

Aggregated Data Import Documentation

13/05/08

Contents

Introduction.....	2
1 Conversion.....	2
1.1 Installing the converting program.....	2
1.1.1 Compiling the conversion-tool.....	2
1.2 Creating the conversion project.....	3
1.2.1 Preparing initial file.....	4
1.2.2 Creating the parameter files.....	5
1.2.2.1 Control-Gendat.....	5
1.2.2.2 Control-Input.....	6
Reading block.....	7
Dummy block.....	9
Aggregations block.....	9
Summations block.....	10
Divisions block.....	11
Median block.....	11
1.2.2.3 Control-Output.....	12
1.2.2.4 Assignments file.....	13
1.2.2.5 Sum column specification file.....	14
1.2.3 Creating thcodnam.....	15
1.2.3.1 Installing program for creation thcodnam.....	16
1.2.3.2 Starting thcodnam creation tool.....	16
1.3 Performing the conversion.....	17
1.4 Conversion result.....	18
1.4.1 cluster.dat.x.....	18
1.4.2 cluster.son.x File.....	20
1.5 Analyzing conversion log.....	20
2 Data base import.....	22
2.1 Installing the import program.....	22
2.2 Creating the import project.....	22
2.3 Performing the data import.....	23
2.4 Evaluating the import log.....	24
2.5 Deleting the data stock.....	24
3 Adding the metadata in the data base.....	25
3.1 Adding the notes to the stock.....	25
3.1.1 Installing the notes administration tool.....	25
3.1.2 Starting the notes administration tool.....	25
3.2 Updating the virtual stocks.....	26
3.2.1 Virtual vs. physical stocks.....	26
3.2.2 Original files with virtual stocks.....	26
3.2.3 cluster.dat.x.HC File.....	27
3.2.4 Format of the cluster.dat.x.HC file.....	28
3.2.5 Format of the cluster.son.x.HC File.....	29
3.2.3 Reading the virtual stocks.....	29

Introduction

The import of aggregated data is performed in two steps:

* Within the first step the original data which are available as text files with fixed field lengths are read and coded using ICE-keys. In doing so the data are transformed if necessary.

Transformations may include transcoding of a field, generating a new field (e.g. of the average age as calculation result of the divisions of other fields), or generating new field values (new rows for subtotals).

* The transformed data are written into several text files with fixed field lengths which are imported into the ICE-data base within the second step.

Unlike individual data each row does not represent an individual case, but similar individual cases are grouped and provided with their quantity.

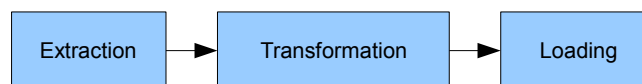


Figure 1: Data import flow

1 Conversion

1.1 Installing the converting program

For converting the data the Fortran program `gendat` is used. It runs under Unix/Linux and does not require installing. The program may be copied into any directory within the file system.

`gendat` was compiled for 32 bit 686 CPUs supporting SSE instructions and linked statically. If you experience problems because your environment does not meet this scope (i.e. on a 64 bit system or you get segmentation faults), you should consider recompiling the program by yourself (→ 1.1.1).

The program works with ASCII files with UNIX-linefeed and does not require a connection to databases.

1.1.1 Compiling the conversion-tool

The package names given here are taken from *Debian GNU/Linux*, but should apply similar to other distributions.

For a successful build of `gendat` you basically need a working Fortran (`f77`) and C (`cc`) compiler environment. Thus, you have to install the packages '`g77`' and '`gcc`' with all required dependencies. Other compilers may work as well, but you have to edit the Makefile '`gendat.make`' respectively. The compiler flags have to be modified to meet

your systems settings. If in doubt, omit the parameters `-march=` and `-m` completely.

Furthermore you need to have the package 'recode' installed.

The compilation process is started with either `make static` or just `make`. The latter will link dynamically against the Fortran runtime-library 'libg2c0', which then is required to be installed on the destination system. It is rather recommend to compile from the sources on the target system itself to avoid problems and therefore it is most likely for that library to already exist.

In case no error occurs while within the compile-process, you find the program `gendat` ready to use in the source directory.

1.2 Creating the conversion project

A conversion project includes all input and output files as well as the parameter files controlling the transformation, a fixed directory structure being predetermined:

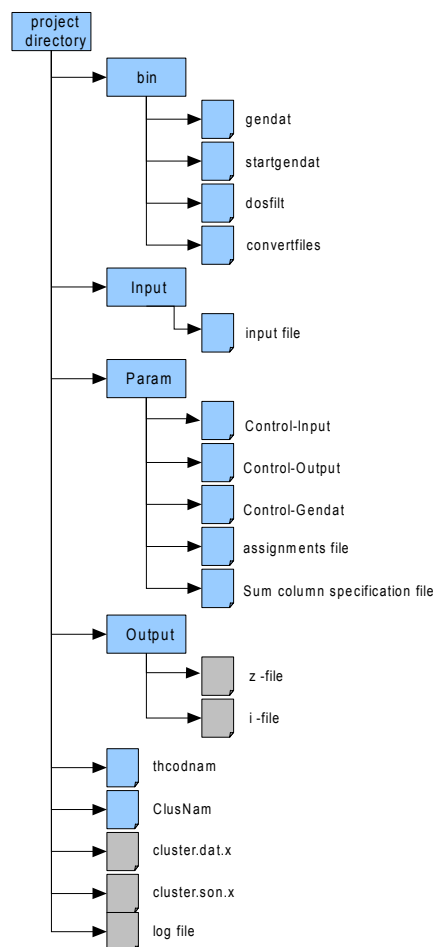


Figure 2: Layout of the project directory for the conversion

The project directory may be given any name. It must contain four subdirectories:

- bin/ contains the program `gendat` as well as some scripts required for the execution of the program.
- Input/ contains the original file.
- Param/ contains the parameter files for the conversion.

