Server Process Toolkit for Interfaces Configuration Guide

Version 2.9.0

Contents

Introduction

About This Manual

This is one of the two manuals for the AGRESSO Business World (ABW) Server Process Toolkit for Interfaces (InterfaceTools). These manuals are:

• Installation Guide	Describes how to install the software.
• Configuration Guide	Describes how to configure the software, so that it operates the way you require.

What is it For?

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

- Standard import routines with 'query' facilities (CS15, GL07, LG04, etc)
- AG16 queries used to import data (COPY IN, COPY FROM)
- AG16 queries used to export data (COPY TO)
- ARW reports to export data (more complex formats)

Six 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 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(s) twice.
- On an update of an existing row "Master file" import routines such as CS15 do not allow you to blank out columns or remove addresses or relation values (where available)
- The ASQL 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

This toolkit addresses these issues by providing:

- A batch-initialise server exit that will convert "Comma Separated Value" (CSV) files to Tab Separated Value (TSV) files that can then be read by the 'COPY IN' ASQL command in the process' query
- A batch-initialise server exit that will convert "Counted Fields per Record" (CFR) files to Tab Separated Value (TSV) files that can then be read by the 'COPY IN' ASQL command in the process' query.
  See 'CFR to TSV File Conversion' for a description of this file format

- A batch-initialise server exit that will convert “Form-feed Separated Record” (FSR) files to Tab Separated Value (TSV) files that can then be read by the ‘COPY IN’ ASQL command in the process’ query.
  See ‘CFR to TSV File Conversion’ for a description of this file format
- A server exit that can be called at batch-end to delete (or move to the “scratch” folder) non-standard import files or at batch-initialise to delete export files
- A report or batch-end server exit 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
- A batch-end server exit that allows you to copy or move output files created by an ARW (for instance) out of the Report Results folder and give them a new name
- A batch-end server exit that allows you to run additional reports and also copy or move the output files
- A batch-initialise or batch-end server exit that runs an AG16 query
- A batch-initialise or batch-end server exit 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, Agresso or third party)
- A number of exits can split XML output into multiple files at a defined element and also make the payload “XML safe”

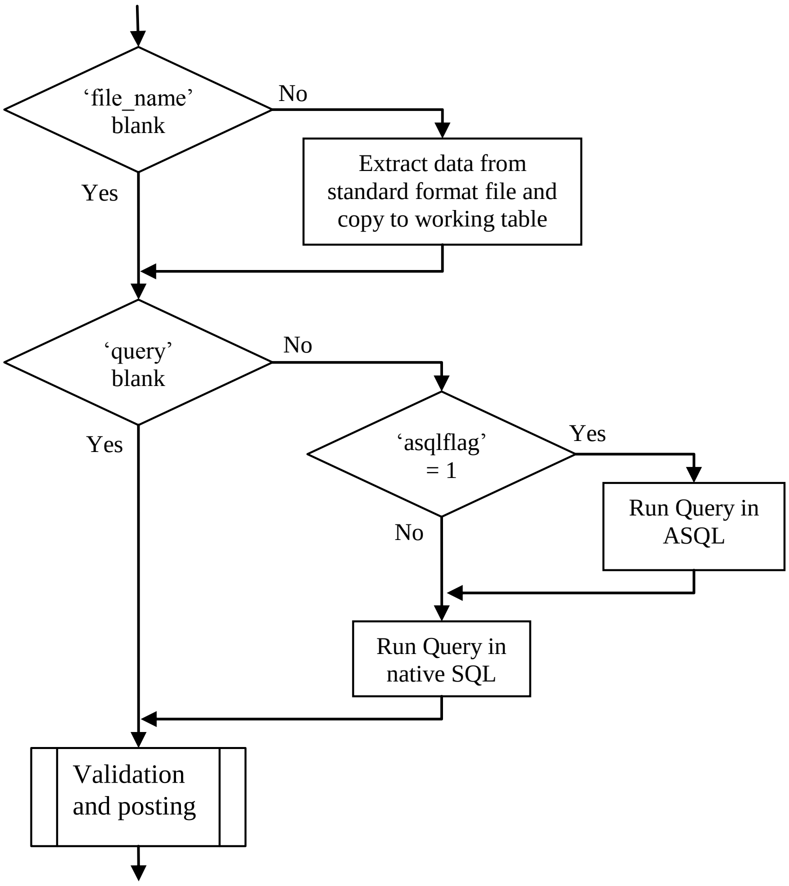
In some versions of ABW 5.5 when you use any of the above batch-end server exits on a server process that has produced no printout then you may get an error:
“Error -2147417848: Method ‘~’ of object ‘~’ failed”,
the toolkit provides a batch-initialise server exit to resolve this problem.

SQL Queries in ABW Data Import Routines

Most AGRESSO Business World (ABW) 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 the validation and posting of this 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 consist of one or more of the following:

- Completing missing data
- Mapping external codes to ABW ones
- Changing the level of detail of the imported data
- Anything else I haven’t thought of

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 ‘file_name’) blank then the import routine will not look for a file and try to copy its contents to its working table, so the first thing that the import routine does is run the nominated query in either ASQL or native SQL syntax.



The ASQL language has a "COPY IN" command that allows you to read the contents of a file into a table. COPY IN 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
- or "delimited" i.e. each field is separated from the next one by a particular character. This character can be either a:

Tab	(ASCII 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 have to be invoked from within a stored procedure

Some import routines do not have a standard file layout at all and rely on you supplying a query to 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.

CSV to TSV File Conversion

Introduction

See ‘SQL Queries in ABW Data Import Routines’ for an overview of using queries to import data.

For many years there has been a de-facto standard for data transfer files known as comma separated value (CSV), unfortunately this format cannot be read by the COPY IN command. This fact has been a minor implementation issue on a number of occasions.

This server exit will take one or more existing CSV files in the AGRESSO_IMPORT folder and translate them into new files where the fields are tab-delimited, and processes text fields so that they are not enclosed by apostrophes or speech marks and removes 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 tab-delimited versions of the CSV files and finally it moves the original CSV files to the AGRESSO_SCRATCH folder, or the folder pointed to by the ‘TEMP’ Windows environment variable if AGRESSO_SCRATCH is not set. The new tab-separated files will have a “tsv” extension and will be 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 new “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. Hence:

- “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 below 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!).

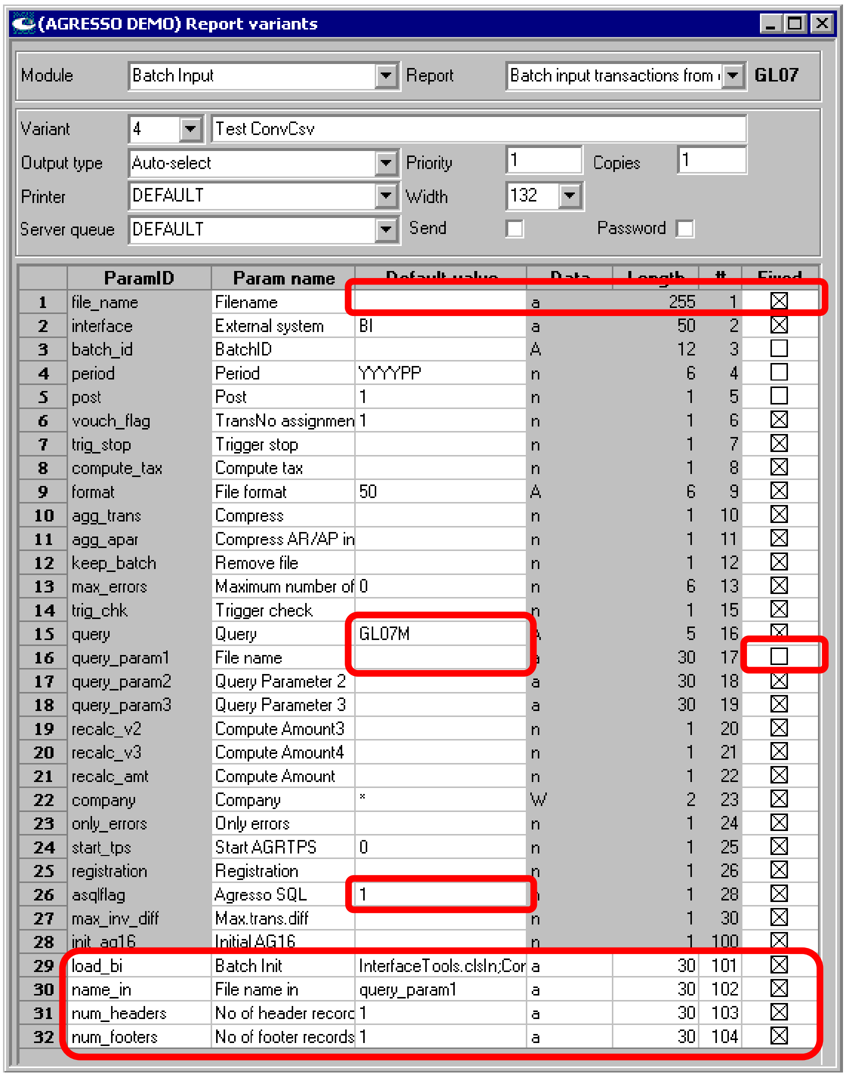
Similarly the table column that the additional field that the program generates is going to be copied into (‘rec_type’ in the query below) must be big enough to hold at least the expected number of detail records, if you could have a million detail records then the column must be at least eight characters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

CSV-Like Files

The true CSV file is one of a family of similar files that all follow the same rules (text fields in quotes, doubled quotes inside quoted text fields). Other members of this family use different characters to separate fields, the tilde "~" for example; this server exit can also be used to process these files.

Set Up

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



Module: Batch Input
Report: Batch input transactions from (GL07

Variant: 4 Test ConvCsv
Output type: Auto-select
Priority: 1
Copies: 1
Printer: DEFAULT
Width: 132
Server queue: DEFAULT
Send: □
Password: □

	ParamID	Param name	Default value	Data	Length	#	Fixed
1	file_name	Filename		a	255	1	☒
2	interface	External system	BI	a	50	2	☒
3	batch_id	BatchID		A	12	3	□
4	period	Period	YYYYPP	n	6	4	□
5	post	Post	1	n	1	5	□
6	vouch_flag	TransNo assignmen	1	n	1	6	☒
7	trig_stop	Trigger stop		n	1	7	☒
8	compute_tax	Compute tax		n	1	8	☒
9	format	File format	50	A	6	9	☒
10	agg_trans	Compress		n	1	10	☒
11	agg_apar	Compress AR/AP in		n	1	11	☒
12	keep_batch	Remove file		n	1	12	☒
13	max_errors	Maximum number of	0	n	6	13	☒
14	trig_chk	Trigger check		n	1	15	☒
15	query	Query	GL07M	A	5	16	☒
16	query_param1	File name		a	30	17	□
17	query_param2	Query Parameter 2		a	30	18	☒
18	query_param3	Query Parameter 3		a	30	19	☒
19	recalc_v2	Compute Amount3		n	1	20	☒
20	recalc_v3	Compute Amount4		n	1	21	☒
21	recalc_amt	Compute Amount		n	1	22	☒
22	company	Company	*	W	2	23	☒
23	only_errors	Only errors		n	1	24	☒
24	start_tps	Start AGRTPS	0	n	1	25	☒
25	registration	Registration		n	1	26	☒
26	asqlflag	Agresso SQL	1	n	1	28	☒
27	max_inv_diff	Max.trans.diff		n	1	30	☒
28	init_ag16	Initial AG16		n	1	100	☒
29	load_bi	Batch Init	InterfaceTools.clsIn;Cor	a	30	101	☒
30	name_in	File name in	query_param1	a	30	102	☒
31	num_headers	No of header recorc	1	a	30	103	☒
32	num_footers	No of footer records	1	a	30	104	☒

Parameter 'file_name'

Must be blank and fixed

Parameter 'query'

Must be set to the appropriate query's name

Parameter 'query_param1' (or similar)

Change the name to 'File name" and unfix it

Parameter 'asqlflag'

Change to a 1 if your query is written in ASQL

New Parameter 'load_bi'

Set 'Default value' to "InterfaceTools.clsIn;ConvCsv"
'Data' to 'a'
'Length' to 30
'#' to On report variants but not user defined reports the '#' entry must be greater than or equal to "100"
and 'Fixed' to checked

New Parameter 'name_in'

Set 'Default value' to 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 the commas are ignored
'Data' to 'a'
'Length' to 30
'#' to one more than the '#' value of the 'load_bi' parameter
and 'Fixed' to checked

New Parameter 'separator'

Set 'Default value' as follows:

- If all files in 'name_in' have the same field separator character then set 'Default value' to that single character
- If the files in 'name_in' have different field separator characters then 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'

This parameter is optional if it is not present then all files in 'name_in' are assumed to use a comma as their field separator character

If you wish to explicitly define a comma as the field separator character then you must enclose it in apostrophes thus: ','

New Parameter 'num_headers'

Set 'Default value' as follows:

- If all files in 'name_in' have the same number of header records then set 'Default value' to that single number
- If the files in 'name_in' have different numbers of header records then 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'

'Data' to 'a'
'Length' to 30
'#' to one more than the '#' value of the 'name_in' parameter
and 'Fixed' to checked

New Parameter 'num_footers'

Set 'Default value' as follows:

- If all files in 'name_in' have the same number of footer records then set 'Default value' to that single number

- If the files in ‘name_in’ have different numbers of footer records then 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’

‘Data’ to ‘a’
‘Length’ to 30
‘#’ to one more than the ‘#’ value of the ‘num_headers’ parameter
and ‘Fixed’ to checked

New Parameter ‘asc_only’

Set ‘Default value’ as follows:

- If all the files in ‘name_in’ contain non-ASCII characters and you want them removed then set ‘Default value’ to “Y”
- If none of the files in ‘name_in’ don’t contain non-ASCII characters or they do but you don’t want them removed then set ‘Default value’ to “N”. This is the default setting if the parameter is missing or left blank
- If you want to remove non-ASCII characters from only some of the files in ‘name_in’ then set ‘Default value’ to a comma separated list of Ys and Ns, , 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’

‘Data’ to ‘A’
‘Length’ to 30
‘#’ to one more than the ‘#’ value of the ‘num_footers’ parameter
and ‘Fixed’ to checked

A non-ASCII character is one whose encoding value is greater than 127. The ASCII character set is primarily that which appears on an English or American typewriter keyboard (numbers, upper and lower case English letters and various symbols and punctuation marks) plus 33 control codes (mainly obsolete printer and modem control characters, but also CR, LF and TAB). There are literally thousands of non-ASCII characters

