |
| be_xml_split | be_xml_sp_2 | be_xml_sp_3 | … | be_xml_sp_20 |

## 3.12 Run a Command-Line Tool as a Process Starts (BiRun)

**WARNING**

**Method 'BiRun' is not implemented in the current version of ITKv3.**

**WARNING**

When it is implemented you will not be able to use the 'BiRun' method in a SaaS environment.

## 3.13 Run a Command-Line Tool as a Process Ends (BeRun)

**WARNING**

**Method 'BeRun' is not implemented in the current version of ITKv3.**

**WARNING**

When it is implemented you will not be able to use the 'BeRun' method in a SaaS environment.

# 3.14 Save Help-Table Name (TabName)

## 3.14.1 Introduction

Business World server processes often create temporary tables (called “help-tables”) in which they store “working information” that assists them in carrying out of their task or sometimes to improve the performance of that task.

Occasionally it would be useful to access one or more of these help-tables in a report or AG16 query that runs at the end of the job (using the ‘Extra’ or ‘BeQuery’ method), but it is not usually possible to reliably determine the names of these help-tables because:

- Help-tables are given a temporary name:
  - On a SQL Server system a help-table’s name has the following form:
    **##H***dddddppppssssooooo*
    Where *ddddd* is the name of the Business World data source (e.g. “agrdemo571”)
    *pppp* is the name of the process (e.g. “GL07”)
    *ssss* is a number that indicates the sequence in which the help-tables were created, padded with leading zeros to be four digits long
    *ooooo* is the order number of the process, padded with leading zeros to be five digits long.
  - On an Oracle system it is somewhat different:
    **H***uuuuupppssssooooo*
    Where *uuuuu* is the **Oracle** user name not Business World user name (e.g. “agresso”)
    *pppp* is the name of the process (e.g. “GL07”)
    *ssss* is a number that indicates the sequence in which the help-tables were created, padded with leading zeros to be four digits long
    *ooooo* is the order number of the process, padded with leading zeros to be five digits long.
- If you enable the ‘Tmp table parameter’ when you order the process then on SQL Server the “**##**” prefix is omitted.
- In some processes the number of help-tables created and therefore the ‘*ssss*’ segment of their names can vary depending on what values you gave to the process parameters.

The ‘TabName’ method gives you the capability of extracting the name of a help-table as it is created and saving it in a process parameter that you can then use in your report or AG16.

## 3.14.2 Set Up

In order to use ‘TabName’ on a process you must first determine:

1. Which of the process’s help-tables you need to capture the names of for your report or AG16 query.
2. The “step ID” of each of the SQL steps that creates one of those help-tables.

### 3.14.2.1 Identifying the Help-table

The first of these activities is basically “detective work” and is likely to involve running the process with both the ‘Tmp table parameter’ enabled and the ‘Log level parameter’ set to higher than “Standard logging” and then inspecting both the more detailed log file and the contents of the help-tables.

Once you have identified the help-table you need it is straightforward to obtain the “step ID” of the table’s creation. Here is an extract from a CS15 log file:

```
…
1:04:28 PM Table '##Hagrdemo571CS15001000058' is referenced as cUpdateRelValueTable
1:04:28 PM [    0] Creating the table for rel-values to upsert (CS15_RELATIONVALUE0130)
1:04:28 PM ACT: OnHelpTableCreated is being handled by ACT extension(s).
1:04:28 PM ACT: OnHelpTableCreated has been handled by ACT extension(s).
1:04:28 PM [    0] Create a table with all the relations to insert. (CS15_RELATIONVALUE0140)
1:04:28 PM Table '##Hagrdemo571CS15001100058' is referenced as cInsertRelValueTable
1:04:28 PM [    0] Creating the table for rel-values to create (CS15_RELATIONVALUE0150)
1:04:28 PM ACT: OnHelpTableCreated is being handled by ACT extension(s).
1:04:28 PM ACT: OnHelpTableCreated has been handled by ACT extension(s).
1:04:28 PM [    0] Create a table with all the relations to insert. (CS15_RELATIONVALUE0160)
1:04:28 PM [    0] Creating cReportTable from cInsertTable. (CS150020)
1:04:28 PM ACT: OnHelpTableCreated is being handled by ACT extension(s).
…
```

Most SQL steps in the server process are labelled with an ID which is shown in brackets at the end of the log file line; so for example if the table whose name you need to store is the one whose logical name is “cReportTable” you can see that the step where it is created has an ID of “CS150020”.

**WARNING**

Some standard Business World server processes create a small number of their help-tables in such a way that ACT cannot detect the creation of that help-table. Once you have installed ITK on your Business World system you can identify these tables in a process’s log file by the fact the message:

```
HH:MM:SS [    n] Creating the table for …
```

(or similar)

is not followed by:

```
HH:MM:SS ACT: OnHelpTableCreated is being handled by ACT extension(s).
HH:MM:SS ACT: OnHelpTableCreated has been handled by ACT extension(s).
```

It is not possible for ‘TabName’ to store the name of such help-tables.