- Control-Gendat describes the files required for the conversion run.
- Control-Input describes the input file and controls the reading thereof,
- Control-Output describes the resulting file,
- assignments file contains the assignment of the external keys to the ICE-keys,
- the sum column specification file describes the layout of sum columns,
- In Output/ the resulting files of the converting step are filed:
 - z-file with the sums being the result of the conversion,
 - i-file with the encryption of each sum row.
- Directly within the project directory there are
 - the thcodnam with the ICE-keys,
 - ClusNam for managing the cluster.son-entries,
 - cluster.dat and cluster.son files. In case they are not yet available, they will be created during conversion and describe the applicable ICE-keys for each converted original file.
 - the logs of the conversion runs.

1.2.1 Preparing initial file

Currently text files with fixed field lengths per input field are supported. Each figure within the original file shall be encrypted such that, using the key description, every sum in the file may be unambiguously identified.

Frequently the data are available unencryptedly, e.g. as Excel spreadsheet. Then the keys must be assigned manually on the basis of the describing text. Within Excel the VisualBasic macros may be used for this purpose.

In certain cases the describing keys do not have to appear in the file. However, a clear pattern must be available then to describe the figures in the file.

Example:

The input file includes three sum fields for female, male and all students in Germany. In each row figures for a particular land and a particular type of institution appear. The land and type of institution keys do not have to show up in the initial file itself if a definite order exists, for example, if the rows are ordered such that each particular type of institution is distinguished according to the federal states (the order of the types of institution being known). With this, the order and the number of the federal states shall be the same for each type of institution.

				492	1	1	2	2	3	3	4	4	5	5
				493	2	1	2	1	2	1	2	1	2	1
605	102	490	491											
20050	2	1	1	-	14	-	11	-	57	-	34	-	32	
20050	2	1	1	-	1	-	4	-	13	7	10	-	-	
20050	2	1	1	-	-	-	1	-	9	-	10	-	2	
20050	2	1	5	-	25	-	5	-	47	-	45	-	2	
20050	2	1	2	-	12	-	2	-	52	-	21	-	-	
20050	2	1	3	-	-	-	1	-	27	-	18	-	-	
20050	2	1	3	-	-	-	-	-	1	-	11	-	-	
20050	2	1	0	-	52	-	24	-	206	7	149	-	36	
20050	2	2	1	-	22	-	12	-	93	68	52	-	-	
20050	2	2	6	-	22	-	1	-	55	-	26	3	31	
20050	2	2	5	-	6	-	-	-	29	28	27	-	-	
20050	2	2	6	-	5	-	-	-	16	4	19	-	-	
20050	2	2	4	-	8	-	1	-	42	-	29	74	30	
20050	2	2	5	-	13	-	4	-	40	42	35	1	-	
20050	2	2	2	-	25	-	3	-	39	107	22	2	-	
20050	2	2	0	-	101	-	21	-	314	249	210	80	61	
20050	2	3	1	-	11	-	6	-	51	3	30	9	8	
20050	2	3	2	-	13	-	2	-	45	1	14	1	-	
20050	2	3	3	-	2	-	1	-	73	4	55	17	-	
20050	2	3	5	-	14	-	2	-	32	2	49	-	-	
20050	2	3	0	-	40	-	11	-	201	10	148	27	8	

Figure 3: Coded initial file

1.2.2 Creating the parameter files

The conversion is organised by several control files:

- The Control-Gendat describes all the files required for the conversion process.
- The Control-Input file describes the input file and controls the reading thereof.
- The Control-Output file describes the resulting file.
- Assignment file specifies ICE-keys for the available foreign keys.
- Sum column specification file describes the complex coding method of the value columns of the initial file.

1.2.2.1 Control-Gendat

The general control of the converting program is performed via this file. It contains the other file names to be used for the conversion. The file information given in the file may contain absolute and relative paths (relative to the project directory).

Rows starting with a blank will not be taken into account and may be used for comments. Rows starting with a semicolon ';' will be treated as comment rows as well.

By definition of the files to be used the assignment of the precise file name to the Fortran channel number is realised:

Channel number	Description
8	File with comment/message output
11	Intermediate file of the program. By default the file 'raus' in the project directory is used.

Channel number	Description
14	ClusNam file
20	z-file (maximum length of the file name is 20 characters)*
21	i-file (maximum length of the file name is 20 characters)*
22	Control-Output file
23	s-file*
26	This channel is for the Control-Gendat file itself and for this reason may not be used on other respects.
30	Original file (= input file)
40	File with the assignments of the foreign keys to the ICE-keys
41	Control-Input file
46	Sum column specification file

* The first parts of the names of z-, i- and s-files should be equal.

Row format:

- 1-digit: F
- 1-digit: Blank
- 2-digit: Channel number
- 7-digit: Blank
- Rest: File name (absolut or relative to the project directory)

```

F 30      Input/staff_2005.prn
F 22      Param/staff_2005      .cout
F 41      Param/staff_2005      .cinp
F 20  44  Output/stud_00_04_05.z
F 21      Output/stud_00_04_05.i
F 14  72  ClusNam
          F 40      Param/staff_2005.assign
          F 46      Param/staff_2005.colspec

```

Figure 4: Control-Gendat-File example

1.2.2.2 Control-Input

The Control-Input file is used for controlling the file reading. The file name actually used is specified in the Control-Gendat file by assignment to channel 41.