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

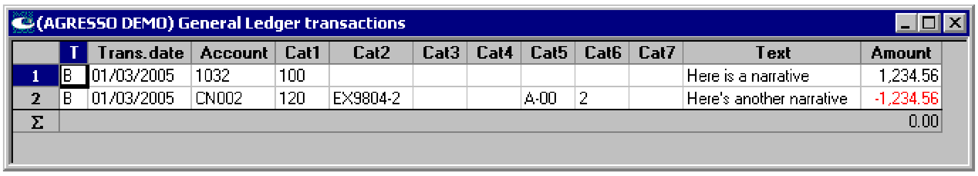
Line	Contents
1	01/03/2005
2	1032,100,’’,’’,’’,’’,’’,’’,1234.56,’Here is a narrative’
3	‘CN002’,120,’EX9804-2’,,’’,’A-00’,’2’,’’,-1234.56,’Here’’s another narrative’
4	1234.56

The SQL Query

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			

<table>
<tr><th>SeqNo</th><th>C</th><th>E</th><th>Description</th></tr>
<tr><td>40</td><td>C</td><td>S</td><td>Error flag</td></tr>
<tr><td colspan="4">DEFINE error_test IDENT(6)</td></tr>
<tr><td>50</td><td>C</td><td>S</td><td>Create help table</td></tr>
<tr><td colspan="4">CREATE TABLE $*tab1 AS SELECT SPACE(25) AS account, description, dim_1, dim_2, dim_3, dim_4, dim_5, dim_6, dim_7, SPACE(15) AS cur_amount, SPACE(5) AS rec_type, sequence_no FROM acrbatchinput WHERE 1 = 2</td></tr>
<tr><td>60</td><td>N</td><td>S</td><td>Get data</td></tr>
<tr><td colspan="4">COPY IN IMPORT FILE = ‘$query_param1’, COLSEP = T, TABLE=$*tab1, rec_type, account, dim_1, dim_2, dim_3, dim_4, dim_5, dim_6, dim_7, cur_amount, description</td></tr>
<tr><td>70</td><td>N</td><td>S</td><td>Remove existing batch data</td></tr>
<tr><td colspan="4">DELETE FROM acrbatchinput WHERE batch_id = ‘$batch_id’ AND client = ‘$client’ AND interface = ‘$interface’</td></tr>
<tr><td>80</td><td>N</td><td>S</td><td>Get date</td></tr>
<tr><td colspan="4">SELECT CONCAT(LEFT(LSHIFT(account, 6), 4), CONCAT(LEFT(LSHIFT(account, 3), 2), LEFT(account, 2))) INTO :file_date FROM $*tab1 WHERE rec_type = ‘H1’</td></tr>
<tr><td>90</td><td>N</td><td>S</td><td>Get total dr</td></tr>
<tr><td colspan="4">SELECT TO_MONEY(account) INTO :file_dr FROM $*tab1 WHERE rec_type = ‘F1’</td></tr>
<tr><td>100</td><td>N</td><td>S</td><td>Remove headers and footers</td></tr>
<tr><td colspan="4">DELETE FROM $*tab1 WHERE LEFT(rec_type, 1) IN (‘H’, ‘F’)</td></tr>
<tr><td>110</td><td>N</td><td>S</td><td>Get total of debits</td></tr>
<tr><td colspan="4">SELECT SUM(TO_MONEY(cur_amount)) INTO :calc_dr FROM $*tab1 WHERE TO_MONEY(cur_amount) > 0</td></tr>
<tr><td>120</td><td>N</td><td>S</td><td>Set column name if footer + balance = zero</td></tr>
<tr><td colspan="4">SELECT ‘col_0’ INTO :error_test FROM asysdummy WHERE col_0 = ‘0’ AND :file_dr = :calc_dr</td></tr>
<tr><td>130</td><td>N</td><td>S</td><td>Generate a syntax error if out of balance</td></tr>
<tr><td colspan="4">SELECT :error_test INTO :error_test FROM asysdummy WHERE col_0 = ‘0’</td></tr>
<tr><td>140</td><td>N</td><td>S</td><td>Sequence no</td></tr>
<tr><td colspan="4">SEQUENCE ON $*tab1 (sequence_no, 0)</td></tr>
<tr><td>150</td><td>N</td><td>S</td><td>Create batch</td></tr>
<tr><td colspan="4">INSERT INTO acrbatchinput (batch_id, interface, voucher_type, trans_type, client, account, dim_1, dim_2, dim_3, dim_4, dim_5, dim_6, dim_7, tax_code, tax_system, currency, dc_flag, cur_amount, amount, number_1, value_1, value_2, value_3, description, trans_date, sequence_no, voucher_date, voucher_no, period) SELECT ‘$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</td></tr>
</table>

The Results



(AGRESSO DEMO) General Ledger transactions

	T	Trans.date	Account	Cat1	Cat2	Cat3	Cat4	Cat5	Cat6	Cat7	Text	Amount
1	B	01/03/2005	1032	100							Here is a narrative	1,234.56
2	B	01/03/2005	CN002	120	EX9804-2			A-00	2		Here's another narrative	-1,234.56
Σ												0.00

CFR to TSV File Conversion

Introduction

See 'SQL Queries in ABW Data Import Routines' for an overview of using queries to import data.

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 a 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.

This server exit will take one or more existing CFR files in the AGRESSO_IMPORT folder and translate them into new files where the fields are tab-delimited. It then changes the value of the "file_name" parameters so that the SQL query can process the tab-delimited versions of the CFR files and finally it moves the original CFR files to the AGRESSO_SCRATCH folder, or the folder pointed to by the 'TEMP' Windows environment variable if AGRESSO_SCRATCH is not set. The new tab-separated files will have a "tsv" extension and will be in the AGRESSO_IMPORT folder.

The content of each line on the input file is not inspected by this server exit 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 ("LEFT(LSHIFT(…)..)" 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 new "tsv" file, this field will be of the form:

"*Xn*"

Where *X* is

- "H" on a header record,
- "D" on a data (or detail) record
- "F" on a footer record

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

- "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.

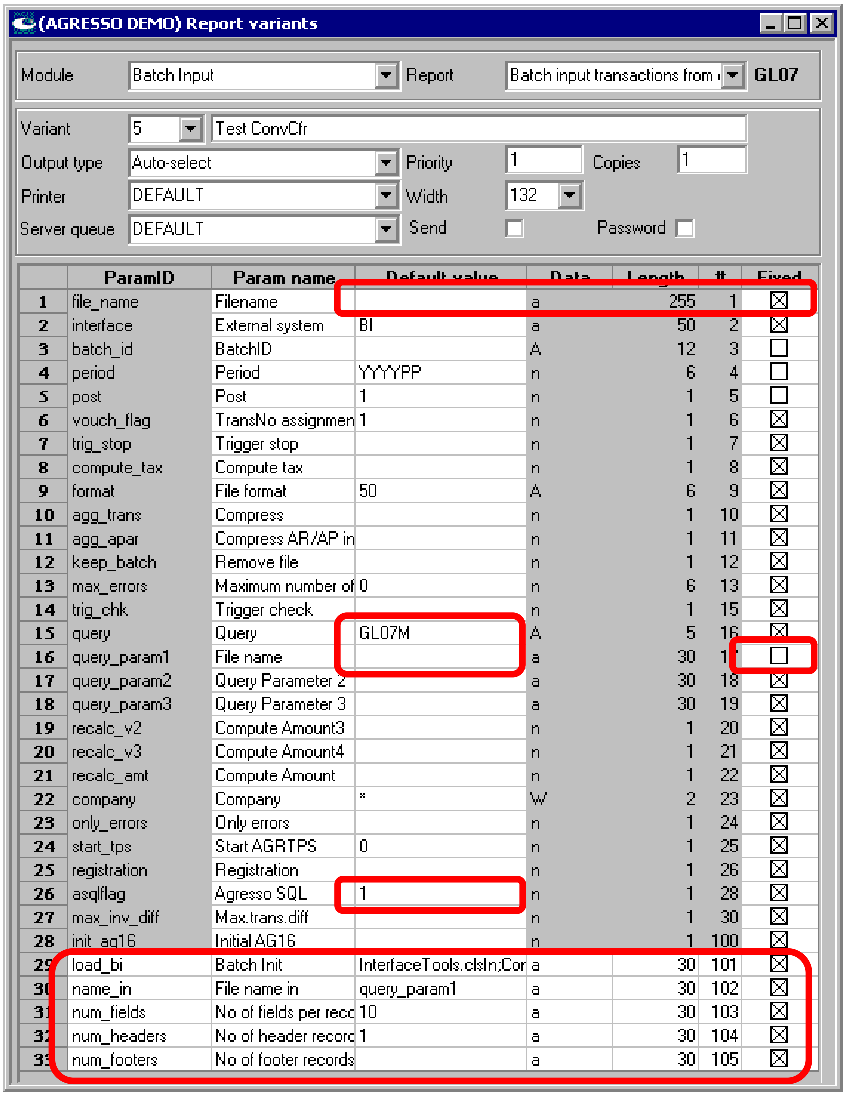
WARNING

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 below 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".

Set Up

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



Module: Batch Input
Report: Batch input transactions from i GL07

Variant: 5 Test ConvCfr
Output type: Auto-select
Priority: 1
Copies: 1
Printer: DEFAULT
Width: 132
Server queue: DEFAULT
Send: □
Password: □

	ParamID	Param name	Default value	Data	Length	#	Fixed
1	file_name	Filename		a	255	1	☒
2	interface	External system	BI	a	50	2	☒
3	batch_id	BatchID		A	12	3	□
4	period	Period	YYYYPP	n	6	4	□
5	post	Post	1	n	1	5	□
6	vouch_flag	TransNo assignmen	1	n	1	6	☒
7	trig_stop	Trigger stop		n	1	7	☒
8	compute_tax	Compute tax		n	1	8	☒
9	format	File format	50	A	6	9	☒
10	agg_trans	Compress		n	1	10	☒
11	agg_apar	Compress AR/AP in		n	1	11	☒
12	keep_batch	Remove file		n	1	12	☒
13	max_errors	Maximum number of	0	n	6	13	☒
14	trig_chk	Trigger check		n	1	15	☒
15	query	Query	GL07M	A	5	16	☒
16	query_param1	File name		a	30	17	□
17	query_param2	Query Parameter 2		a	30	18	☒
18	query_param3	Query Parameter 3		a	30	19	☒
19	recalc_v2	Compute Amount3		n	1	20	☒
20	recalc_v3	Compute Amount4		n	1	21	☒
21	recalc_amt	Compute Amount		n	1	22	☒
22	company	Company	*	W	2	23	☒
23	only_errors	Only errors		n	1	24	☒
24	start_tps	Start AGRTPS	0	n	1	25	☒
25	registration	Registration		n	1	26	☒
26	asqlflag	Agresso SQL	1	n	1	28	☒
27	max_inv_diff	Max.trans.diff		n	1	30	☒
28	init_ag16	Initial AG16		n	1	100	☒
29	load_bi	Batch Init	InterfaceTools.clsIn;Cor	a	30	101	☒
30	name_in	File name in	query_param1	a	30	102	☒
31	num_fields	No of fields per recc	10	a	30	103	☒
32	num_headers	No of header recorc	1	a	30	104	☒
33	num_footers	No of footer records		a	30	105	☒

Parameter 'file_name'

Must be blank and fixed

Parameter 'query'

Must be set to the appropriate SQL query's name

Parameter 'query_param1' (or similar)

Change the name to 'File name" and unfix it

Parameter 'asqlflag'

Change to a 1 if your SQL query is written in ASQL

New Parameter 'load_bi'

Set 'Default value' to "InterfaceTools.clsIn;ConvCfr"
'Data' to 'a'
'Length' to 30
'#' to On report variants but not user defined reports the '#' entry must be greater than or equal to "100"
and 'Fixed' to checked

New Parameter 'name_in'

Set 'Default value' to 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 the commas are ignored
'Data' to 'a'
'Length' to 30
'#' to one more than the '#' value of the 'load_bi' parameter
and 'Fixed' to checked

New Parameter 'num_fields'

Set 'Default value' as follows:

- If all files in 'name_in' have the same number of fields per record then set 'Default value' to that single number
- If the files in 'name_in' have different numbers of fields per record then 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'

'Data' to 'a'
'Length' to 30
'#' to one more than the '#' value of the 'name_in' parameter
and 'Fixed' to checked

New Parameter 'num_headers'

Set 'Default value' as follows:

- If all files in 'name_in' have the same number of header records then set 'Default value' to that single number
- If the files in 'name_in' have different numbers of header records then 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'

'Data' to 'a'
'Length' to 30
'#' to one more than the '#' value of the 'num_fields' parameter
and 'Fixed' to checked

New Parameter 'num_footers'

Set 'Default value' as follows:

- If all files in 'name_in' have the same number of footer records then set 'Default value' to that single number
- If the files in 'name_in' have different numbers of footer records then 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'

‘Data’ to ‘a’
‘Length’ to 30
‘#’ to one more than the ‘#’ value of the ‘num_headers’ parameter
and ‘Fixed’ to checked

New Parameter ‘asc_only’

Set ‘Default value’ as follows:

- If all the files in ‘name_in’ contain non-ASCII characters and you want them removed then set ‘Default value’ to “Y”
- If none of the files in ‘name_in’ don’t contain non-ASCII characters or they do but you don’t want them removed then set ‘Default value’ to “N”. This is the default setting if the parameter is missing or left blank
- If you want to remove non-ASCII characters from only some of the files in ‘name_in’ then set ‘Default value’ to a comma separated list of Ys and Ns, , 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’

‘Data’ to ‘A’
‘Length’ to 30
‘#’ to one more than the ‘#’ value of the ‘num_footers’ parameter
and ‘Fixed’ to checked

A non-ASCII character is one whose encoding value is greater than 127. The ASCII character set is primarily that which appears on an English or American keyboard (numbers, upper and lower case English letters and various symbols and punctuation marks) plus 33 control codes (mainly obsolete printer and modem control characters, but also CR, LF and TAB). There are literally thousands of non-ASCII characters

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

The File

Line	Contents
1	01/03/2005
2	1032
3	100
4	
5	
6	
7	
8	
9	
10	1234.56
11	Here is a narrative
12	CN002
13	120
14	EX9804-2
15	
16	
17	A-00
18	2
19	