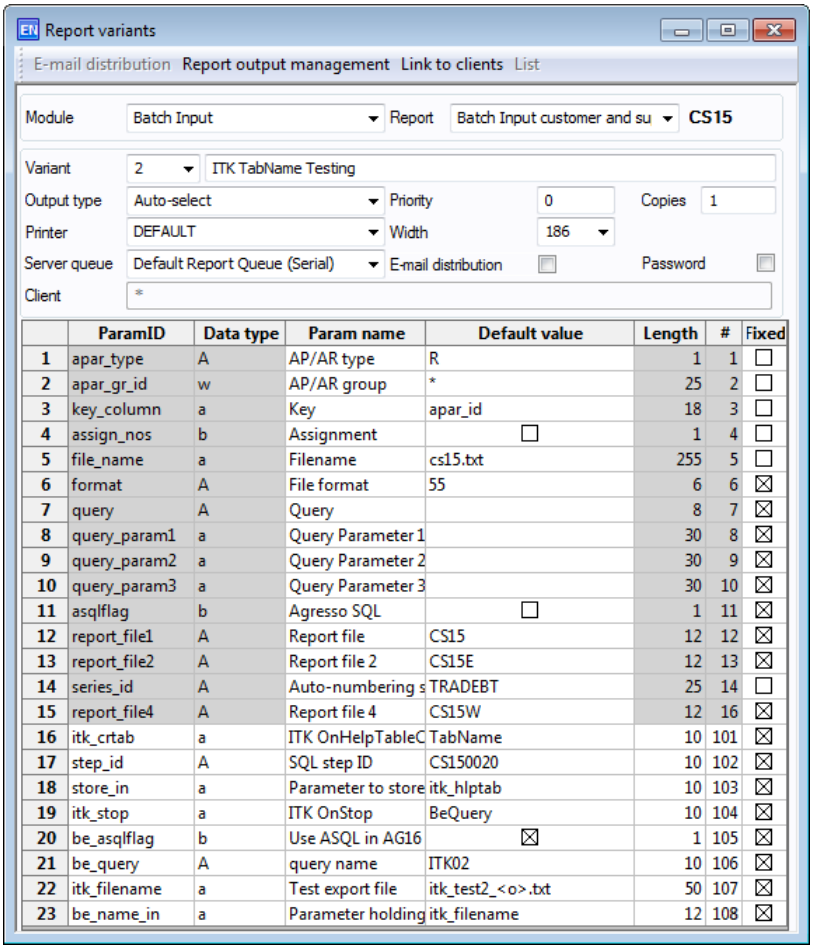
Examples of such help-tables can be seen in steps “CS15_RELATIONVALUE0140” and “CS15_RELATIONVALUE0160” in the above log file extract.

**WARNING**

Some standard Business World server processes will label more than one SQL step with the same ID, if you wish to access one of these SQL steps then ‘TabName’ has an ‘occur_no’ parameter (see below) that allows you to specify which encounter of the ID you wish to utilise.

### 3.14.2.2 The Process Parameters

Once you have identified the value of the ID of the SQL step where the table is created you must create a variant with some new parameters added or add these parameters to your ‘User defined report’. The example is for a CS15 variant that is also using the ‘BeQuery’ method to run an AG16 query but the same principle applies to any other server process and also to running a report using the ‘Extra’ method.




| | ParamID | Data type | Param name | Default value | Length | # | Fixed |
|---|---|---|---|---|---|---|---|
| 1 | apar_type | A | AP/AR type | R | 1 | 1 | ☐ |
| 2 | apar_gr_id | w | AP/AR group | * | 25 | 2 | ☐ |
| 3 | key_column | a | Key | apar_id | 18 | 3 | ☐ |
| 4 | assign_nos | b | Assignment | ☐ | 1 | 4 | ☐ |
| 5 | file_name | a | Filename | cs15.txt | 255 | 5 | ☐ |
| 6 | format | A | File format | 55 | 6 | 6 | ☒ |
| 7 | query | A | Query | | 8 | 7 | ☒ |
| 8 | query_param1 | a | Query Parameter 1 | | 30 | 8 | ☒ |
| 9 | query_param2 | a | Query Parameter 2 | | 30 | 9 | ☒ |
| 10 | query_param3 | a | Query Parameter 3 | | 30 | 10 | ☒ |
| 11 | asqlflag | b | Agresso SQL | ☐ | 1 | 11 | ☒ |
| 12 | report_file1 | A | Report file | CS15 | 12 | 12 | ☒ |
| 13 | report_file2 | A | Report file 2 | CS15E | 12 | 13 | ☒ |
| 14 | series_id | A | Auto-numbering s | TRADEBT | 25 | 14 | ☐ |
| 15 | report_file4 | A | Report file 4 | CS15W | 12 | 16 | ☒ |
| 16 | itk_crtab | a | ITK OnHelpTableC | TabName | 10 | 101 | ☒ |
| 17 | step_id | A | SQL step ID | CS150020 | 10 | 102 | ☒ |
| 18 | store_in | a | Parameter to store | itk_hlptab | 10 | 103 | ☒ |
| 19 | itk_stop | a | ITK OnStop | BeQuery | 10 | 104 | ☒ |
| 20 | be_asqlflag | b | Use ASQL in AG16 | ☒ | 1 | 105 | ☒ |
| 21 | be_query | A | query name | ITK02 | 10 | 106 | ☒ |
| 22 | itk_filename | a | Test export file | itk_test2_<o>.txt | 50 | 107 | ☒ |
| 23 | be_name_in | a | Parameter holding | itk_filename | 12 | 108 | ☒ |

The AG16 query referred to in the screenshot uses ASQL's "COPY TO" command to copy the contents of the 'cReportTable' help-table to an export file whose name is defined by the 'itk_filename' parameter, which will then be the parameter that you use in your report or query to access the table.