Rows beginning with a semicolon ';' are regarded as comment rows. In case of an error, however, they will be considered for the row number indication.

The file defines which information is read from the input file and which transformations shall be performed with the data. It consists of several blocks being separated by the row containing „99“ at its beginning:

- Within the first block the reading formats for the Input file including die assignment to the ICE-keys of the data to be read are defined.
- Within the second block the summation dummies are defined.
- Within the third block the aggregations are defined.
- Within the fourth block the summations are defined.
- Within the fifth block the divisions are defined.
- Within the sixth block the median, the lower and upper quartile and the weighted average are defined.

Reading block

In the *first row* the format for reading the sum columns from the input file appears. 1000 characters are readable from one row of the data table. Maximum 100 sum columns can be read. For integer values a format string of the I format, as is common in Fortran, is used (e.g. (20X, 6I12)). For floating point numbers with or without decimal point a Fortran real format (F format) is used, the format string being prepended by a * and a multiplier, by which read values are multiplied before they are rounded to the next integer value (e.g. 100*(20X, 6F12.2)), since the Gendat program internally works with integer numbers. If no multiplier is given (e.g. (20X, 6F12.2)), an internal multiplier is determined automatically in order no loss of digits to occur. A matching decimal places number is entered. If less decimal places are given in the format than are actually marked by the decimal point in the data, the additional digits will be dropped with the internal rounding to integer values.

Example: Format (F9.1), applied to the value 8765.4321, then gives 87654 with the information that the data stock has one decimal place. With format (F9.4) 875.4321 and 87654321 will become 87654321 with the information entered that the data stock has 4 decimal places.

In the *second row* the format for reading the attributes for the variables appears, with which the values in the input file are coded. A total of 1000 characters are readable from one row of the data table. All attributes have to be read in the 'A' format.

Between the second and third rows further rows may appear, which are *optional*:

- DECIMAL_PLACES <number>

In this row the decimal places number is given, which shall be carried over into the description of the data table in the cluster.dat file. Values between 0 and 9 are allowed. This specification overrides any other determination of the decimal places number.

- BLANK <0 or - respectively>

Besides the figures the sum columns in the input file may contain the blanks as well. By default the blanks are evaluated as '0' ('BLANK 0'). If it is required that the blanks should be interpreted as missing value and thus do not appear in the output file, '-' must be given in the row ('BLANK -').

- SPECIAL_CHARACTERS <character in input file> <character in z-file> <number at calculation>

e.g. 'SPECIAL_CHARACTERS XXX uu 0'

Besides the figures the sum columns in the input file may contain the special characters as well (e.g. 'k.A.', 'x'). At this place it is defined what to write into the

output file (e.g. uu) instead of the special character string (e.g. XXX) and which numerical value to use with the calculations (e.g. 0).

Up to 10 rows with special character definitions may be given.

Not all calculations can be done with the data tables containing special characters in sum columns. Adding-up instead of overwriting the data values when reading, which may be switched on by a '+' in the third row of the Control-Input file, does not yet work with special values.

- MEDIAN <1 or 2 respectively>

Herewith the type of the median or quartile calculation is specified, respectively. 'MEDIAN 1' means that median and quartile are calculated in a „conventional“ manner. 'MEDIAN 2' means that median and quartile are calculated in a way deviating from the conventional method and used by the Statistisches Bundesamt (Federal Statistical Office).

In the `third row` the number of variables being read or converted appears, respectively (given with 4 digits). The number is limited to 20 variables. As fifth digit in the row a '+' may appear if, when reading, adding-up of the sum values with the same key assignment is desired. In normal cases such values will be overwritten. Furthermore, the number of the header rows which will not be considered when reading the input file may be given at the digits 6 to 10.

From the `fourth row` onwards information regarding the keys is given with which the values in the input file are coded. For this, one goes through the columns, where the coding appears (see reading format in the 2nd row), in the order from left to right. The rows describing the keys look as follows:

- 2-digit: Number of the variable being involved in the coding. Starting with 1 it is incremented always by 1. If multiple rows are needed for the key information of one variable, they appear under the same variable number.
- 5-digit: The ICE-characteristic-number assigned to the variable. The number shall be available in the 'thcodnam' file. If the variable is used for the description of the value columns (so-called column characteristic), the ICE-characteristic-number is prepended by a '-'.
5-digit: Here it is defined how to handle the variable when reading. The following types of the variable are defined:

1 – The variable is not read from the input file. It may happen that the original file does not contain certain variables which, however, are needed for coding the figures. or it may be the case that the coding of the input file is missing completely, but the data rows have a 'symmetrical' structure. While the variable will not be read in such cases (and does not appear in the reading format in the 2nd row), it still will be enumerated and quoted with the variable type '1'. For the files with the 'symmetrical' structures the order of the variables and the attribute pairs are of importance. The order of the characteristics is from the outside inwards: If, for example, the rows are arranged by type of institution and below by länder, the information regarding the type of institution has to be given first, and those regarding the land next. The order of the attribute pairs is geared to their appearance in the input file as well. The same principle applies to the columns.

2 – The variable contains foreign keys. It is read from the input file, the foreign keys being converted to ICE-keys using the assignments file.