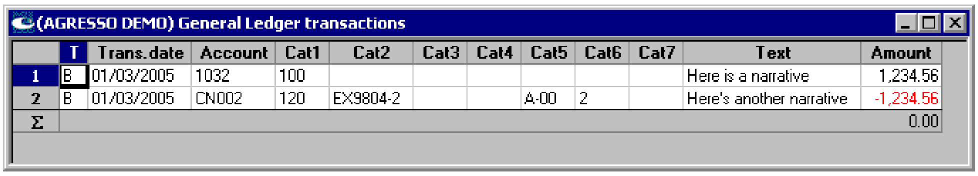
Line	Contents
20	-1234.56
21	Here’s another narrative

There are ten fields per record

The SQL Query

<table>
<tr><th>SeqNo</th><th>C</th><th>E</th><th>Description</th></tr>
<tr><td>10</td><td>C</td><td>S</td><td>Header date</td></tr>
<tr><td colspan="4">DEFINE file_date CHAR(8)</td></tr>
<tr><td>20</td><td>C</td><td>S</td><td>Create help table</td></tr>
<tr><td colspan="4">CREATE TABLE $*tab1 AS SELECT SPACE(25) AS account, description, dim_1, dim_2, dim_3, dim_4, dim_5, dim_6, dim_7, SPACE(15) AS cur_amount, SPACE(5) AS rec_type, sequence_no FROM acrbatchinput WHERE 1 = 2</td></tr>
<tr><td>30</td><td>N</td><td>S</td><td>Get data</td></tr>
<tr><td colspan="4">COPY IN IMPORT FILE = ‘$query_param1’, COLSEP = T, TABLE=$*tab1, rec_type, account, dim_1, dim_2, dim_3, dim_4, dim_5, dim_6, dim_7, cur_amount, description</td></tr>
<tr><td>40</td><td>N</td><td>S</td><td>Remove existing batch data</td></tr>
<tr><td colspan="4">DELETE FROM acrbatchinput WHERE batch_id = ‘$batch_id’ AND client = ‘$client’ AND interface = ‘$interface’</td></tr>
<tr><td>50</td><td>N</td><td>S</td><td>Get date</td></tr>
<tr><td colspan="4">SELECT CONCAT(LEFT(LSHIFT(account, 6), 4), CONCAT(LEFT(LSHIFT(account, 3), 2), LEFT(account, 2))) INTO :file_date FROM $*tab1 WHERE rec_type = ‘H1’</td></tr>
<tr><td>60</td><td>N</td><td>S</td><td>Remove headers</td></tr>
<tr><td colspan="4">DELETE FROM $*tab1 WHERE LEFT(rec_type, 1) = ‘H’</td></tr>
<tr><td>70</td><td>N</td><td>S</td><td>Sequence no</td></tr>
<tr><td colspan="4">SEQUENCE ON $*tab1 (sequence_no, 0)</td></tr>
<tr><td>80</td><td>N</td><td>S</td><td>Create batch</td></tr>
<tr><td colspan="4">INSERT INTO acrbatchinput (batch_id, interface, voucher_type, trans_type, client, account, dim_1, dim_2, dim_3, dim_4, dim_5, dim_6, dim_7, tax_code, tax_system, currency, dc_flag, cur_amount, amount, number_1, value_1, value_2, value_3, description, trans_date, sequence_no, voucher_date, voucher_no, period) SELECT ‘$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</td></tr>
</table>

The Results



	T	Trans.date	Account	Cat1	Cat2	Cat3	Cat4	Cat5	Cat6	Cat7	Text	Amount
1	B	01/03/2005	1032	100							Here is a narrative	1,234.56
2	B	01/03/2005	CN002	120	EX9804-2			A-00	2		Here's another narrative	-1,234.56
Σ												0.00

FSR to TSV File Conversion

Introduction

See ‘SQL Queries in ABW Data Import Routines’ for an overview of using queries to import data.

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 code 12) I have coined the acronym FSR (Form-feed Separated Record) to describe this type of file.

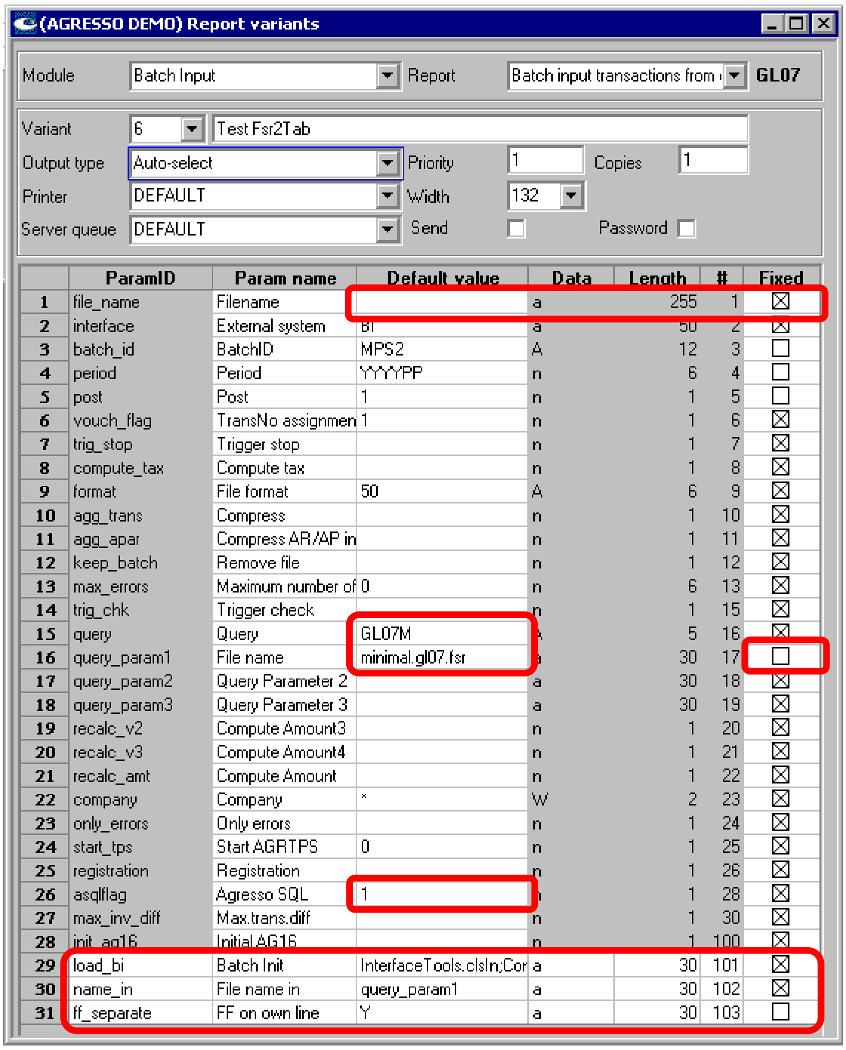
Depending partly on the report writer used to create the file the form-feed character at the beginning of each record is always on a line on its own on others it is simply the first character on the first line of the record.

This server exit will take one or more existing FSR files in the AGRESSO_IMPORT folder and translate them into new files where the fields are tab-delimited. It then changes the value of the “file_name” parameters so that the SQL query can process the tab-delimited versions of the FSR files and finally it moves the original FSR files to the AGRESSO_SCRATCH folder, or the folder pointed to by the ‘TEMP’ Windows environment variable if AGRESSO_SCRATCH is not set. The new tab-separated files will have a “tsv” extension and will be in the AGRESSO_IMPORT folder.

The content of each line on the input file is not inspected by this server exit 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 (“LEFT(LSHIFT(…)..)” in ASQL).

Set Up

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



(AGRESSO DEMO) Report variants

Module	Batch Input	Report	Batch input transactions from	GL07
Variant	6	Test Fsr2Tab		
Output type	Auto-select	Priority	1	Copies 1
Printer	DEFAULT	Width	132	
Server queue	DEFAULT	Send	□	Password □

	ParamID	Param name	Default value	Data	Length	#	Fixed
1	file_name	Filename		a	255	1	☒
2	interface	External system	BI	a	50	2	☒
3	batch_id	BatchID	MPS2	A	12	3	□
4	period	Period	YYYYPP	n	6	4	□
5	post	Post	1	n	1	5	□
6	vouch_flag	TransNo assignmen	1	n	1	6	☒
7	trig_stop	Trigger stop		n	1	7	☒
8	compute_tax	Compute tax		n	1	8	☒
9	format	File format	50	A	6	9	☒
10	agg_trans	Compress		n	1	10	☒
11	agg_apar	Compress AR/AP in		n	1	11	☒
12	keep_batch	Remove file		n	1	12	☒
13	max_errors	Maximum number of	0	n	6	13	☒
14	trig_chk	Trigger check		n	1	15	☒
15	query	Query	GL07M	A	5	16	☒
16	query_param1	File name	minimal.gl07.fsr	a	30	17	□
17	query_param2	Query Parameter 2		a	30	18	☒
18	query_param3	Query Parameter 3		a	30	19	☒
19	recalc_v2	Compute Amount3		n	1	20	☒
20	recalc_v3	Compute Amount4		n	1	21	☒
21	recalc_amt	Compute Amount		n	1	22	☒
22	company	Company	*	W	2	23	☒
23	only_errors	Only errors		n	1	24	☒
24	start_tps	Start AGRTPS	0	n	1	25	☒
25	registration	Registration		n	1	26	☒
26	asqlflag	Agresso SQL	1	n	1	28	☒
27	max_inv_diff	Max.trans.diff		n	1	30	☒
28	init_ag16	Initial AG16		n	1	100	☒
29	load_bi	Batch Init	InterfaceTools.clsIn;Cor	a	30	101	☒
30	name_in	File name in	query_param1	a	30	102	☒
31	ff_separate	FF on own line	Y	a	30	103	□

Parameter 'file_name'

Must be blank and fixed

Parameter 'query'

Must be set to the appropriate SQL query's name

Parameter 'query_param1' (or similar)

Change the name to 'File name" and unfix it

Parameter 'asqlflag'

Change to a 1 if your SQL query is written in ASQL

New Parameter 'load_bi'

Set 'Default value' to "InterfaceTools.clsIn;ConvFsr"
'Data' to 'a'
'Length' to 30

‘#’ to On report variants but not user defined reports the ‘#’ entry must be greater than or equal to “100”

and ‘Fixed’ to checked

New Parameter ‘name_in’

Set ‘Default value’ to 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 the commas are ignored

‘Data’ to ‘a’

‘Length’ to 30

‘#’ to one more than the ‘#’ value of the ‘load_bi’ parameter

and ‘Fixed’ to checked

New Parameter ‘ff_separate’

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.

Set ‘Default value’ as follows:

- If all files in ‘name_in’ need the same ‘ff_separate’ value then set ‘Default value’ to that single value
- If the files in ‘name_in’ need different ‘ff_separate’ values then 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’

‘Data’ to ‘a’

‘Length’ to 30

‘#’ to one more than the ‘#’ value of the ‘name_in’ parameter

and ‘Fixed’ to checked

New Parameter ‘asc_only’

Set ‘Default value’ as follows:

- If all the files in ‘name_in’ contain non-ASCII characters and you want them removed then set ‘Default value’ to “Y”
- If none of the files in ‘name_in’ don’t contain non-ASCII characters or they do but you don’t want them removed then set ‘Default value’ to “N”. This is the default setting if the parameter is missing or left blank
- If you want to remove non-ASCII characters from only some of the files in ‘name_in’ then set ‘Default value’ to a comma separated list of Ys and Ns, , 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’

‘Data’ to ‘A’

‘Length’ to 30

‘#’ to one more than the ‘#’ value of the ‘ff_separate’ parameter

and ‘Fixed’ to checked

A non-ASCII character is one whose encoding value is greater than 127. The ASCII character set is primarily that which appears on an English or American keyboard (numbers, upper and lower case English letters and various symbols and punctuation marks) plus 33 control codes (mainly obsolete printer and modem control characters, but also CR, LF and TAB). There are literally thousands of non-ASCII characters

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.

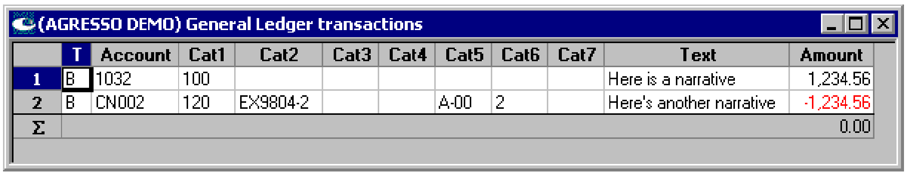
The File

Line	Contents
1	FF
2	1032
3	100
4	
5	
6	
7	
8	
9	
10	1234.56
11	Here is a narrative
12	FF
13	CN002
14	120
15	EX9804-2
16	
17	
18	A-00
19	2
20	
21	-1234.56
22	Here’s another narrative

The SQL Query

<table>
<tr><th>SeqNo</th><th>C</th><th>E</th><th>Description</th></tr>
<tr><td>10</td><td>C</td><td>S</td><td>Create help table</td></tr>
<tr><td colspan="4">CREATE TABLE $*tab1 AS SELECT account, description, dim_1, dim_2, dim_3, dim_4, dim_5, dim_6, dim_7, SPACE(15) AS cur_amount, sequence_no FROM acrbatchinput WHERE 1 = 2</td></tr>
<tr><td>20</td><td>N</td><td>S</td><td>Get data</td></tr>
<tr><td colspan="4">COPY IN IMPORT FILE = $query_param1, COLSEP = T, TABLE=$*tab1, account, dim_1, dim_2, dim_3, dim_4, dim_5, dim_6, dim_7, cur_amount, description</td></tr>
<tr><td>30</td><td>N</td><td>S</td><td>Remove existing batch data</td></tr>
<tr><td colspan="4">DELETE FROM acrbatchinput WHERE batch_id = ‘$batch_id’ AND client = ‘$client’ AND interface = ‘$interface’</td></tr>
<tr><td>40</td><td>N</td><td>S</td><td>Sequence no</td></tr>
<tr><td colspan="4">SEQUENCE ON $*tab1 (sequence_no, 0)</td></tr>
<tr><td>50</td><td>N</td><td>S</td><td>Create batch</td></tr>
<tr><td colspan="4">INSERT INTO acrbatchinput (batch_id, interface, voucher_type, trans_type, client, account, dim_1, dim_2, dim_3, dim_4, dim_5, dim_6, dim_7, tax_code, tax_system, currency, dc_flag, cur_amount, amount, number_1, value_1, value_2, value_3, description, trans_date, sequence_no, voucher_date, voucher_no, period) SELECT ‘$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, TODAY, sequence_no, TODAY, 1, $period FROM $*tab1</td></tr>
</table>