## New Parameter 'ITK OnHelpTableCreated (itk_crtab{_*n*})'

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "a" |
| **Default value** | "TabName" |
| **Length** | "30" |
| **#** | *On a report variant this must be greater than or equal to "100", on a user defined report this can be the next available number.* |
| **Fixed** | ☒ |

## New Parameter 'SQL step ID (step_id{_*n*})'

This parameter defines the value of the ID that the SQL step is labelled with.

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "A" |

| Column | Value |
|---|---|
| **Default value** | *The ID with which the SQL step whose table name you wish to store is labelled.* |
| **Length** | "30" |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

### New Parameter 'Parameter to store name (store_in{_*n*})'

This parameter defines the name of the new process parameter that will be created to store the help-table's name in.

This parameter is mandatory and must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "A" |
| **Default value** | *The name of the new process parameter that you want to store the table name in.* |
| **Length** | "30" |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

### New Parameter 'Encounter number (occur_no{_*n*})'

This parameter allows you to make use of SQL Step IDs that occur more than once in the process.

This parameter is optional so if there will only be one occurrence of the ID in a process or there are several but you want to use the first one then this parameter can either be omitted or it can be included and its value set to 1. If the parameter is present it must be defined as follows:

| Column | Value |
|---|---|
| **Data type** | "n" |
| **Default value** | *The number of the encounter of this SQL step Id whose table name you want to store.* |
| **Length** | "4" |
| **#** | *The next available number.* |
| **Fixed** | ☒ |

## 3.14.3 Summary of the Use of Multiple Parameters

'TabName' fully supports multiple 'itk_crtab' parameters so that you can store the name of more than one of the process's help-tables, the parameter names should be defined as follows:

| itk_crtab | itk_crtab_2 | itk_crtab_3 | … | itk_crtab_20 |
|---|---|---|---|---|
| step_id | step_id_2 | step_id_3 | … | step_id_20 |
| store_in | store_in_2 | store_in_3 | … | store_in_20 |
| occur_no | occur_no_2 | occur_no_3 | … | occur_no_20 |

# 4 Additional Information

## 4.1 Converting a Report into a Server Process

Due to a technical restriction ITKv3 methods can only be used on server processes, therefore if you wish to use a method on a simple report that report must be converted into a server process. This can be done by turning it into an "AG16" type process with a query that doesn't do anything. You do this by making the following changes:

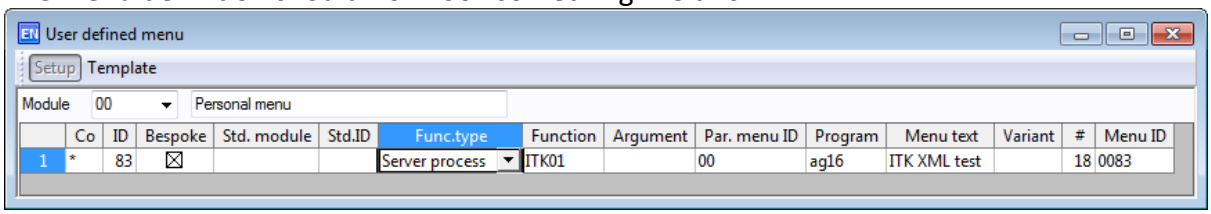
- Modify your report's menu entry,
- Create a new 'SQL query',
- Add additional parameters to your report's 'User defined report' definition.

### 4.1.1 The Menu Entry

Use the **Settings ► System administration ► Menus ► User defined menu** screen (AG27) to change the following columns:

- 'Func type' from "Report" to "Server process",
- 'Program' from blank to "ag16".

The menu definition should now look something like this:

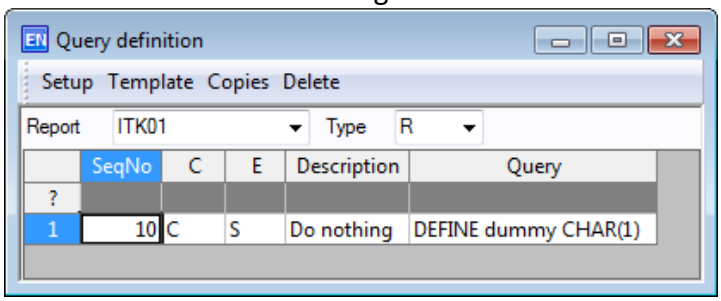



### 4.1.2 The SQL Query

Use the **Settings ► System administration ► Reports ► SQL queries ► Query definition** screen (AG14) to create a new SQL query with the same name as your report:

- Its 'Type' must be "R" (for "Report"),
- 'SeqNo' should be "10",
- 'C' should be "C",
- 'E' should be "S",
- 'Description' should be "Do nothing",
- 'Query' should be "DEFINE dummy CHAR(1)".