3 – The variable is read from the input file one-by-one.

4 – The attribute pairs following this are not read from the input file, but will be calculated by summations and divisions, respectively. For this, the attributes with the type of variable 1, 2 or 3, respectively, have to be given for the same variable beforehand. If only calculated attributes shall be output, an attribute not existing in the key has to be given with the type of variable 6? for the variable. This will be employed with the calculation of quotas or average values using the division.

6 – The variables are read from the sum column specification file. The type of variable is used with column variables only.

- 5x(5-digit + 5-digit): Hereafter all attributes are given which shall be read from the input file. The enumeration is done by giving the 'from' and the 'to' attribute. If there are gaps (attributes being present in the key, but missing in the input file), multiple pairs have to be given. Up to 5 attribute pairs may be entered per row. A total of up to 2000 attributes per variable may be used.

Dummy block

The keys being enumerated in the reading block will also appear in the output file (including the matching data). Sometimes it happens that certain attributes shall not be output, but, since they are used for the data conversion, they are still of relevance.

For each dummy the following row is defined:

- 2-digit: the consecutive number of the variable from the reading block,
- 5-digit: the ICE-characteristic-number for the variable from the reading block,
- 8-digit: the attribute to read with an offset of 100000, e.g. 100055, if attribute 55 shall be read in the input file.

Aggregations block

For each aggregation the ICE-characteristic, followed by the aggregated attributes, are given in the row. After conversion the data are filed in the output file under the first attribute. Under the further attributes given in the aggregation no data are output.

The number of digits occupied by the particular attribute in the aggregation is defined by the correct entry in the Control-Output file. Thus, with a 1-digit attribute 1234 would be separated into 1, 2, 3, and 4, with a 2-digit one into 12 and 34.

The aggregations can only be gathered with the type of variable 3 and 6, respectively. The attributes used for the aggregation also have to be given in the reading block. Otherwise they will be entered as missing.

Example: For the variable 309 the attributes 02 and 03 are gathered. Then the attribute 203 or 0203 appears in the input file, respectively. The row in the control-read in aggregations-block is '309 0203'. In the resulting file the values are coded with the attribute 02, therefore the cluster.son file gets the entry '309 2 3' (without a minus after the characteristic number).

Summations block

In the summations block sums and differences may be formed. Here for each ICE-characteristic-attribute being gathered with type of variable 4 it is given, in which way it can be formed from other attributes of the same characteristic or using the dummies, respectively. Each summation is built as follows:

- 3-digit: the ICE-characteristic-number for the variable from the reading block,
- 2-digit: blanks,
- 5-digit: the attribute to be determined by the summation,
- 1-digit: blank or continuation character. If the formula has more than 10 summands, it will be split into multiple rows. The continuation character '#' marks the rows of a summation having following rows.
- 1-digit: 's' for sums being formed even if not all the summands are present in the original file; 'v' for sums being written into the output file only if all the summands involved exist,
- up to 10 summands per row built as follows:
 - 1-digit: blank,
 - 1-digit: plus or minus character, if the attribute shall be added or subtracted, respectively. The attribute '0' may not be subtracted.
 - 6-digit: the ICE-attribute or dummy, respectively.

It is possible to define the summation without to count all summands. To do this, you need to enter the first and the last attribute for the characteristic with '_' in the column 22. For example:

```
303  0 s +  1 _ 548
```

The summations given as described above are calculated and written into the output file. If it is desired not to output the sums but to let them being calculated by the system at run time, it is possible to cause that merely the attributes to be calculated and the calculation rules are written into the output file. For the output an s-file (channel 23) has to be given in the Control-Gendat file. The sums which must be calculated at run time are identified by an 's' at the 4th digit in the summation (i.e. directly following the 3-digit ICE-characteristic-number). Or it may be as well sufficient to enter merely the ICE-characteristic-number (3-digit), the 's', a blank and the attribute to be determined (5-digit) instead of the full summations row.

Besides summations at run time also those ICE-keys may be calculated, which consist of attributes of another characteristic, the so called aggregations. For example subject groups from study areas may be calculated. The configuration of the aggregation is similar to the information for summation: the ICE-characteristic-number to be built (3-digit), the 'a', a blank and the ICE-characteristic-number for the output variable (5-digit).

Divisions block

The results of the divisions will be rounded to nearest. If always the same integer power of ten is taken as multiplier (e.g. 10), the corresponding number of decimal places (1 for the example) will be entered in the cluster.dat file.

The divisions are built as follows:

- 3-digit: the ICE-characteristic-number for the variable from the reading block,

- 1-digit: blank,
- 6-digit: the attribute to be determined by the division,
- 1-digit: blank,
- 1-digit: 'd' for division,
- 1-digit: blank,
- 1-digit: '*',
- 6-digit: multiplier being multiplied with the numerator previous to the division and not being involved in determining the decimal places for the output file,
- 1-digit: blank,
- 1-digit: '*',
- 6-digit: multiplier being multiplied with the numerator previous to the division. This multiplier specifies the number of the decimal places for the output file, if:
 - the multiplier is a power of ten within the range 1 10 100 ... 1E8 1E9 (0, 1, 2... 8, 9 decimal places correspondingly),
 - no other multiplier is used in one of the other divisions,
 - no number of decimal places is given in the reading block, neither explicitly nor by the format string.

'0' decimal places will be entered otherwise.
- 2-digit: blank,
- 6-digit: the attribute of the numerator of the division,
- 1-digit: blank,
- 1-digit: division character '/',
- 6-digit: the attribute of the denominator of the division.