The Results



(AGRESSO DEMO) General Ledger transactions

	T	Account	Cat1	Cat2	Cat3	Cat4	Cat5	Cat6	Cat7	Text	Amount
1	B	1032	100							Here is a narrative	1,234.56
2	B	CN002	120	EX9804-2			A-00	2		Here's another narrative	-1,234.56
Σ											0.00

Delete or Move Input Files

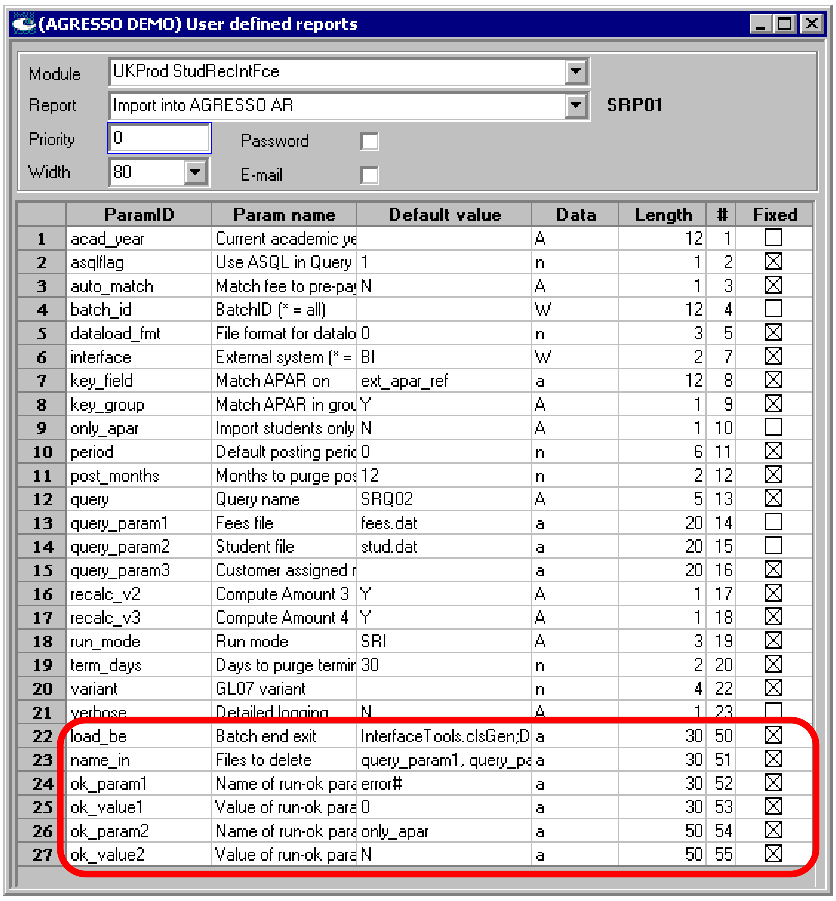
Introduction

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 server exits allow you to either delete the file(s) or move them to the ABW “Scratch” folder. A may be used in combination with ConvCsv, ConvCfr or ConvFsr documented earlier.

Set Up

In order to use one these functions on a process you must set up a variant with two or more additional parameters or add these parameters to your ‘User defined report’. The example is for the ‘Integrated Financials Interface for Student Records’ process “Import into AGRESSO AR” and deletes the files but the same principle applies to moving the files and to any other server process.



Module	UKProd StudRecIntFce
Report	Import into AGRESSO AR — SRP01
Priority	0
Password	□
Width	80
E-mail	□

	ParamID	Param name	Default value	Data	Length	#	Fixed
1	acad_year	Current academic ye		A	12	1	□
2	asqlflag	Use ASQL in Query	1	n	1	2	☒
3	auto_match	Match fee to pre-pa	N	A	1	3	☒
4	batch_id	BatchID (* = all)		W	12	4	□
5	dataload_fmt	File format for datalo	0	n	3	5	☒
6	interface	External system (* =	BI	W	2	7	☒
7	key_field	Match APAR on	ext_apar_ref	a	12	8	☒
8	key_group	Match APAR in grou	Y	A	1	9	☒
9	only_apar	Import students only	N	A	1	10	□
10	period	Default posting peric	0	n	6	11	☒
11	post_months	Months to purge pos	12	n	2	12	☒
12	query	Query name	SRQ02	A	5	13	☒
13	query_param1	Fees file	fees.dat	a	20	14	□
14	query_param2	Student file	stud.dat	a	20	15	□
15	query_param3	Customer assigned r		a	20	16	☒
16	recalc_v2	Compute Amount 3	Y	A	1	17	☒
17	recalc_v3	Compute Amount 4	Y	A	1	18	☒
18	run_mode	Run mode	SRI	A	3	19	☒
19	term_days	Days to purge termir	30	n	2	20	☒
20	variant	GL07 variant		n	4	22	☒
21	verbose	Detailed logging	N	A	1	23	□
22	load_be	Batch end exit	InterfaceTools.clsGen;D	a	30	50	☒
23	name_in	Files to delete	query_param1, query_pa	a	30	51	☒
24	ok_param1	Name of run-ok para	error#	a	30	52	☒
25	ok_value1	Value of run-ok para	0	a	30	53	☒
26	ok_param2	Name of run-ok para	only_apar	a	50	54	☒
27	ok_value2	Value of run-ok para	N	a	50	55	☒

New Parameter 'load_be'

The 'Default value' of this new parameter must have one of two values either "InterfaceTools.clsGen;DelFile" or "InterfaceTools.clsIn;MovFile".

On report variants but not user defined reports the '#' entry must be greater than or equal to "100"

New Parameter 'name_in'

Set 'Default value' to a comma separated list of the names of the parameters containing the file names to delete or move ("query_param1, query_param2" in the example) spaces on either side of the commas are ignored
'Data' to 'a'
'Length' to 30
'#' to one more than the '#' value of the 'load_be' parameter
and 'Fixed' to checked

New Parameters 'ok_param*n*' and 'ok_value*n*'

By default the 'DelFile' and 'MovFile' server exits will carry out their action on the files stored in the 'name_in' parameter whether the data import works or not. If the data import fails and you delete the input file then you may not be able to re-process the data in that file. In order to minimise this problem up to five pairs of optional parameters are provided; 'ok_param1' and 'ok_value1' to 'ok_param5' and 'ok_value5'…

Some server processes set an internal parameter to a specific value to indicate that no errors have occurred, if you set 'ok_param1' to the name of that parameter and 'ok_value1' to the value that means that there were no errors then the server exit will only carry out its action if the named parameter is set to the specified value.

If there are additional parameters that determine whether to delete the file or not even if the run is successful, then these can be defined with up to four additional pairs of parameters. All the defined parameters must have the specified values in order for the files to be deleted or moved.

Processes that set a "run OK" parameter include:

Process	ok_param1	ok_value1	ok_param2	ok_value2
GL07	trig_stat	N	keep_batch	N or 0[1]
SRP01	error#	0 (zero)	only_apar	N

If you do not define any 'ok_param*n*' parameters or leave their values blank then the files will be deleted or moved regardless of whether the data import was successful.

[1] Prior to ABW 5.5 the 'keep_batch' value is "N", from 5.5 onwards it is "0" (zero).

Blank Columns and Remove Addresses and Relation Values

Introduction

The standard "master file" import routines such as 'Batch input customer and supplier information' (CS15) allow you to create and amend address (and relation values from version 5.5) but do not allow you to remove them. In addition some routines will not let you blank out columns on existing rows.

This server exit will blank out specified columns (see The "INI" File 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.

The server exit 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
DataLoad - APAR	Load customers and suppliers
DataLoad - RESO	Load resources

NB
Please be aware that 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 instance to remove a relation value the "delete me" character must be defined as a valid value of that attribute

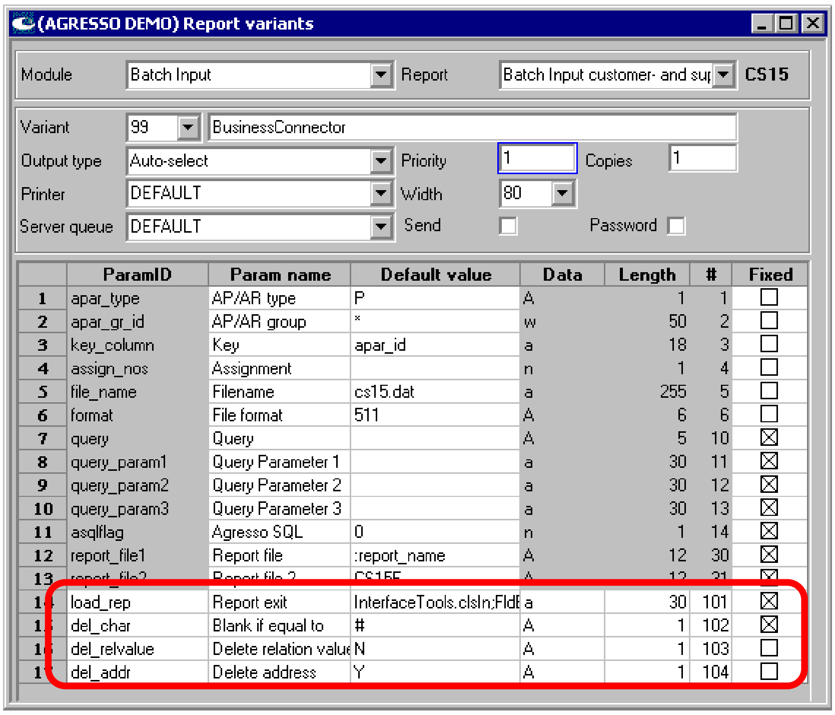
Set Up

Process Parameters

In order to use this function on a process you must set up a variant with one or more additional parameters or add these parameters to your 'User defined report'.
The way this server exit is configured for use depends on which process it is attached to and also which ABW version you are running on;

Process	ABW Version	Server Exit	Notes
CS15	All	load_rep	Prior to 5.4 Sp2 the del_relvalue parameter cannot be set to "Y"
PR43	Pre 5.5	load_be	del_relvalue cannot be set to "Y"
	5.5 option 1	load_bi load_be	Use clsGen;LoadBi (see page 50) del_relvalue cannot be set to "Y"
	5.5 option 2	load_be	Use 'FldBlnkPR43'
DataLoad	All	load_be	Only APAR and RESO are supported

The example is for the 'AGRESSO Financials ► Batch input' process "Batch input customer and supplier information" but the same principle applies to any other server process except the 'AGRESSO Human Resources ► Batch input' process "Update employees/resources" on ABW 5.5.



(AGRESSO DEMO) Report variants

Module	Batch Input	Report	Batch Input customer- and sup	CS15
Variant	99	BusinessConnector		
Output type	Auto-select	Priority	1	Copies 1
Printer	DEFAULT	Width	80	
Server queue	DEFAULT	Send	☐	Password ☐

	ParamID	Param name	Default value	Data	Length	#	Fixed
1	apar_type	AP/AR type	P	A	1	1	☐
2	apar_gr_id	AP/AR group	*	w	50	2	☐
3	key_column	Key	apar_id	a	18	3	☐
4	assign_nos	Assignment		n	1	4	☐
5	file_name	Filename	cs15.dat	a	255	5	☐
6	format	File format	511	A	6	6	☐
7	query	Query		A	5	10	☒
8	query_param1	Query Parameter 1		a	30	11	☒
9	query_param2	Query Parameter 2		a	30	12	☒
10	query_param3	Query Parameter 3		a	30	13	☒
11	asqlflag	Agresso SQL	0	n	1	14	☒
12	report_file1	Report file	:report_name	A	12	30	☒
13	report_file2	Report file 2	CS15F	A	12	31	☒
14	load_rep	Report exit	InterfaceTools.clsIn;FldE	a	30	101	☒
15	del_char	Blank if equal to	#	A	1	102	☒
16	del_relvalue	Delete relation valu	N	A	1	103	☐
17	del_addr	Delete address	Y	A	1	104	☐

New Parameter ‘load_rep’

The ‘Default value’ of this new parameter must have the value “InterfaceTools.clsIn;FldBlnk”.

On PR43 (pre ABW 5.5) and on DataLoad this should be ‘ParamID’ “load_be” not “load_rep” and the ‘Param name’ should be “Batch end exit”

On report variants but not user defined reports the ‘#’ entry must be greater than or equal to “100”

New Parameter ‘del_char’

The ‘Default value’ of this new parameter should be the “delete me” character that you are setting columns, relation values and addresses to.

New Parameter ‘del_relvalue’

Set the ‘Default value’ of this new parameter to “Y” if you want relation values deleted if they are set to the “delete me” character, set it to “N” to leave relation values alone.

This parameter cannot be set to “Y” on CS15 prior to ABW 5.4 SP2 because there was no facility to load relation values before that release.

New Parameter ‘del_addr’

Set the ‘Default value’ of this new parameter to “Y” if you want addresses deleted if their ‘address’ column is set to the “delete me” character, set it to “N” to leave addresses alone

PR43 on ABW 5.5

If you use “FldBlnk” on PR43 in ABW 5.5 then you may get a runtime error:

```
Error -2147417848: Method ‘~' of object '~' failed
```

To avoid this you must do one of two things:

1. Set the “load_be” parameter’s ‘Default value’ to “InterfaceTools.clsIn;FldBlnkPR43”. The other new parameters shouldn’t be defined as this version of the user exit assumes the following ‘Default value’ settings:

ParamID	Default value
del_char	#
del_relvalues	N
del_addr	Y

2. Alternatively, and this now is the preferred approach, set the “load_bi” parameter’s ‘Default value’ to “InterfaceTools.clsGen;LoadBi” and set the other new parameters as normal. See ‘Access Process Parameters on 5.5’ on page 50 for more details.

The “INI” File

CS15 and PR43 will not blank out the value of a column when they are amending a customer, supplier or resource, so a mechanism is needed to control which columns should be blanked and which should not. This mechanism is provided by the ‘InterfaceTools.ini’ file which can be found in ‘Agresso\Server’ (pre ABW 5.5) or ‘Agresso\Bin’ folder (ABW 5.5) on the Business Server. Each of these two processes has a section in the file: CS15’s is the [acsheaderinput] section and PR43’s is the [ahsresources] section. In each case the section consists of a list of the columns on the main table that can be left blank (i.e. are not mandatory), each column name is 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 aka ‘(bank) Account name’ or message_text aka ‘Message’ will be blanked if their value is set to the “delete me” character.