The query definition should now look something like this:




### 4.1.3 The Additional Parameters

Use the **Settings ► System administration ► Reports ► User defined report** screen (AG35) to add two new parameters to the report:

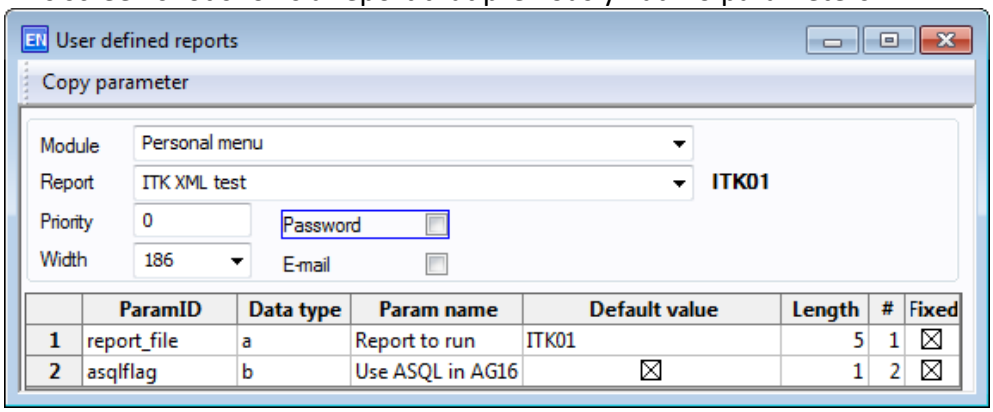
- Parameter one:

| Column Name | Value |
|---|---|
| **ParamID** | "report_file" |
| **Data type** | "a" |
| **Param name** | "Report to run" *(or similar)* |
| **Default value** | *The name of your report* |
| **Length** | "30" |
| **#** | *The next available number* |
| **Fixed** | ☒ |

- Parameter two

| Column Name | Value |
|---|---|
| **ParamID** | "asqflag" |
| **Data type** | "b" |
| **Param name** | "Use ASQL in AG16" *(or similar)* |
| **Default value** | ☒ |
| **Length** | "1" |
| **#** | *The next available number* |
| **Fixed** | ☒ |

This screen shot shows a report that previously had no parameters:




Now check that your report still works and if it does you can now add the ITK method(s) that you want to use to your user defined report definition.

## 4.2 The Query for the Import File Conversions

This query is used in the examples used in:

- 'CSV Import File Conversion (ConvCsv)',
- 'Excel Workbook Conversion (ConvXls)',
- 'CFR Import File Conversion (ConvCfr)' and
- 'FSR Import File Conversion (ConvFsr)'

to turn each of the sample files into a GL07 batch.

| SeqNo | C | E | Description |
|---|---|---|---|
| 10 | C | S | Header date |
| DEFINE file_date CHAR(8) | | | |
| 20 | C | S | Footer total |
| DEFINE file_dr FLOAT | | | |
| 30 | C | S | Balance amount |
| DEFINE calc_dr FLOAT | | | |
| 40 | C | S | Error flag |
| DEFINE error_test IDENT(6) | | | |
| 50 | C | S | Create help table |
| CREATE TABLE $*tab1 AS SELECT SPACE(25) AS account, description, dim_1, dim_2, dim_3, dim_4, dim_5, dim_6, dim_7, SPACE(15) AS cur_amount, SPACE(5) AS rec_type, sequence_no FROM acrbatchinput WHERE 1 = 2 | | | |
| 60 | N | S | Get data |
| COPY IN IMPORT FILE = '$query_param1', COLSEP = T, TABLE=$*tab1, rec_type, account, dim_1, dim_2, dim_3, dim_4, dim_5, dim_6, dim_7, cur_amount, description | | | |
| 70 | N | S | Remove existing batch data |
| DELETE FROM acrbatchinput WHERE batch_id = '$batch_id' AND client = '$client' AND interface = '$interface' | | | |
| 80 | N | S | Get date |
| SELECT CONCAT(SUBSTR(account, 7, 4), CONCAT(SUBSTR(account, 4, 2), SUBSTR(account, 1, 2))) INTO :file_date FROM $*tab1 WHERE rec_type = 'H1' | | | |
| 90 | N | S | Get total dr |
| SELECT TO_MONEY(account) INTO :file_dr FROM $*tab1 WHERE rec_type = 'F1' | | | |
| 100 | N | S | Remove headers and footers |
| DELETE FROM $*tab1 WHERE LEFT(rec_type, 1) IN ('H', 'F') | | | |
| 110 | N | S | Get total of debits |
| SELECT SUM(TO_MONEY(cur_amount)) INTO :calc_dr FROM $*tab1 WHERE TO_MONEY(cur_amount) > 0 | | | |
| 120 | N | S | Set column name if footer + balance = zero |
| SELECT 'col_0' INTO :error_test FROM asysdummy WHERE col_0 = '0' AND :file_dr = :calc_dr | | | |
| 130 | N | S | Generate a syntax error if out of balance |
| SELECT :error_test INTO :error_test FROM asysdummy WHERE col_0 = '0' | | | |
| 140 | N | S | Sequence no |
| SEQUENCE ON $*tab1 (sequence_no, 0) | | | |