Example:

106 14 d * 10 100001 /100002

521 0 d * 10 * 10 100001 /100002

In the second example the 2nd '10' represents the number (= 1) of the decimal places in the output.

Median block

Here the following statistical measures may be determined:

- average,
- lower quartile,
- median,
- upper quartile.

...

```

;Reading-Block
;Reading format for value columns
(48X,12I12)
;Reading format for variables columns
(4A12)
;Number of decimal places
NACHKOMMASTELLEN 0
;SONDERZEICHEN - -
;Number of variables for coding,
;Number of rows in the beginning that should not be read
6+ 3
;Variables in detail
1 605 32005020050
2 102 3 2 2
3 490 3 1 1 2 2 3 3 4 4
3 490 3 5 5 6 6 7 7 8 8
3 490 3 9 9 10 10 11 11 12 12
3 490 3 14 14 13 13 0 0
4 491 3 1 1 5 5 2 2 3 3
4 491 3 0 0 6 6 4 4
5 -492 1 1 1 2 2 3 3 4 4
5 -492 1 5 5 0 0
6 -493 1 2 2 1 1
6 -493 4 0 0
99
;Dummy-Block
99
;Aggregation-Block
99
;Summations-Block
493 0 s + 1 + 2
99
;Divisions-Block
99

```

Figure 5: Control-Input-File example

1.2.2.3 Control-Output

The output is controlled by the Control-Output file. The file name actually used is specified in the Control-Gendat file by assignment to channel 22.

The file may not contain any blank or comment row. The configuration of the file is specified precisely.

In the first row appear:

- the number of integer formats in the format information for the output of the attributes (2-digit),
- blank,
- the number of Characters for the attribute output (2-digit),
- blank,
- the format information for the attribute output (20? characters maximum).

The further rows have the same configuration schemes:

- the number of characters which the output of the attribute occupies (1-digit),
- the ICE-characteristic-number (5-digit). A negative sign indicates that the characteristic stands for the description of the sum column (is a so called column characteristic).

- the 'from' attribute number (5-digit),
- the 'to' attribute number (5-digit).

In the 2nd to 4th rows information appears describing the data source (2nd row), the type of presentation (3rd row) and the data quality (4th row). The information regarding the 'from' attribute shall match the 'to' attribute for these three characteristics (so-called managing characteristics), since the information applies to all figures in the output stock and may be defined with one characteristic attribute only.

The fifth row describes the topic area for the output file. For this the characteristics from 101 to 109 are used. The topic characteristic determines which cluster file is used for the output, and shall be entered in ClusNam.

In the last row of the Control-Output the column characteristic is defined. There may be only one column characteristic.

The order of the characteristics in the output file is arbitrary from the sixth row onwards.

```
05 11 (i2,i5,i2,i1,i1)
2007180000100001
2002010000100001
2007010000200002
2001020000200002
5006052005020050
2004900000000014
1004910000000006
1004920000000005
1 -49300000000002
```

Figure 6: Control-Output-File example

1.2.2.4 Assignments file

The input files may contain coding working with other attribute numbers than the ICE-key. Although the numbers are read, they must be replaced by the ICE-keys during the conversion. The converting table is available then as assignments file, and is specified in the Control-Gendat file by the assignment to channel 40.

Rows beginning with a semicolon ';' are not taken into account by the program and may be used as comment rows.

In the first row the reading format for the subsequent rows of the file appears. The information is in Fortran format.

Example 1: The format (i5, i7, i6) specifies that the assignments appearing in the file shall be read as integer values. For each row, the first 5 characters are read first, then the next 7 characters and finally the following 6 characters.

Example 2: If the external attributes contain alphabetic characters, then they should be read by an ASCII format string (i5, A7, A6).

All other rows contain the assignment of the foreign attribute to the ICE-attribute for each ICE-characteristic concerned. The specifications appear in the order: ICE-characteristic, ICE-attribute, foreign attribute.

All rows of the assignments file (except the first format row) must be compatible to the

reading format. The row being read with '-1' or '99' as ICE-characteristic-information serves as end of file marker. The subsequent rows will not be read.

```

; Reading format for the file
(7x,i7,i10,x,i10)
; Charact ICE_Attrib external_key
STB 502 1 0
STB 502 2 1
STB 305 11 05
STB 305 15 10
STB 305 21 15
STB 305 22 25
STB 305 23 35
STB 305 24 40
STB 305 25 45
STB 305 26 50
STB 305 18 55
STB 305 30 60
STB 305 40 65
STB 403 11 1
STB 403 12 2
STB 403 13 3
STB 403 14 4
STB 403 15 5
STB 403 16 6
STB 403 17 7
STB 403 18 8
STB 403 19 9
STB 403 26 16
-1

```

Figure 7: Assignments file example

1.2.2.5 Sum column specification file

If the coding of the sum columns in the input file turns out to be complicated, the description of the coding of the sum columns may appear in the separate file. This file is specified by assignment to channel 46 in Control-Gendat. With a complex configuration the sum columns may be described merely with multiple variables, the attributes of which appear irregularly.

	A	B	C	D	E	F	G	H	I
1		492	1	2	3	4	5	0	0
2		501	0	0	0	0	0	1	2
3									
4		University/Faculty	Professor	Associate Professor	Senior Lecturer	Lecturer/ Prob. Lecturer	Academic Support Staff	Total	
5								male	female
6									
7									
8									
9	COLUMBO	Arts							
10		Education							
11		Law							
12		Medicine							
13		Science							
14		Management Studies							
15		Sri Palee Campus							

Figure 8: Example of initial file with complex column coding

Rows beginning with a semicolon ';' are not taken into account by the program and may be used as comment rows.