After the list of main-table columns there is a number of repeating groups can appear, each group consists of a line “**key.***table_name*=*key_details*” where *table_name* is a subsidiary table (e.g. agladdress) followed by a number of lines “*table_name***.***column_name*=Y/N”. Each group represents another table that can be updated by this process, e.g. addresses on customers and suppliers; suppliers and addresses on resources.

For example (pre ABW 5.5):

```
keys.agladdress=dim_value-TO_CHAR(apar_id)
agladdress.cc_name=Y
agladdress.e_mail=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.to_name=Y
agladdress.zip_code=N
```

These settings mean that the apar_id (CUST.ID or SUPP.ID) converted to a character string 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 “key...”.

There are two versions of the INI file one for pre ABW 5.5 and one for ABW 5.5 systems, this is because of column name changes and additions in ABW 5.5. The installation routine should have installed the correct version for your system

The INI file only controls the blanking of table columns on updates; it has no affect on the removal of relations or addresses.

Technical Notes

It Works In Two Different Ways

The FldBlnk exit works in two different ways:

- On CS15 it is a ‘load_rep’ exit and uses the acsheaderinput table to see which customers or suppliers to scan for ”delete me” characters
- On all other processes it is a ‘load_be’ exit and scans entities that have been changed in the current client[1], by the correct user ID[2] in the last ten minutes

FldBlnkPR43 works in the non-CS15 mode.

When in the non-CS15 mode it is possible that some “delete me” characters could be missed if a very large batch of entities is processed; i.e. a batch that runs for more than ten minutes.

[1] In FldBlnkPR43 all clients are scanned because it doesn’t know what the current one is!

[2] In PR43 the user ID is “PR43” in other processes it is the user who ordered the process

Delete Output Files

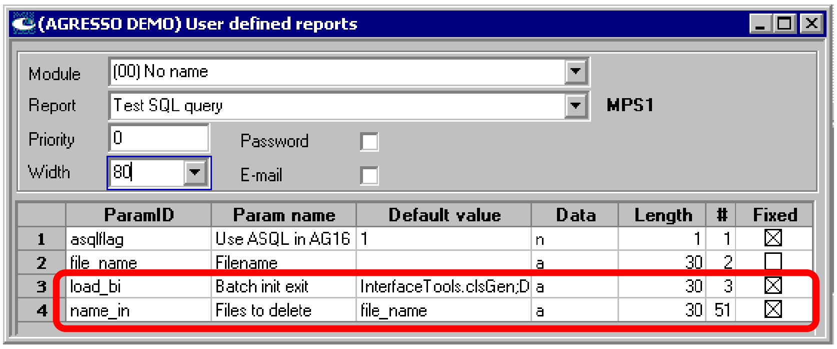
Introduction

When you are using an SQL query to copy ABW data into one or more export files the query may fail if any of the files already exist in the destination folder. The names of these files can be held in one or more process parameters.

This server exit allows you to delete the file(s) before the query runs.

Set Up

In order to use this function in a process you must set up a variant with two additional parameters 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.



(AGRESSO DEMO) User defined reports

Module: (00) No name
Report: Test SQL query   MPS1
Priority: 0   Password ☐
Width: 80   E-mail ☐

	ParamID	Param name	Default value	Data	Length	#	Fixed
1	asqlflag	Use ASQL in AG16	1	n	1	1	☒
2	file_name	Filename		a	30	2	☐
3	load_bi	Batch init exit	InterfaceTools.clsGen;D	a	30	3	☒
4	name_in	Files to delete	file_name	a	30	51	☒

New Parameter 'load_bi'

The 'Default value' of this new parameter must have the value "InterfaceTools.clsGen;DelFile".

On report variants but not user defined reports the '#' entry must be greater than or equal to "100"

New Parameter 'name_in'

Set 'Default value' to a comma separated list of the names of the parameters
containing the file names to delete or move ("file_name" in the example)
spaces on either side of the commas are ignored
'Data' to 'a'
'Length' to 30
'#' to one more than the '#' value of the 'load_bi' parameter
and 'Fixed' to checked

Copy or Move Output File

Introduction

Files created with the ASQL '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 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.

  It is possible to produce a CSV file using COPY TO as long as non 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("',', 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 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

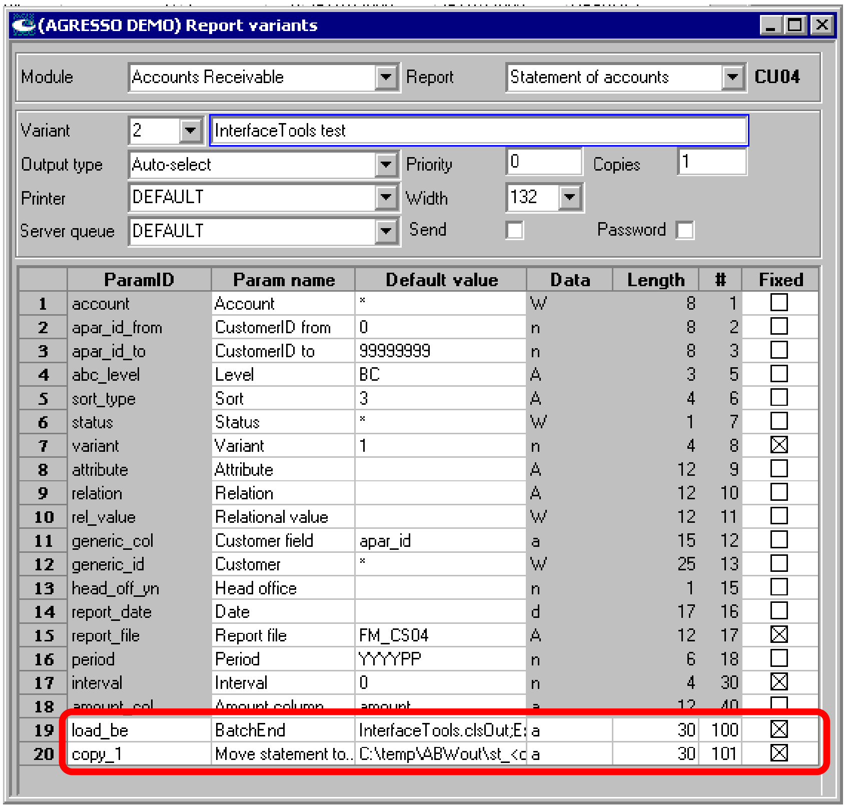
In order to transmit the file to another system it often needs to be stored elsewhere and ideally with a more meaningful name. ARW has an .OUTPUT command that allows you to specify the file name (and optional path) that your report's output be written to instead of the above "lis" file but you can only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

This server exit provides a more flexible solution to the above than the 'OUTPUT command:

- variables can be included anywhere in the file name
- the file name is a process parameter instead of hard-coded in the report

Set Up

In order to use this server exit on a process you must set up a variant with some additional parameters. The example is for Customer statements but the same principle applies to any other server process.



(AGRESSO DEMO) Report variants

Module: Accounts Receivable | Report: Statement of accounts | CU04

Variant: 2 | InterfaceTools test
Output type: Auto-select | Priority: 0 | Copies: 1
Printer: DEFAULT | Width: 132
Server queue: DEFAULT | Send: □ | Password: □

	ParamID	Param name	Default value	Data	Length	#	Fixed
1	account	Account	*	W	8	1	□
2	apar_id_from	CustomerID from	0	n	8	2	□
3	apar_id_to	CustomerID to	99999999	n	8	3	□
4	abc_level	Level	BC	A	3	5	□
5	sort_type	Sort	3	A	4	6	□
6	status	Status	*	W	1	7	□
7	variant	Variant	1	n	4	8	☒
8	attribute	Attribute		A	12	9	□
9	relation	Relation		A	12	10	□
10	rel_value	Relational value		W	12	11	□
11	generic_col	Customer field	apar_id	a	15	12	□
12	generic_id	Customer	*	W	25	13	□
13	head_off_yn	Head office		n	1	15	□
14	report_date	Date		d	17	16	□
15	report_file	Report file	FM_CS04	A	12	17	☒
16	period	Period	YYYYPP	n	6	18	□
17	interval	Interval	0	n	4	30	☒
18	amount_col	Amount column	amount	a	12	40	□
19	load_be	BatchEnd	InterfaceTools.clsOut;E:	a	30	100	☒
20	copy_1	Move statement to..	C:\temp\ABWout\st_<c	a	30	101	☒

New Parameter 'load_be'

This new parameter must be entered exactly as shown, in particular the 'Default value' entry must be "InterfaceTools.clsOut;Export".

On report variants but not user defined reports the '#' entry must be greater than or equal to "100"

New Parameter 'copy_1' (or 'move_1')

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 five are not currently supported. If you wish to move the file instead of copying it then change the **ParamID** to 'move_1'. 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 just define copy_2. You can also mix 'copy' and 'move' parameters on the same variant as long as they are for different sequence numbers.

Note that from ABW 5.5 that the first report output's sequence number is zero not one.

Each of these parameters takes a **Default value** that determines the file path and name that you want to copy or move the file to. The value entered is the path name including the file name.

The **Default value** entry can include fixed text and one or more of the following tags:

Tag	Meaning	Description	Example
<c>	Client	The code of the ABW client in which the report was run	EN

Tag	Meaning	Description	Example
<d>	Date	The date on which the report was run ("ddmmyy" format)	030805
<dr>	Date reverse	The date on which the report was run ("yymmdd" format)	050803
<d4>	Date 4 digit year	The date on which the report was run ("ddmmyyyy" format)	03082005
<d4r>	Date 4 digit year reverse	The date on which the report was run ("yyyymmdd" format)	20050803
<dd>	Date day	The day on which the report was run	03
<dmn>	Date month number	The month in which the report was run (numeric format)	08
<dma>	Date month abbrev.	The month in which the report was run (abbreviated format)	Aug
<dmf>	Date month full	The month in which the report was run (full format)	August
<dy2>	Date year 2 digits	The year in which the report was run (short format)	05
<dy4>	Date year 4 digits	The year in which the report was run (full format)	2005
<e>	Export	The path of the ABW 'Data Export' folder	C:\Agresso 5.4\Data Export
<f>	File name	The default report output name	cu04en_100.lis
<l>	"Lis"	The path of the ABW 'Report Results' folder	C:\Agresso 5.4\Report Results
<o>	Order number	The report'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	In the screen shot above <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
<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 report was run ("hhmmss" format)	130523
<th>	Time hours	The hour in which the report was run	13
<tm>	Time minutes	The minute in which the report was run	05
<ts>	Time seconds	The second in which the report was run	23

One level of folder creation is supported; for example in the 'Example 2' below if both the "AbwOut" and "W1" folders did not exist then an error would occur, whereas if only the "W1" folder doesn't exist then it will be created within the "AbwOut" folder.

If the expanded file path and name equals the original file name then this is only allowed if you are moving the file **AND** either the corresponding 'asc_only_*n*' and/or the 'xml_safe_*n*' parameter is equal to "Y"

From version 2.8.0 the original case of fixed text and substituted values is preserved

Nested tags are not allowed, for instance "<s|<r>_FOLDER>" is illegal and will give unexpected results

In order to stop this server exit from causing serious damage to your system the file name is not allowed to end with ".dll", ".exe" or ".ocx"

Examples

Example One
If the entry in the screen shot is "C:\temp\AbwOut\st_<c><o>.txt" is run in client "EN" and its order number is 85 it will create a file called "st_EN85.txt" in the "C:\temp\AbwOut" folder.

Example 2
If "C:\ AbwOut \<c>\REMIT<dy4>–<dmn>–<dd>", if is run in client "W1" on 03/08/2005 a file called "REMIT2005–08–03" will be created in the "C:\ AbwOut \W1" folder.

Example 3
If the following set up is run on process "CU04" with order number 1234 in a client where a Common, System or Client parameter "XML_OUT_FOLDER" has the value of "E:\AbwOut\" then a file called "ABW-CU04-1234.XML" will be created in folder "E:\AbwOut"

ParamID	Default value
copy_0	<p|folder><p|file>
file	ABW-<r>-<o>.XML
folder	<s|XML_OUT_FOLDER>

This admittedly rather circuitous technique is useful if the report is intended to be run by Agresso IntellAgent as in this case parameter values must be kept short to avoid IntellAgent stopping with an error (the maximum allowed length varies with ABW version)

New Parameter 'asc_only_1'

Set 'Default value' as follows:

- If the 'copy_1' or 'move_1' file contains non-ASCII characters and you want them removed then set 'Default value' to "Y"
- If the 'copy_1' or 'move_1' file doesn't contain non-ASCII characters or it does but you don't want them removed then set 'Default value' to "N". This is the default setting if the parameter is missing or left blank

'Data' to 'A'
'Length' to 1
'#' to one more than the '#' value of the corresponding 'copy_*n*' or 'move_*n*' parameter
and 'Fixed' to checked

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

A non-ASCII character is one whose encoding value is greater than 127. The ASCII character set is primarily that which appears on an English or American keyboard (numbers, upper and lower case English letters and various symbols and punctuation marks) plus 33 control codes (mainly obsolete printer and modem control characters, but also CR, LF and TAB). There are literally thousands of non-ASCII characters

New Parameter 'xml_safe_1'

Set 'Default value' as follows:

- If the 'copy_1' or 'move_1' file contains XML and may contain "unsafe" characters then set 'Default value' to "Y"
- If the 'copy_1' or 'move_1' file doesn't contain XML or it does but you know that there are cannot be any unsafe characters then set 'Default value' to "N". This is the default setting if the parameter is missing or left blank

'Data' to 'A'
'Length' to 1
'#' to one more than the '#' value of the corresponding 'copy_*n*' or 'move_*n*' parameter or the 'asc_only_*n*' parameter if it is present
and 'Fixed' to checked

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

If any the following characters are contained in the payload (or value) of an XML element then this will cause programs to fail to read that XML. If any these "unsafe" characters are present they must be replaced by their "entity" symbol to avoid this problem

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

Setting 'xml_safe' to "Y" will automatically replace any unsafe characters by their entities

New Parameter 'xml_split_1'

Set 'Default value' as follows:

- If the 'copy_1' or 'move_1' XML file contains a repeating element and you wish each occurrence of that repeating element to appear in a separate file (due to restrictions in the receiving system) then set 'Default value' to the name of the repeating element without the "<" or ">" characters or any of its attributes, e.g. "Supplier". The value of this parameter is case sensitive
- Otherwise omit this parameter or set 'Default value' to blank

'Data' to 'a'
'Length' to 30
'#' to one more than the '#' value of the corresponding 'xml_safe_*n*' parameter
and 'Fixed' to checked

You can optionally have a 'xml_split_*n*' parameter for each 'xml_safe_*n*' that is present and its number (_n) must match

If more than one file is created then each file will have a sequence number starting at one.