| SeqNo | C | E | Description |
|---|---|---|---|
| 150 | N | S | Create batch |
| INSERT INTO acrbatchinput (batch_id, interface, voucher_type, trans_type, client, account, dim_1, dim_2, dim_3, dim_4, dim_5, dim_6, dim_7, tax_code, tax_system, currency, dc_flag, cur_amount, amount, number_1, value_1, value_2, value_3, description, trans_date, sequence_no, voucher_date, voucher_no, period) SELECT '$batch_id', '$interface', 'GL', 'GL', '$client', account, dim_1, dim_2, dim_3, dim_4, dim_5, dim_6, dim_7, '0', ' ', 'GBP', 0, TO_MONEY(cur_amount), TO_MONEY(cur_amount), 0, 0, 0, 0, description, ISO2DATE(:file_date), sequence_no, ISO2DATE(:file_date), 1, $period FROM $*tab1 | | | |

## 4.3 File Name Tags

Some ITK methods allow the names of files being handled to be modified at run time, you do this by including one or more of the following tags in the 'Default value' of the parameter that contains the file name.

| Tag | Meaning | Description | Example |
|---|---|---|---|
| <c> | Client | The code of the Business World client in which the process was run | EN |
| <d> | Date | The date on which the process started ("ddmmyy" format) | 141016 |
| <dr> | Date reverse | The date on which the process started ("yymmdd" format) | 161014 |
| <d4> | Date 4 digit year | The date on which the process started ("ddmmyyyy" format) | 14102016 |
| <d4r> | Date 4 digit year reverse | The date on which the process started ("yyyymmdd" format) | 21061014 |
| <dd> | Date day | The day on which the process started | 14 |
| <dmn> | Date month number | The month in which the process started (numeric format) | 10 |
| <dma> | Date month abbrev. | The month in which the process started (abbreviated format) | Oct |
| <dmf> | Date month full | The month in which the process started (full format) | October |
| <dy2> | Date year 2 digits | The year in which the process started (short format) | 16 |
| <dy4> | Date year 4 digits | The year in which the process started (full format) | 2016 |
| <e> | Export | The path of the Business World 'Data Export' folder | C:\Agresso 5.7.1\Data Export |
| <f> | File name | The default report output name | cu04en_100.lis |
| <i> | Import | The path of the Business World 'Data Import' folder | C:\Agresso 5.7.1\Data Import |
| <l> | "Lis" | The path of the Business World 'Report Results' folder | C:\Agresso 5.7.1\Report Results |
| <o> | Order number | The process's order number | 100 |
| <p\|*id*> | Process parameter value | Where *id* is the 'ParamID' of another process parameter of this job. The *id* must **EXACTLY** match that parameter's ID and must not be the name of the current parameter. E.g. <p\|copy_1> is not allowed to appear in the "copy_1" parameter | if the process has a parameter 'abc_level' that is set to "BC" then <p\|abc_level> would be replaced by "BC" |
| <r> | Report name | This is the column headed 'Report' in the maintenance of report printouts screen | CU04 |

| Tag | Meaning | Description | Example |
|---|---|---|---|
| <s\|*id*> | System parameter value | Where *id* is the name of a client, system or common parameter | <s\|MAX_DATE> will be replaced by "31-DEC-2099" |
| <t> | Time | The time at which the process started (in "hhmmss" format) | 130523 |
| <th> | Time hours | The hour in which the process started | 13 |
| <tm> | Time minutes | The minute in which the process started | 05 |
| <ts> | Time seconds | The second in which the process started | 23 |

Nested tags are not allowed, for instance "<s|<r>_FOLDER>" is illegal and will give unexpected results. However if the parameter referred to by a "<p|*id*>" tag has file name tags in its value then those tags will also be replaced.

## Examples

**Example One**

If the entry in a file name parameter's 'Default value' is "C:\temp\BwOut\st_<c><o>.txt" then if the process is run in client "EN" and its order number is 85 the file will be called "st_EN85.txt" in the "C:\temp\BwOut" folder.

**Example 2**

If the entry in a file name parameter's 'Default value' is "C:\BwOut\<c>\REMIT<dy4>-<dmn>-<dd>" then if the process is run in client "W1" on 03/08/2017 then the file will be called "REMIT2017-08-03" in the "C:\BwOut\W1" folder.

**Example 3**

If the following set up is run on a process with one report output:

| ParamID | Default value | Fixed |
|---|---|---|
| **copy_0** | E:\BwOut\<p\|file>.XML | ☒ |
| **file** | EnterYourFileNameHere | ☐ |

Then the user ordering the process will only see the 'file' parameter and if she enters "Mytest" as that parameter's value then the process's report file will be copied to a file called "Mytest.XML" in folder "E:\BwOut".