In all other rows the variable appears first (in the 7-digit field), followed by up to 10 attributes (in the 7-digit field as well, respectively). If the original file has more than 10 sum columns, further rows for the same variable have to be added. Thus an input file with up to 100 sum columns may be described.

```

; Reading format for this file (i7,7i7)
492      1      2      3      4      5      0      0
501      0      0      0      0      0      1      2

```

Figure 9: Sum column specification file example

1.2.3 Creating thcodnam

In the `thcodnam` file all the ICE-keys are present. The file is created by a Java program from the database. In this text file the characteristic number, attribute number, and key label appear in the particular format (the label only serving to improve the readability of the file). If for the conversion process a key is used which does not show up in `thcodnam`, the conversion process will be cancelled.

```

...
10100001 Persons qualified to enter higher education
10100002 Undergraduate admitted
10100003 New entrants (1st subject-related semester)
10100004 Enroled students
10100005 Examinations
10100006 Graduate output
10100007 Persons doing their doctorate
10100008 Persons doing their postdoctoral lecture qualification
10100009 New entrants (1st subject-related semester) relating to graduates
10100011 Changer of degree programmes
10100012 Changer of academic subjects
10100013 Changer of institutes of higher education
10100014 Student dropouts
10100016 Synopses/mixed groups
10100021 Unemployed graduates
10100022 German students abroad
10100023 German graduates with study experience abroad
10100024 Students who passed final examinations
10100025 Students to UNESCO-recordal
10100026 Students abroad to UNESCO-recordal
10100027 New entrants with completed vocational training
...

```

Figure 10: Extract from `thcodnam`

1.2.3.1 Installing program for creation thcodnam

The `thcodnam` creation tool requires a Java Runtime Environment (JRE) Version 1.5 or higher. It runs on all operating systems where Java is executed. The `Thcodnam.jar` program may be installed on any computer by copying it into any directory of the file system. To ease the call from the prompt the best way is to copy it into a directory not too deep in the file system.

The program communicates via the socket connection with the ICE-application-server, which is launched as daemon on the server side and listens to client requests. The information regarding the database connection the application server takes from the configuration file `config/ICETab.cfg`.

To start the application server one changes in a console window to the directory with the ICE-server:

```
cd %TOMCAT_HOME%/webapps/iceproject/WEB-INF/classes
```

Prior to the start the path of the classes must be set:

```
export CLASSPATH=./path/to/jdbcdriver:/path/to/log4jjar
```

The daemon process will be started by root:

```
/path/to/java server.TabServer [portnr]
```

and runs on port 50 by default.

1.2.3.2 Starting thcodnam creation tool

In a console window go to the directory with Thcodnam.jar and start the program as following:

```
/path/to/java -jar Thcodnam.jar
```

thcodnam file can be created either from the database or from file. Depending on the chosen way it has to be either the database connection created or file name specified. The method from database however is to prefer to be sure all required keys for the subsequent database import have been entered properly.

To load keys from a database the menu entry `File/Load Keys/From database` is to select. In a new dialog the name of the host machine (port if required), on which the application server runs, as well as the database segment and the password must be entered. By clicking the `Connect` button the socket connection will be established.

Then the menu entry `thcodnam/Create` is to select. By clicking it, the loaded ICE-keys are converted into the required format and you will be asked afterwards to locate a file in File-Dialog to which thcodnam shall be written.

1.3 Performing the conversion

If the files required for the conversion are created and copied into the directory structure for the project, one can start with converting:

- The file to be imported is copied into the project directory `Input`. The name of the import file may be defined arbitrarily, but is limited to 20 characters.
- The `Param` folder contains the parameter files required for converting (Control-Input, Control-Output, Control-Gendat, and assignments file and sum column specification file as appropriate).
- Directly in the project directory there are the `thcodnam`, `ClusNam` files.

In a console window, for the conversion one changes over to the project directory.

If the `ClusNam` file is not yet available, it can be created by copying `ClusNam.ori`.

```
cp ClusNam.ori ClusNam
```

Before converting it should be made sure that the row information in `ClusNam` matches with the actual size of the `cluster.son` files. If this is not the case, the information in `ClusNam` must be corrected. The differing information may occur if the last conversion process has been cancelled.

The program assumes that all files are available in Unix format. Therefore it is recommended to convert all the files in the `Input` and `Param` directories into Unix prior to starting the conversion, by executing the `convertfiles` script:

```
./bin/convertfiles
```

The program is started with:

startgendat <NameOfControlGendatFile>

Depending on the size of the original file and the number of the transformations the conversion may take a few seconds, but some hours as well.

When starting the program copies the Control-Gendat file specified into the project directory under the name of Control-Gendat. Thereupon it verifies whether all the files required are available, reads them and initially creates an intermediate file. This intermediate file contains fields for all key combinations, which may emerge from crossing the read field values with those to be calculated. Once this huge intermediate file matrix is ready it will be filled up with the values of the original file. Thereafter the matrix fields will be filled up which emerge from the transformations.

1.4 Conversion result

Following the successful conversion the program writes the conversion result into z- and i-files. The z-file contains all the sums with their coding included, and the i-file contains the coding of the sums only. The z-file is always one row longer than the i-file, since the first row of the z-file contains information regarding the sum columns and the length of the index field.