Example

If the copy_1 parameter is "C:\temp\AbwOut\st_<c><o>.xml" and it is run in client "EN" and its order number is 85 and one file is created in the "C:\temp\AbwOut" folder then its name will be "st_EN85.xml". If three files are created then their names will be "st_EN85_1.xml", "st_EN85_2.xml" and "st_EN85_3.xml".

Additional Reports

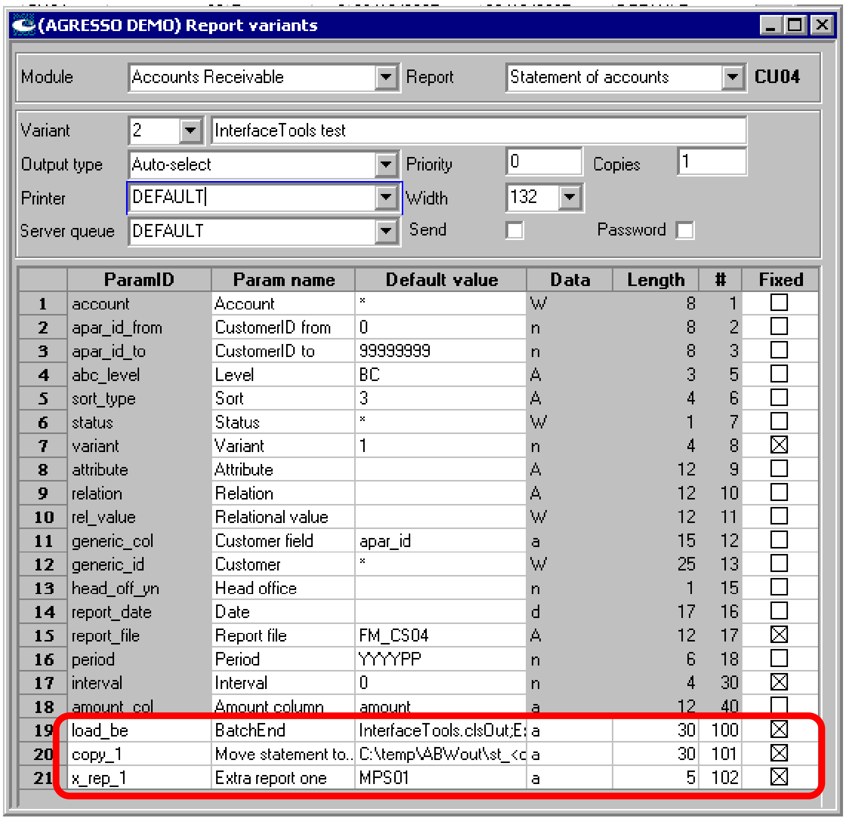
Introduction

Many standard server processes run only one report file, or non at all! Sometimes you might want to produce both a print and an export file from a process and to do that you need to be able to run additional reports.

This server exit gives you the ability to run additional report files as well as the ability to copy or move the files out of the 'Report Results' folder.

Set Up

In order to use this server exit on a process you must set up a variant with some additional parameters. The example is for Customer statements (don't ask) but the same principle applies to any other server process.



(AGRESSO DEMO) Report variants

Module: Accounts Receivable
Report: Statement of accounts
CU04

Variant: 2 InterfaceTools test
Output type: Auto-select
Priority: 0
Copies: 1
Printer: DEFAULT
Width: 132
Server queue: DEFAULT
Send: □
Password: □

	ParamID	Param name	Default value	Data	Length	#	Fixed
1	account	Account	*	W	8	1	□
2	apar_id_from	CustomerID from	0	n	8	2	□
3	apar_id_to	CustomerID to	99999999	n	8	3	□
4	abc_level	Level	BC	A	3	5	□
5	sort_type	Sort	3	A	4	6	□
6	status	Status	*	W	1	7	□
7	variant	Variant	1	n	4	8	☒
8	attribute	Attribute		A	12	9	□
9	relation	Relation		A	12	10	□
10	rel_value	Relational value		W	12	11	□
11	generic_col	Customer field	apar_id	a	15	12	□
12	generic_id	Customer	*	W	25	13	□
13	head_off_yn	Head office		n	1	15	□
14	report_date	Date		d	17	16	□
15	report_file	Report file	FM_CS04	A	12	17	☒
16	period	Period	YYYYPP	n	6	18	□
17	interval	Interval	0	n	4	30	☒
18	amount_col	Amount column	amount	a	12	40	□
19	load_be	BatchEnd	InterfaceTools.clsOut;E	a	30	100	☒
20	copy_1	Move statement to..	C:\temp\ABWout\st_<c	a	30	101	☒
21	x_rep_1	Extra report one	MPS01	a	5	102	☒

New Parameter 'load_be'

This new parameter must be entered exactly as shown, in particular the 'Default value' entry must be "InterfaceTools.clsOut;Extra"

On report variants but not user defined reports the '#' entry must be greater than or equal to "100"

New Parameter 'copy_1' (or 'move_1')

See the parameter of the same name in 'Copy or Move Output File'. The 'asc_only_*n*', 'xml_safe_*n*' and 'xml_split_*n*' can also be used

New Parameter 'x_rep_1'

This new parameter is an example of a group of parameters that you can set up to instruct the server process to run up to five extra reports (x_rep_1 to x_rep_5). The default value is the name of the report to run excluding its file type (ARW, CRW or RPT) The numeric part of these parameters is simply defines the order in which these reports will be run and is not linked in any way to the numeric part of the 'copy_*n*' or 'move_*n*' parameters.

New Parameter 'x_test_1'

This parameter is used to control whether or not the report named in the x_rep_*n* parameter with the same number should be run or not.

Most reports run from help tables created by the process that they are attached to, the names of these help tables are stored in parameters created by the process which are then used in the body of the report; these parameters have names like "hlptab", "helptable", "helptab01", etc. If the process fails to create the help table then it generally does not assign the table name(s) to the parameter(s) and when your report runs "`SELECT … FROM $hlptab t WHERE …`" (or whatever) 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".

If you put the name of the help table parameter into an x_test_*n* parameter then if that parameter has a blank value the report in the corresponding x_rep_*n* parameter will not be run. If you are using more than one help table in your report then put a comma separated list of the names of the parameter containing the help table names into the x_test_*n* parameter and the corresponding report will not be run if any of them have blank values.

In some circumstances the parameter containing the help table name may not be available to the report being called (the report will think its value is blank), using this parameter has the added benefit of overcoming this problem.

The copy, move and x_rep_*n* and x_test_*n* parameters can be entered in any order in the variant.

Run a Query

Introduction

Sometimes when configuring a system you need to be able to run a database update before and/or after a standard server process or report is run. Even if that process has a query parameter it may be that this query happens too early or too late for your database update and also there is only one query.

Three server exits are provided:

Exit name	batch-initialise	batch-end
BiQuery	✓	
BeQuery		✓
RunQuery	✓	✓

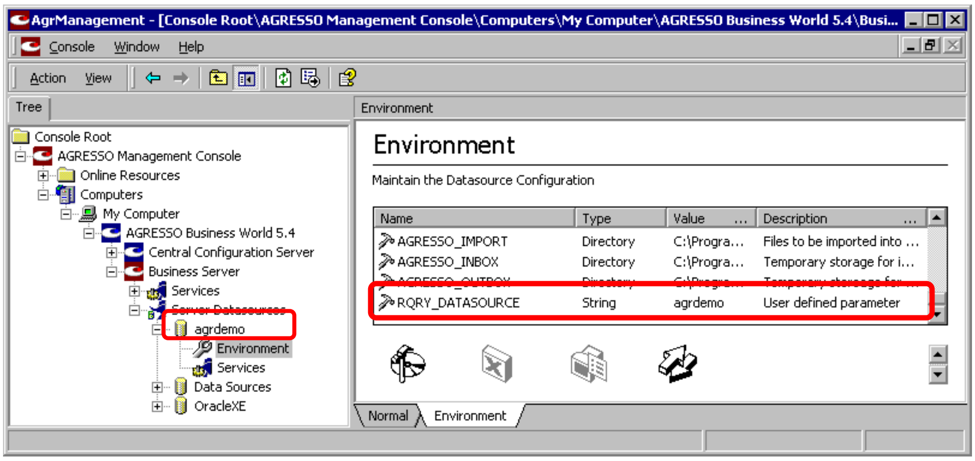
All of these exits run a named AG16 query.

RunQuery is provided to maintain backward compatibility with earlier versions of InterfaceTools and cannot be used to run two queries on the same process, so you should avoid using it when setting up new report variants

Set Up

In order to use any of these server exits on a process you must set up an environment variable and a variant with some additional parameters.

The environment variable must be defined in the AGRESSO Management Console and only needs to be defined once per data source no matter how many processes you use the server exit on.



Environment

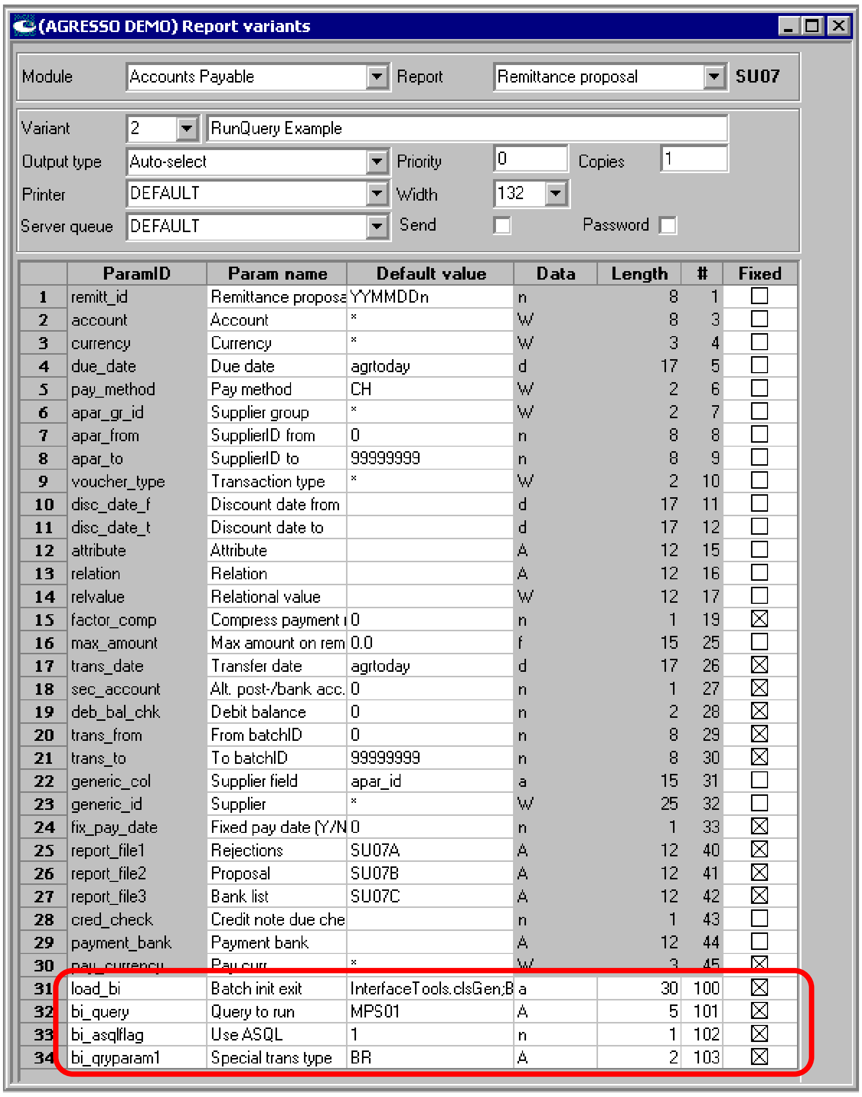
Maintain the Datasource Configuration

Name	Type	Value ...	Description ...
AGRESSO_IMPORT	Directory	C:\Progra...	Files to be imported into ...
AGRESSO_INBOX	Directory	C:\Progra...	Temporary storage for i...
AGRESSO_OUTBOX	Directory	C:\Progra...	Temporary storage for ...
RQRY_DATASOURCE	String	agrdemo	User defined parameter

The screen shot show the ABW 5.4 Management Console but the same principle applies in the ABW 5.5 version.

For each data source that you wish to use the server exit in create an environment variable, its ‘Name’ must be “RQRY_DATASOURCE”, its ‘Type’: “String” and its ‘Value’ should be the name of the Server Datasource, so the ‘RQRY_DATASOURCE’ parameter in the ‘OracleXE’ Datasource in the screen shot would have its ‘Value’ set to “OracleXE”.

The example is for the Remittance Proposal but the same principle applies to any other server process.



(AGRESSO DEMO) Report variants

Module	Accounts Payable	Report	Remittance proposal	SU07

Variant	2	RunQuery Example			
Output type	Auto-select	Priority	0	Copies	1
Printer	DEFAULT	Width	132		
Server queue	DEFAULT	Send	□	Password	□

	ParamID	Param name	Default value	Data	Length	#	Fixed
1	remitt_id	Remittance proposa	YYMMDDn	n	8	1	□
2	account	Account	*	W	8	3	□
3	currency	Currency	*	W	3	4	□
4	due_date	Due date	agrtoday	d	17	5	□
5	pay_method	Pay method	CH	W	2	6	□
6	apar_gr_id	Supplier group	*	W	2	7	□
7	apar_from	SupplierID from	0	n	8	8	□
8	apar_to	SupplierID to	99999999	n	8	9	□
9	voucher_type	Transaction type	*	W	2	10	□
10	disc_date_f	Discount date from		d	17	11	□
11	disc_date_t	Discount date to		d	17	12	□
12	attribute	Attribute		A	12	15	□
13	relation	Relation		A	12	16	□
14	relvalue	Relational value		W	12	17	□
15	factor_comp	Compress payment r	0	n	1	19	☒
16	max_amount	Max amount on rem	0.0	f	15	25	□
17	trans_date	Transfer date	agrtoday	d	17	26	☒
18	sec_account	Alt. post-/bank acc.	0	n	1	27	☒
19	deb_bal_chk	Debit balance	0	n	2	28	☒
20	trans_from	From batchID	0	n	8	29	☒
21	trans_to	To batchID	99999999	n	8	30	☒
22	generic_col	Supplier field	apar_id	a	15	31	□
23	generic_id	Supplier	*	W	25	32	□
24	fix_pay_date	Fixed pay date (Y/N	0	n	1	33	☒
25	report_file1	Rejections	SU07A	A	12	40	☒
26	report_file2	Proposal	SU07B	A	12	41	☒
27	report_file3	Bank list	SU07C	A	12	42	☒
28	cred_check	Credit note due che		n	1	43	□
29	payment_bank	Payment bank		A	12	44	□
30	pay_currency	Pay curr	*	W	3	45	☒
31	load_bi	Batch init exit	InterfaceTools.clsGen;B	a	30	100	☒
32	bi_query	Query to run	MPS01	A	5	101	☒
33	bi_asqlflag	Use ASQL	1	n	1	102	☒
34	bi_qryparam1	Special trans type	BR	A	2	103	☒

New Parameter 'load_bi'

If you want a query to be run at the beginning of the process the 'load_bi', 'bi_query' and 'bi_asqlflag' parameters should be used.