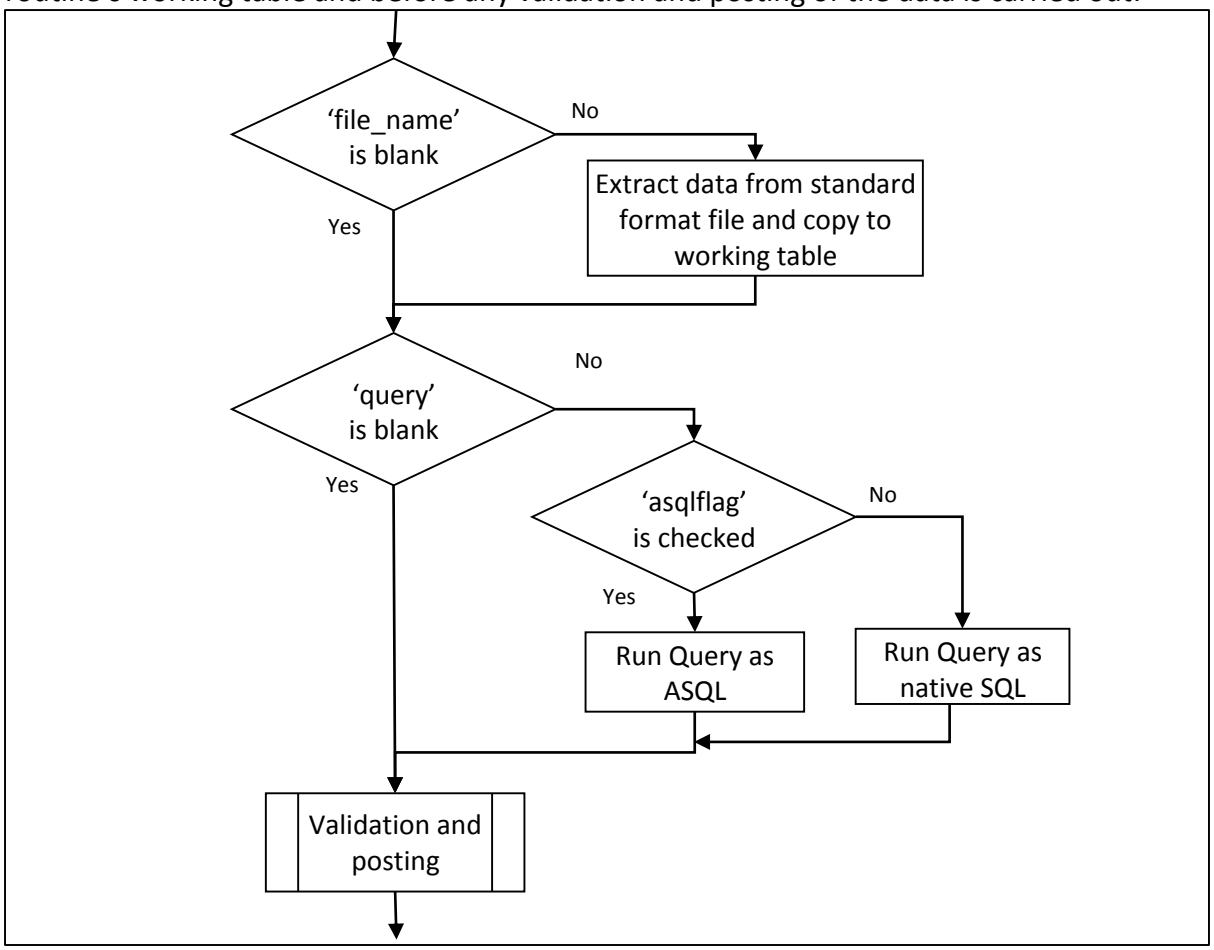
The technique shown in example three can be useful if:

- The structure of the export file's path is complex and users are only allowed to change parts of it.
- The report is intended to be run by an IntellAgent event because on some Business World versions the maximum length of a parameter value that IntellAgent can handle is relatively short (the maximum allowed length varies with Business World version).

# 5 Background Information

## 5.1 SQL Queries in Business World Data Import Routines

Many Business World data import routines have a process parameter called 'query'. Setting the 'query' parameter instructs the import routine to run a block of SQL written using the **Settings ► System Administration ► Reports ► SQL Queries ► Query definition** screen (AG14). This query is run after the contents of the input file have been copied to the import routine's working table and before any validation and posting of the data is carried out.




In its simplest form a query can manipulate the data copied from the file so that it can then be correctly processed by the import routine; this manipulation can include, but is not limited to:

- Completing missing data,
- Mapping external codes to Business World ones,
- Changing the level of detail of the imported data.

A more advanced use of a query is to allow the import routine to process data contained in a file that does not conform to the standard layout for that import routine. This works because if you leave the process parameter that points to the input file (usually called 'file_name') blank then the import routine will not look for a file or try to copy its contents to its working table, so in that case the first thing that the import routine does is run the nominated query.