The width of sum columns is calculated by the program automatically.

Furthermore a row is entered into the cluster.dat file for each file transformed. It contains the names of the z- and i-files, the reading format for the i-file, and describes which keys are involved in the coding of the sums. To begin with, each data file is described with one respective key of the so called managing characteristics: data source, type of presentation, and data quality. The other characteristics follow thereafter. To keep it compact only the 'from' and 'to' attributes are given for each characteristic involved (except managing characteristics). If the attributes lying in between are missing in the converted file, they will be entered into the cluster.son file by the program. With the characteristic in the cluster.dat file the 1 (instead of the 0) appears immediately after the characteristic number. The corresponding row numbers of the cluster.son file are given at the end of the cluster.dat row. In the cluster.dat file the column characteristic is marked by a '-' previous to the characteristic number. The second-to-last digit in the cluster.dat row indicates, how much decimal places the sums in the z-file have.

The cluster.dat file can contain 200 rows maximum.

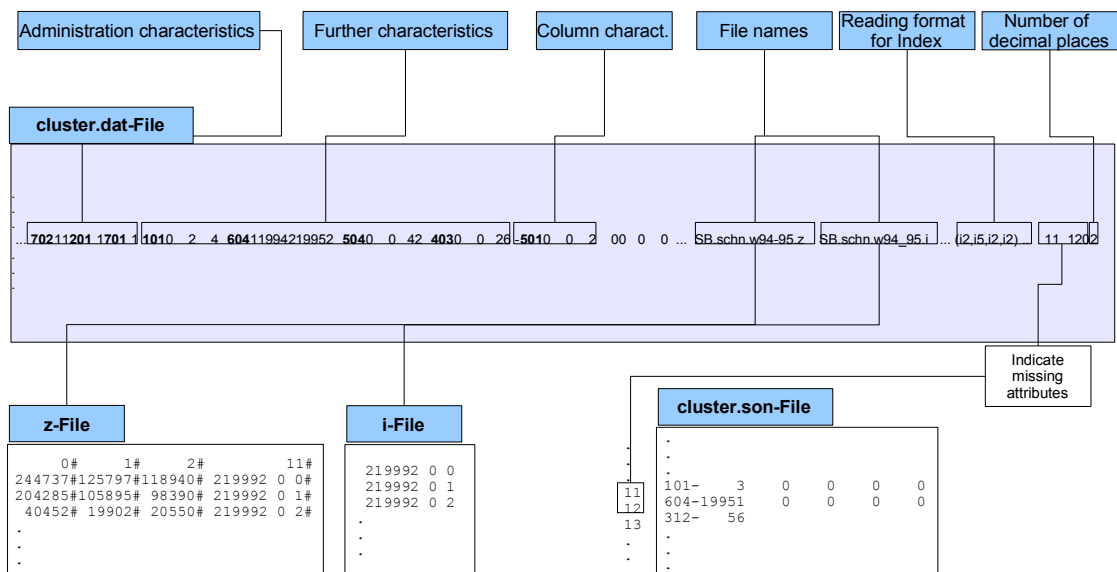


Figure 11: Conversion result

1.4.1 cluster.dat.x

Each topic area has a unique cluster.dat.x file. The trailing 'x' denotes the topic area number. After each conversion, 'gendat' will append a new line to the relevant cluster.dat.x file.

The format of such a line is as follows.

From digit	To digit	Description
1	12	Deprecated
13	27	administrative characteristics
13	15	Characteristic number for the source
16	17	Attribute number for the source characteristic
18	20	Characteristic number for the notation
21	22	Attribute number for the notation characteristic
23	25	Characteristic number for the data quality
26	27	Attribute number for the data quality characteristic
28	347	further characteristics – row and column characteristics
...		
28	32	Characteristic number (negative characteristic number for column characteristic)
33	33	Flag for missing values (0 for no missing values, otherwise 1)
34	38	From-attribute
39	43	To-attribute
...		
348	367	Name of the Z-file
368	387	Name of the I-file
388	393	(Number of the start line) deprecated
394	399	Number of lines in the I-file
400	429	Reading format for the keys in th i-file or z-file
430	434	From-row in cluster.son.x
435	439	To-row in cluster.son.x
440	440	Number of decimal places for the sums in z-file
441	441	Not important

1.4.2 cluster.son.x File

Each topic area has a unique cluster.son.x file. The trailing 'x' in the file name denotes the topic area number.

This file contains the missing attributes for each topic area. After each conversion 'gendat' will append rows to cluster.son.x file in following format. If there are no missing attributes for a topic area, the relevant cluster.son.x file will remain empty. If there are missing attributes for a topic area, at least found during one, of many conversions, they will be indicated here.

From	To	Description
1	3	Characteristic number of the missing attribute(s)
4	4	Minus (-) indicating missing
5	29	Attribute number for each missing attribute (5 digits). Placeholder value is 0.