The 'load_bi' parameter must be entered exactly as shown, in particular the 'Default value' entry must be "InterfaceTools.clsGen;BiQuery" or "InterfaceTools.clsGen;RunQuery"

On report variants but not user defined reports the '#' entry must be greater than or equal to "100"

New Parameter 'bi_query'

This parameter should have its 'Default value' set to the name of the AG16 query that you want to be run at the beginning of the process.

Its '#' entry should be set to one more than that of the 'load_bi' parameter.

New Parameter 'bi_asqlflag'

If the 'bi_query' AG16 query is written in ASQL set this parameter's 'Default value' to "1" (one), otherwise set it to "0" (zero)

Its '#' entry should be set to one more than that of the 'bi_query' parameter.

New Parameter 'bi_name_in'

If you wish to use either the 'bi_asc_only' or 'bi_xml_safe' parameters then you must give this parameter a name.

Set 'Default value' to 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 the commas are ignored
'Data' to 'a'
'Length' to 30
'#' to one more than the '#' value of the 'bi_asqlflag' parameter
and 'Fixed' to checked

New Parameter 'bi_asc_only'

Set 'Default value' as follows:

- If all the files in 'bi_name_in' contain non-ASCII characters and you want them removed then set 'Default value' to "Y"
- If none of the files in 'bi_name_in' contain non-ASCII characters or they do but you don't want them removed then set 'Default value' to "N". This is the default setting if the parameter is missing or left blank
- If you want to remove non-ASCII characters from only some of the files in 'bi_name_in' then set 'Default value' to a comma separated list of Ys and Ns, , 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'

'Data' to 'A'
'Length' to 30
'#' to one more than the '#' value of the 'bi_name_in' parameter
and 'Fixed' to checked

A non-ASCII character is one whose encoding value is greater than 127. The ASCII character set is primarily that which appears on an English or American keyboard (numbers, upper and lower case English letters and various symbols and punctuation marks) plus 33 control codes (mainly obsolete printer and modem control characters, but also CR, LF and TAB). There are literally thousands of non-ASCII characters

New Parameter 'bi_xml_safe'

Set 'Default value' as follows:

- If all the files in 'bi_name_in' contain XML and may contain "unsafe" characters then set 'Default value' to "Y"
- If none of the files in 'bi_name_in' contain XML or they do but you know that there are cannot be any unsafe characters then set 'Default value' to "N". This is the default setting if the parameter is missing or left blank
- If some of the files in 'bi_name_in' may contain "unsafe" characters then set 'Default value' to a comma separated list of Ys and Ns, , 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'

'Data' to 'A'
'Length' to 30

‘#’ to one more than the ‘#’ value of the ‘bi_name_in’ parameter or the ‘bi_asc_only’ parameter if it is present

and ‘Fixed’ to checked

If any the following characters are contained in the payload (or value) of an XML element then this will cause programs to fail to read that XML. If any these “unsafe” characters are present they must be replaced by their “entity” symbol to avoid this problem

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

Setting ‘xml_safe’ to “Y” will automatically replace any unsafe characters by their entities

New Parameter ‘bi_xml_split’

Set ‘Default value’ as follows:

- If all the files in ‘bi_name_in’ contain XML and contain the same repeating element and you wish each occurrence of that repeating element to appear in a separate file (due to restrictions in the receiving system) then set ‘Default value’ to the name of the repeating element without the “<” or “>” characters or any of its attributes, e.g. “Supplier”. The value of this parameter is case sensitive
- 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 omit this parameter or set ‘Default value’ to blank.
- If some of the files in ‘bi_name_in’ contain XML and you wish to split different repeating elements into different files or not split them at all then 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 “''”

‘Data’ to ‘a’

‘Length’ to 30

‘#’ to one more than the ‘#’ value of the corresponding ‘xml_safe_*n*’ parameter

and ‘Fixed’ to checked

If more than one file is created then each file will have a sequence number starting at one.

Example

If the parameter pointed to by ‘bi_name_in’ contains “C:\temp\AbwOut\st_<c><o>.xml” and it is run in client “EN” and its order number is 85 and one file is created in the “C:\temp\AbwOut” folder then its name will be “st_EN85.xml”. If three files are created then their names will be “st_EN85_1.xml”, “st_EN85_2.xml” and “st_EN85_3.xml”.

New Parameter ‘load_be’

If you want a query to be run at the end of the process the ‘load_be’, ‘be_query’ and ‘be_asqlflag’ parameters should be used.

The ‘load_be’ parameter must be entered exactly as shown, in particular the ‘Default value’ entry must be “InterfaceTools.clsGen;BeQuery” or “InterfaceTools.clsGen;RunQuery”

On report variants but not user defined reports the ‘#’ entry must be greater than or equal to “100”

New Parameter 'be_query'

This parameter should have its 'Default value' set to the name of the AG16 query that you want to be run at the end of the process.

Its '#' entry should be set to one more than that of the 'load_be' parameter.

New Parameter 'be_asqlflag'

If the 'be_query' AG16 query is written in ASQL set this parameter's 'Default value' to "1" (one), otherwise set it to "0" (zero)

Its '#' entry should be set to one more than that of the 'be_query' parameter.

New Parameter 'be_name_in'

If you wish to use either the 'be_asc_only' or 'be_xml_safe' parameters then you must give this parameter a name.

Set 'Default value' to 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 the commas are ignored
'Data' to 'a'
'Length' to 30
'#' to one more than the '#' value of the 'be_asqlflag' parameter
and 'Fixed' to checked

New Parameter 'be_asc_only'

Set 'Default value' as follows:

- If all the files in 'be_name_in' contain non-ASCII characters and you want them removed then set 'Default value' to "Y"
- If none of the files in 'be_name_in' contain non-ASCII characters or they do but you don't want them removed then set 'Default value' to "N". This is the default setting if the parameter is missing or left blank
- If you want to remove non-ASCII characters from only some of the files in 'be_name_in' then set 'Default value' to a comma separated list of Ys and Ns, , 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'

'Data' to 'A'
'Length' to 30
'#' to one more than the '#' value of the 'be_name_in' parameter
and 'Fixed' to checked

A non-ASCII character is one whose encoding value is greater than 127. The ASCII character set is primarily that which appears on an English or American keyboard (numbers, upper and lower case English letters and various symbols and punctuation marks) plus 33 control codes (mainly obsolete printer and modem control characters, but also CR, LF and TAB). There are literally thousands of non-ASCII characters

New Parameter 'be_xml_safe'

Set 'Default value' as follows:

- If all the files in 'be_name_in' contain XML and may contain "unsafe" characters then set 'Default value' to "Y"
- If none of the files in 'be_name_in' contain XML or they do but you know that there are cannot be any unsafe characters then set 'Default value' to "N". This is the default setting if the parameter is missing or left blank

- If some of the files in ‘be_name_in’ may contain “unsafe” characters then set ‘Default value’ to a comma separated list of Ys and Ns, , 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’

‘Data’ to ‘A’
‘Length’ to 30
‘#’ to one more than the ‘#’ value of the ‘be_name_in’ parameter or the ‘be_asc_only’ parameter if it is present
and ‘Fixed’ to checked

If any the following characters are contained in the payload (or value) of an XML element then this will cause programs to fail to read that XML. If any these “unsafe” characters are present they must be replaced by their “entity” symbol to avoid this problem

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

Setting ‘xml_safe’ to “Y” will automatically replace any unsafe characters by their entities

New Parameter ‘be_xml_split’

Set ‘Default value’ as follows:

- If all the files in ‘be_name_in’ contain XML and contain the same repeating element and you wish each occurrence of that repeating element to appear in a separate file (due to restrictions in the receiving system) then set ‘Default value’ to the name of the repeating element without the “<” or “>” characters or any of its attributes, e.g. “Supplier”. The value of this parameter is case sensitive
- 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 omit this parameter or set ‘Default value’ to blank.
- If some of the files in ‘be_name_in’ contain XML and you wish to split different repeating elements into different files or not split them at all then 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 “''”

‘Data’ to ‘a’
‘Length’ to 30
‘#’ to one more than the ‘#’ value of the corresponding ‘xml_safe_*n*’ parameter
and ‘Fixed’ to checked

If more than one file is created then each file will have a sequence number starting at one.

Example

If the parameter pointed to by ‘be_name_in’ contains “C:\temp\AbwOut\st_<c><o>.xml” and it is run in client “EN” and its order number is 85 and one file is created in the “C:\temp\AbwOut” folder then its name will be “st_EN85.xml”. If 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. In the screen shot one additional parameter is present ‘bi_qryparam1’

Run a Command-Line Tool

Introduction

While the other server exits provided by InterfaceTools can cater for many of the situations found while implementing a system. There will sometimes be requirements not provided as standard and it may be that you have access to a command-line tool that meets this requirement. This server exit allows you to run that command-line tool thus meeting your requirement.

An example of such a requirement might be to read data from an XML file:

- A free download 'Microsoft® LogParser' can be used to translate files from one format to another, XML to TSV being only one of many conversions it can carry out. Microsoft® LogParser and its documentation can be downloaded from http://www.microsoft.com/downloads/details.aspx?FamilyID=890cd06b-abf8-4c25-91b2-f8d975cf8c07&displaylang=en
- An AG16 query could then be written to read the resulting TSV file

Two server exits are provided:

Exit name	batch-initialise	batch-end
BiRun	✓	
BeRun		✓

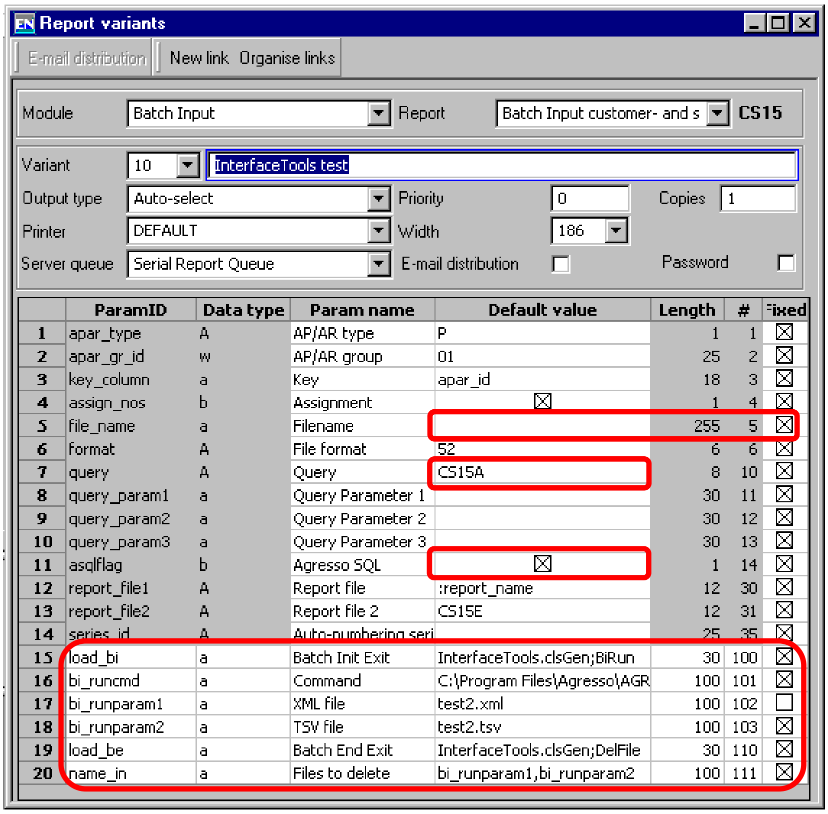
WARNINGS

These server exits do not check that the command-line tool has run without errors, so it is essential that you thoroughly test the tool to be run in an command-prompt window before linking it to the report variant

The use of Microsoft® LogParser in this section is purely illustrative and cannot be taken as an endorsement or recommendation of that software by Agresso Limited.

Set Up

In order to run a command-line tool you must set up a variant with some of the existing parameters changed and some new parameters added. The example is for CS15 but the same principle applies to any other data import routine



Module	Batch Input	Report	Batch Input customer- and s	CS15	
Variant	10	InterfaceTools test			
Output type	Auto-select	Priority	0	Copies	1
Printer	DEFAULT	Width	186		
Server queue	Serial Report Queue	E-mail distribution	□	Password	□

	ParamID	Data type	Param name	Default value	Length	#	Fixed
1	apar_type	A	AP/AR type	P	1	1	☒
2	apar_gr_id	w	AP/AR group	01	25	2	☒
3	key_column	a	Key	apar_id	18	3	☒
4	assign_nos	b	Assignment	☒	1	4	☒
5	file_name	a	Filename		255	5	☒
6	format	A	File format	52	6	6	☒
7	query	A	Query	CS15A	8	10	☒
8	query_param1	a	Query Parameter 1		30	11	☒
9	query_param2	a	Query Parameter 2		30	12	☒
10	query_param3	a	Query Parameter 3		30	13	☒
11	asqlflag	b	Agresso SQL	☒	1	14	☒
12	report_file1	A	Report file	:report_name	12	30	☒
13	report_file2	A	Report file 2	CS15E	12	31	☒
14	series_id	A	Auto-numbering seri		25	35	☒
15	load_bi	a	Batch Init Exit	InterfaceTools.clsGen;BiRun	30	100	☒
16	bi_runcmd	a	Command	C:\Program Files\Agresso\AGR	100	101	☒
17	bi_runparam1	a	XML file	test2.xml	100	102	□
18	bi_runparam2	a	TSV file	test2.tsv	100	103	☒
19	load_be	a	Batch End Exit	InterfaceTools.clsGen;DelFile	30	110	☒
20	name_in	a	Files to delete	bi_runparam1,bi_runparam2	100	111	☒

Parameter 'file_name'

Must be blank and fixed

Parameter 'query'

Must be set to the appropriate query's name

Parameter 'asqlflag'

Change the 'Default value' to 1 (pre 5.5) or check it (5.5 onwards) if your query is written in ASQL

New Parameter 'load_bi'

If you want a command-line tool to be run at the beginning of the process the 'load_bi', 'bi_runcmd' and 'bi_runparam1' to 'bi_runparam*n*' parameters should be used.