In this context the main advantage of writing and running your query in ASQL[13] syntax is that ASQL has a "COPY IN" command that allows you to read the contents of a file into a table. The "COPY IN" command can read files consisting of a series of lines (or records) and each record can contain multiple fields. The fields in a record are either:

- "Fixed width" e.g. characters 1 to 4 are field one, 5 to 7 are field two, etc. on every record.
- "Delimited" i.e. each field is separated from the next one by a particular character. This character can be either a:
    - Tab (ASCII[14] code 9)
    - "@" (at sign)
    - "£" (sterling sign)
    - "$" (dollar sign)
    - "%" (percentage symbol)
    - "&" (ampersand)
    - "." (full stop)
    - or ";" (semicolon)

  Using the Tab character is the safest option as this is a non-printing character and is therefore highly unlikely to appear in the data.

The database's native SQL will have a similar facility to "COPY IN" but whilst this may be less restrictive in the file formats it can read it is not as convenient to use: it may, for instance, only be possible to use it from within a stored procedure.

Some import routines do not have a standard file layout at all and rely on you to supply a query that will read the file that you wish to import. Examples of this include:

- 'Import of direct deb. Agreement' (CU08)
- 'Import bank statement' (CB05)
- 'Batch input of employees/resources' (PR43)

Queries are usually named after the process that invokes them, hence a query run in 'Batch input transactions from external system' (GL07) will be called a "GL07 query", similarly "CU08 queries", "LG04 queries", etc. Queries can also be written that appear as menu items in their own right and these queries are run by a server process called AG16, these queries are "AG16 queries", because of this all queries are sometimes referred to as AG16 queries whether they are run by AG16, GL07 or some other process.

**WARNING**

As I have explained elsewhere in this document; the use of queries (AG16 or otherwise) is severely restricted in a SaaS environment.

[13] *Agresso Structured Query Language.*
[14] *See section 5.2.*

## 5.2 The ASCII Character Set

### 5.2.1 Overview

**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange (ASCII) is one of many character-encoding systems that are used to represent text in computers, telecommunications equipment, and other devices. The list of characters that a character-encoding system can portray is called its "character set": hence "the ASCII character set".

The first edition of the ASCII standard was published in 1963; it underwent a major revision in 1967, and experienced its most recent update in 1986. Because of its longevity ASCII is often used, despite its restrictions, as a "lowest common denominator" when communicating between computers with different operating systems or hardware architectures.

The ASCII character set contains:

- Those characters that are available on a North-American keyboard:
  - Numbers,
  - Upper-case Latin (i.e. English) letters,
  - Lower-case Latin letters,
  - Latin punctuation marks,
  - A selection of symbols.
- Some non-printing characters (represented below as abbreviations of their name: $^{X}_{YZ}$):
  - The 'Escape' ($^{E}_{SC}$), 'Backspace' ($^{B}_{S}$), 'Tab' ($^{H}_{T}$), 'Carriage-return' ($^{C}_{R}$ + $^{L}_{F}$) and 'Delete' ($^{D}_{EL}$) keys from a North-American or European keyboard,
  - Some printer control characters (mostly obsolete),
  - Modem control characters (obsolete in most of the world now).

Being a seven-bit encoding system it has only 128 characters in its character set and these characters have codes ranging from 0 to 127, therefore if a character has a code higher than 127 it is a non-ASCII character. Globally there are thousands of non-ASCII characters, examples of these include:

- Letters and logograms from non-Latin alphabets (Arabic, Chinese, Cyrillic, Greek, etc.),
- Non-Latin punctuation marks («, ¿, etc),
- Mathematical symbols (±, ÷, etc),
- Printer's ornaments (also known as "Dingbats"),
- Box drawing characters.

These non-ASCII characters are included in a wide range of encoding systems and in many cases the first 128 characters of each system's character set is the same as the ASCII codes, when this is true it makes that encoding system "ASCII compatible."

**WARNING**

Do not confuse “ASCII encoding” with “ANSI encoding”:

- ANSI is an eight bit encoding system most commonly used by Microsoft® Windows®.
- ANSI is sometimes inaccurately (and ambiguously) called “eight bit ASCII” but it is not a true standard because the content of its character set depends on which region or “code page” your copy of Windows is using.
- Because ANSI is an eight-bit encoding system its character set contains a total of 256 characters; the first 128 characters are the same as the ASCII codes but there are many region specific letters and punctuation marks in the remaining 128 characters.
- Thus the non-ASCII characters in an ANSI encoded file created by a computer running Windows in the Czech Republic (a Cyrillic region) will be “garbled” if the file is then processed by a computer running Windows in Britain.
- The version of ANSI used in the North American and Western European regions is more correctly (and unambiguously) called “Windows-1252”.