Set 'Default value' to "InterfaceTools.clsGen;BiRun"
'Data' to 'a'
'Length' to 30
'#' to On report variants but not user defined reports the '#' entry must be greater than or equal to "100"
and 'Fixed' to checked

New Parameter 'bi_runcmd'

The path of the command-line tool to be run. This can either be a absolute path (as shown) or can be relative to the Server (pre 5.5) or Bin (5.5 onwards) folder. The path can contain spaces.

New Parameter 'bi_runlog'

This parameter is not shown in the screenshot.

By default this server exit will "capture" the output of the command specified by the 'bi_runcmd' parameter and display it in the process' log file. If you do not require this then add this parameter to the variant and set its value to "N".

Valid values for this parameter are "Y", "N" or blank, blank being the same as "Y"

New Parameter 'bi_runparam1' to 'bi_runparam*n*'

Each of these parameters corresponds to a command line argument, 'bi_runparam1' is the first argument, 'bi_runparam2' is the second and so on. There is no fixed limit to the number of these arguments, the program will stop looking when it encounters a parameter that is missing or has a blank default value, e.g. 'bi_runparam3' in the screen shot. A parameter's value can contain spaces.

So if 'bi_runcmd' is set to "C:\my script.vbs" and 'bi_runparam1' is set to "A" and 'bi_runparam2' is set to "B D" the following command will be executed:

```
"C:\my script.vbs" "A" "B D"
```

New Parameter 'load_be'

If you want a command-line tool to be run at the end of the process the 'load_be', 'be_runcmd', 'be_runlog' and 'be_runparam1' to 'be_runparam*n*' parameters should be used.

Set 'Default value' to "InterfaceTools.clsGen;BeRun"
'Data' to 'a'
'Length' to 30
'#' to On report variants but not user defined reports the '#' entry must be greater than or equal to "100"
and 'Fixed' to checked

New Parameter 'be_runcmd'

The path of the command-line tool to be run. This can either be a absolute path or can be relative to the Server (pre 5.5) or Bin (5.5 onwards) folder. The path can contain spaces.

New Parameter 'be_runlog'

By default this server exit will "capture" the output of the command specified by the 'be_runcmd' parameter and display it in the process' log file. If you do not require this then add this parameter to the variant and set its value to "N".

Valid values for this parameter are "Y", "N" or blank, blank being the same as "Y"

New Parameter 'be_runparam1' to 'be_runparam*n*'

Each of these parameters corresponds to a command line argument, 'be_runparam1' is the first argument, 'be_runparam2' is the second and so on. There is no fixed limit to the number of these arguments, the program will stop looking when it encounters a parameter that is missing or has a blank default value. A parameter's value can contain spaces.

So if 'be_runcmd' is set to "C:\my program.exe" and 'be_runparam1' is set to "1", 'be_runparam2' is set to "2 4" and 'be_runparam3' is set to "567" the following command will be executed:

```
"C:\my program.exe" "1" "2 4" "567"
```

Note

In the screen shot above the ‘load_be’ parameter is being used to delete the XML and TSV files at the end of the run to make it impossible to process the same file twice, see ’Delete or Move Input Files’ on page 24 for more details

Examples

Reading an XML File

This example uses Microsoft® LogParser (see Introduction) to convert a non-standard CS15 XML file into a TSV file so that a query can read the data into CS15’s import table (acsheaderinput). The screenshot above relates to this example

The XML File

```
<?xml version="1.0" encoding="utf-8"?>
<ROOT>
  <ROW>
    <apar_id>1</apar_id>
    <apar_name>My first supplier</apar_name>
    <client>D3</client>
    <cred_limit>1000</cred_limit>
    <ext_apar_ref>MFS1</ext_apar_ref>
  </ROW>
  <ROW>
    <apar_id>2</apar_id>
    <apar_name>My second supplier</apar_name>
    <client>D3</client>
    <cred_limit>1500</cred_limit>
    <ext_apar_ref>MFS2</ext_apar_ref>
  </ROW>
</ROOT>
```

The Command File

```
"C:\Program Files\Log Parser 2.2\LogParser.exe" -i:XML -fMode:Tree
        -o:TSV -headers:OFF
        "SELECT client, apar_id, apar_name, cred_limit
        INTO 'C:\Program Files\Agresso\AGRESSO 5.5\Data Files\agresso\
                Data Import\%~2'
        FROM 'C:\Program Files\Agresso\AGRESSO 5.5\Data Files\agresso\
                Data Import\%~1'"
```

Microsoft® LogParser is installed in its default folder: ”C:\Program Files\Log Parser 2.2”

The paths in the ‘INTO’ and ‘FROM’ clauses may need adjusting to your system

There is only one line in the real file; this line is very long and has been split into seven indented lines above in an attempt to improve readability

This Microsoft® LogParser command converts the XML file whose name is in command line argument one into a two line TSV file whose name is in command line argument two each line contains five fields which contain the contents of the client, apar_id, apar_name, cred_limit and ext_apar_ref nodes respectively.

The SQL Query

The query reads the TSV file created by LogParser into a temporary table, supplies missing mandatory values (one of which is the supplier group from the ’apar_gr_id’ process parameter) it then picks up the real values for these values for existing customers and finally inserts the data into acsheaderinput for processing by CS15

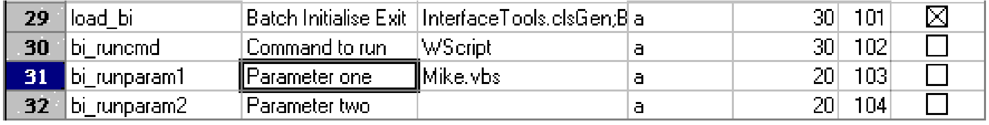
SeqNo	C	E	Description
10	C	S	Create temp table

<table>
<tr><th>SeqNo</th><th>C</th><th>E</th><th>Description</th></tr>
<tr><td colspan="4">CREATE TABLE $*tab1 AS SELECT address, client, apar_gr_id, apar_id, apar_name, apar_type, change_status, country_code, credit_limit, ext_apar_ref, full_record, pay_method, short_name, status FROM acsheaderinput WHERE 1 = 2</td></tr>
<tr><td>20</td><td>N</td><td>S</td><td>Copy from file</td></tr>
<tr><td colspan="4">COPY IN IMPORT FILE = '$bi_runparam2', COLSEP = T, TABLE = $*tab1, client, apar_id, apar_name, credit_limit, ext_apar_ref</td></tr>
<tr><td>30</td><td>N</td><td>S</td><td>Set missing values</td></tr>
<tr><td colspan="4">UPDATE $*tab1 h SET h.address = 'TBA', h.apar_gr_id = '$apar_gr_id', h.apar_type = 'P', h.change_status = 'I', h.country_code = 'GB', h.full_record = 1, h.pay_method = 'CH', h.short_name = UPPER(LEFT(apar_name, 10)), h.status = 'N'</td></tr>
<tr><td>50</td><td>N</td><td>S</td><td>Flag existing suppliers</td></tr>
<tr><td colspan="4">UPDATE $*tab1 h FROM asuheader s SET h.apar_gr_id = s.apar_gr_id, h.change_status = 'U', h.country_code = s.country_code, h.pay_method = s.pay_method, h.short_name = s.short_name, h.status = s.status WHERE s.client = h.client AND s.apar_id = h.apar_id</td></tr>
<tr><td>60</td><td>N</td><td>S</td><td>Get existing supplier address</td></tr>
<tr><td colspan="4">UPDATE $*tab1 h FROM agladdress a SET h.address = a.address WHERE h.change_status = 'U' AND a.client = h.client AND a.attribute_id = 'A5' AND a.dim_value = h.apar_id AND a.address_type = '1' AND a.sequence_no = 0</td></tr>
<tr><td>70</td><td>C</td><td>S</td><td>Transfer</td></tr>
<tr><td colspan="4">INSERT INTO acsheaderinput (address, client, apar_gr_id, apar_id, apar_name, apar_type, change_status, country_code, credit_limit, ext_apar_ref, full_record, pay_method, short_name, status) SELECT address, client, apar_gr_id, apar_id, apar_name, apar_type, change_status, country_code, credit_limit, ext_apar_ref, full_record, pay_method, short_name, status FROM $*tab1</td></tr>
</table>

Running a Windows Scripting Program

If you cannot find a third-party utility to do what you want then one solution would be to write a windows scripting program that does it. This example shows VB Script being used but the principle applies equally to J Script.

The tricky bit here (assuming that you know how to write VBS or JS is how to define the process parameters:



29	load_bi	Batch Initialise Exit	InterfaceTools.clsGen;B	a	30	101	☒
30	bi_runcmd	Command to run	WScript	a	30	102	☐
31	bi_runparam1	Parameter one	Mike.vbs	a	20	103	☐
32	bi_runparam2	Parameter two		a	20	104	☐

Note that:

- The value of **bi_runcmd** is the 'Windows Script Host' program; this is located in \WinNT\system32 which is normally in the 'PATH' environment variable and therefore doesn't need a fully qualified file path
- The value of **bi_runparam1** is the name of the script and MUST include the ".vbs" or ".js" suffix
- The values of **bi_runparam2-*n*** are parameters 1 to *n*-1 of the script

.

Access Process Parameters on 5.5

Introduction

In some versions of ABW 5.5 a batch-end server exit has no access to the process' parameters if the server process has produced no printout. If the process you are adding one of the Toolkit's batch-end routines to suffers from this problem you will know because you will get the following error in the log file:

```
Error -2147417848: Method '~' of object '~' failed
```

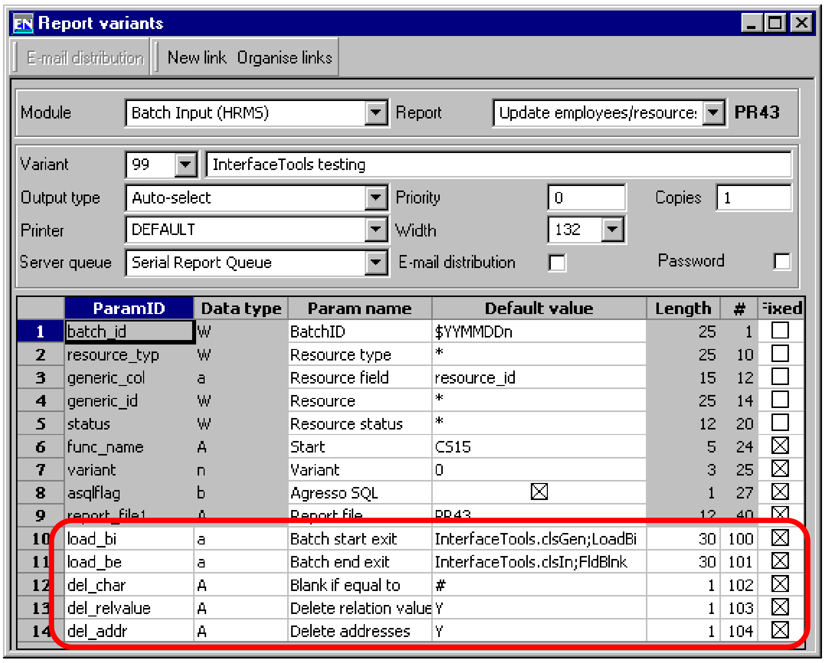
If this happens then you will need to use this batch-initialise server exit on the process.

If you already have one of the toolkit's batch-initialise server exits on the process these routines call the new routine automatically, i.e.:

- clsGen.DelFile
- clsGen.BiQuery
- clsGen.RunQuery
- clsIn.ConvCfr
- clsIn.ConvCsv
- clsIn.ConvFsr

Set Up

In order to use this server exit on a process you must set up a variant with an additional parameter. The example is for 'AGRESSO Human Resources ▶ Batch input' process "Update employees/resources" (PR43) but the same principle applies to any other server process. As this server exit is only needed on ABW 5.5 systems the screen-shot is from 5.5



Module: Batch Input (HRMS)
Report: Update employees/resource: PR43
Variant: 99 InterfaceTools testing
Output type: Auto-select
Priority: 0
Copies: 1
Printer: DEFAULT
Width: 132
Server queue: Serial Report Queue
E-mail distribution: □
Password: □

	ParamID	Data type	Param name	Default value	Length	#	Fixed
1	batch_id	W	BatchID	$YYMMDDn	25	1	□
2	resource_typ	W	Resource type	*	25	10	□
3	generic_col	a	Resource field	resource_id	15	12	□
4	generic_id	W	Resource	*	25	14	□
5	status	W	Resource status	*	12	20	□
6	func_name	A	Start	CS15	5	24	☒
7	variant	n	Variant	0	3	25	☒
8	asqlflag	b	Agresso SQL	☒	1	27	☒
9	report_file1	A	Report file	PR43	12	40	☒
10	load_bi	a	Batch start exit	InterfaceTools.clsGen;LoadBi	30	100	☒
11	load_be	a	Batch end exit	InterfaceTools.clsIn;FldBlnk	30	101	☒
12	del_char	A	Blank if equal to	#	1	102	☒
13	del_relvalue	A	Delete relation value	Y	1	103	☒
14	del_addr	A	Delete addresses	Y	1	104	☒

New Parameter 'load_bi'

The 'load_bi' parameter must be entered exactly as shown, in particular the 'Default value' entry must be "InterfaceTools.clsGen;LoadBi"

On report variants but not user defined reports the ‘#’ entry must be greater than or equal to “100”

No other parameters are required

Note

The screen-shot is from my 5.5 test system and shows the standard “FldBlnk exit being used with PR43 rather than “FldBlnkPR43”: see ‘Blank Columns and Remove Addresses and Relation Values’ on page 26 for more details