### 5.2.2 A List of the ASCII Characters

| Encoding | Character | Encoding | Character | Encoding | Character | Encoding | Character | Encoding | Character | Encoding | Character | Encoding | Character | Encoding | Character |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NUL | 1 | SOH | 2 | STX | 3 | ETX | 4 | EOT | 5 | ENQ | 6 | ACK | 7 | BEL |
| 8 | BS | 9 | HT | 10 | LF | 11 | VT | 12 | FF | 13 | CR | 14 | SO | 15 | SI |
| 16 | DLE | 17 | DC1 | 18 | DC2 | 19 | DC3 | 20 | DC4 | 21 | NAK | 22 | SYN | 23 | ETB |
| 24 | CAN | 25 | EM | 26 | SUB | 27 | ESC | 28 | FS | 29 | GS | 30 | RS | 31 | US |
| 32 |  | 33 | ! | 34 | " | 35 | # | 36 | $ | 37 | % | 38 | & | 39 | ' |
| 40 | ( | 41 | ) | 42 | * | 43 | + | 44 | , | 45 | - | 46 | . | 47 | / |
| 48 | 0 | 49 | 1 | 50 | 2 | 51 | 3 | 52 | 4 | 53 | 5 | 54 | 6 | 55 | 7 |
| 56 | 8 | 57 | 9 | 58 | : | 59 | ; | 60 | < | 61 | = | 62 | > | 63 | ? |
| 64 | @ | 65 | A | 66 | B | 67 | C | 68 | D | 69 | E | 70 | F | 71 | G |
| 72 | H | 73 | I | 74 | J | 75 | K | 76 | L | 77 | M | 78 | N | 79 | O |
| 80 | P | 81 | Q | 82 | R | 83 | S | 84 | T | 85 | U | 86 | V | 87 | W |
| 88 | X | 89 | Y | 90 | Z | 91 | [ | 92 | \ | 93 | ] | 94 | ^ | 95 | _ |
| 96 | ` | 97 | a | 98 | b | 99 | c | 100 | d | 101 | e | 102 | f | 103 | g |
| 104 | h | 105 | i | 106 | j | 107 | k | 108 | l | 109 | m | 110 | n | 111 | o |
| 112 | p | 113 | q | 114 | r | 115 | s | 116 | t | 117 | u | 118 | v | 119 | w |
| 120 | x | 121 | y | 122 | z | 123 | { | 124 | \| | 125 | } | 126 | ~ | 127 | DEL |

The character whose code is 32 is the ‘Space-bar’ character.

Please note that the only currency symbol that is an ASCII character is “$”, or to put it another way the “£” and the “€” symbols are both non-ASCII characters.

### 5.2.3 Why is Any of This Relevant to ITK?

The main reason that some of the ITK methods have an ‘asc_only’ (or a similarly named) parameter is to enable you to remove the “Byte Order Mark” (BOM) from a file. The remainder of this section provides an explanation of the preceding sentence.

Many interface files today are encoded using the UTF-8 encoding system which is a form of Unicode[15]. UTF-8 is ASCII compatible so unless there it has a BOM present a UTF-8 encoded file whose “payload” consists solely of ASCII characters is indistinguishable from an ASCII encoded file. According to the Unicode standard a BOM is optional on UTF-8 encoded files and if present simply confirms that the file is UTF-8 encoded. If a BOM is present then it is the first two bytes of the file and both bytes are non-ASCII characters.

It can be difficult to manually determine if there is a BOM in a file because if, for instance, you decide to look for it by opening the file in ‘Notepad’ you will not see it even if there is one, this is because ‘Notepad’ hides it deliberately; so try using a different text editor.

ITK can cater for the following situations:

1. If a Business World server process has problems reading a TSV file created by the ‘ConvCsv’, ‘ConvCfr’ or ‘ConvFsr’ methods then it is possible that this is due to the file having a BOM, if you have confirmed that this is the cause of the problem then adding a “checked” ‘asc_only’ parameter will remove the BOM from the TSV file which will solve the problem.
2. If a Business World server process has problems reading a file created by the ‘BiQuery’ method then it is possible that this is due to the file having a BOM, if you have confirmed that this is the cause of the problem then adding a “checked” ‘bi_asc_only’ parameter will remove the BOM from the file which will solve the problem.
3. If the receiving system objects to the presence of a BOM in a file exported from Business World using the ‘Export’ method then adding the corresponding “checked” ‘asc_only_*n*’ parameter will remove the BOM from the file which will allow the receiving system to read the file.
4. If the receiving system objects to the presence of a BOM in a file exported from Business World using the ‘BeQuery’ method then adding a “checked” ‘be_asc_only’ parameter will remove the BOM from the file which will allow the receiving system to read the file.

Options one and two are rarely an issue today, because recent Business World releases are generally quite good at determining the encoding of an import file.

Options three and four are still relevant because there are still third-party systems around that do not fully support Unicode so the presence of a BOM in a Business World export file can be an issue for these systems.

---

[15] *Unicode is a computing industry standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems.*
From WIKIPEDIA.

**WARNING**

The database in recent Business World releases supports Unicode which means that foreign company names, personal names and place names stored in Business World may contain non-ASCII characters such as accented vowels and additional consonants.

Checking the 'Export' method's 'asc_only_*n*' parameter or the 'BeQuery' method's 'be_asc_only' parameter will remove these non-ASCII characters from the export file as well as the BOM, which is "a good thing" if the receiving system only supports the ASCII character set.