1.5 Analyzing conversion log

After finishing the conversion or while the program is still running the log file in the project directory should be checked for the conversion being executed successfully. This file contains information regarding the course of the conversion:

- copy of the Control-Gendat file with all files involved with the conversion run,

```
Unterprogramm Files
 1 F 30      Input/pers_2005.prn
 2 F 22      Param/pers_2005.stout
 3 F 41      Param/pers_2005.stles
 4 F 20      Output/pers_2005.z
 5 F 21      Output/pers_2005.i
 6 F 14      72 ClusNam
 7 F 40      Param/zuord.gus
 8 F 46      Param/pers_2005.ZeiVor
Dateiende von Steuer-Gendat erreicht: Zeilenanzahl= 9
Ende von Files
```

Figure 12: Extract from log file (1)

- topic area determination using ClusNam

```
**** vor call idein
Suche Cluster-Datei passend zur Variable 102
ClusNam Zugriffsindex: 1
0 101 1cluster.dat.1          cluster.son.1
ClusNam Zugriffsindex: 2
0 102 2cluster.dat.2          cluster.son.2
```

Figure 13: Extract from log file (2)

- size of the intermediate file to be created,

```

Zeilenanzahl fuer Zwischendatei (Saetzi): 630
Zahl der Datenwerte pro Ausgabezeile (Wanz): 3
Auspraee-Format:>(i2,i5,i2,i1,i1)      <
Auspraegungs-Leseformat:
(4A12)
Vorbsetzen der Zwischendatei
Es wird jetzt eine Zwischendatei von etwa 7 KBytes angelegt.

```

Figure 14: Extract from log file (3)

- number of rows in the beginning that should not be read

```

3 Headerzeile(n) zum Ueberlesen:
                                     492         1         1
                                     493         2         1
        605         102         490         491

```

Figure 15: Extract from log file (4)

- list of the field values being skipped or added-up during reading of the original file, respectively,

```

===== UMSETZEN DER DATEN =====
Letzter Datensatz gelesen und in Temporaerdatei geschrieben!
Es wurden 622 Datenelemente in die Zwischendatei geschrieben
Es wurden 87 Datenzeilen aus der Eingabedatei (ohne Header) gelesen.
Es wurden zu 74 Werten ungleich Null Datenwerte aufaddiert.

```

Figure 16: Extract from log file (5)

- the calculations being performed,

```

1. Summe  Summationsvariable= 493
Auch unvollstaendige Summanden werden geschrieben
Summenformel fuer Variable 493:
      0 = +      1 +      2
Minnimaler, Maximaler Datenwert: 0 4768
Anzahl der zu bestimmenden Summen: 630
Anzahl der geschriebenen Summen: 369

```

Figure 17: Extract from log file(6)

- the list of the keys missing in the initial file,

```

### ***** Fehlende Auspraegungen
Kontrollieren ! *****
Sonderfall: Fuer var      504 Auspraegung      30
Keine Datenwerte gefunden!
ASCII-Wert vom "Zeilenende" chu:      10
### Warnung: Fuer var      305 Summationsdummy-Auspraegung      106078
Keine Daten-
werte gefunden! (Wird nicht als Sonderfall eingetragen.)
Sonderfall: Fuer var      312 Auspraegung      48
Keine Datenwerte gefunden!
Sonderfall: Fuer var      312 Auspraegung      56
Keine Datenwerte gefunden!
### *****

```

Figure 18: Extract from log file (7)

- the number of the rows written in the z- and i-files.

Es wurden 630 Sätze in die Zwischendatei geschrieben.
Es wurden 369 Sätze in die Ausgabe geschrieben.

Figure 19: Extract from log file (8)

In case of a fatal error (e.g. when attempting to read the original data with a wrong reading format) processing will be cancelled. The error message including the cause of the cancellation appears in the log file.

2 Data base import

2.1 Installing the import program

The import program requires a Java Runtime Environment (JRE) Version 1.5 or higher. It runs with all operating systems where Java is executed (Windows, Linux etc.). JDBC drivers are used for the database connection.

For the installation the Java classes of the ICE-server have to be copied into an arbitrary directory of the file system. The best way is to use the classes already filed with the ICE-installation into the directory `%TOMCAT-HOME%/webapps/iceproject/WEB-INF/classes`.

The file `config/ICETab.cfg` has to be adapted for the configuration of the import program. It contains property pairs required by various ICE-applications. The property names start with a '\$', the values with a '#'. For the import program those data are of relevance which define the database connection: `DatabaseHost`, `DatabaseName`, `DatabaseURL`, `DatabaseDriver`, `UserName`, `openDatabase`, `dbsystem`. The structure of the import project may be configured here as well (`MetaDataPath`, `DataPath`).

2.2 Creating the import project

For the import into the database the following files of the conversion run are used:

- z- and i-files,
- cluster.dat and cluster.son files,

a fixed directory structure being predetermined:

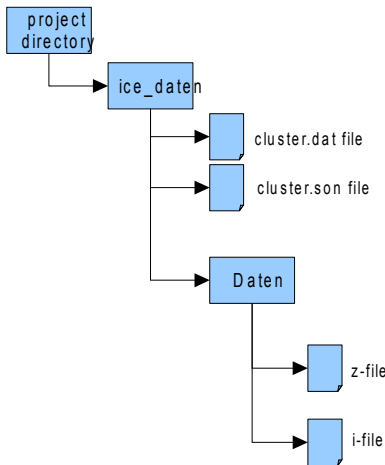


Figure 20: Layout of the project directory for the database import

The project directory may be given any name. It must contain a subdirectory `ice_daten`, where the cluster file with the data rows to be imported will be filed. The `ice_daten` folder

shall contain the directory Daten, where the z- and i-files to be imported are.

2.3 Performing the data import

Once the directory structure for the project is created with all the files required for the import, one can start with importing.

In a console window, for the import one changes over to the directory with the ICE-server.

```
cd %TOMCAT_HOME%/webapps/iceproject/WEB-INF/classes
```

Prior to the start the path of the classes must be set:

```
export CLASSPATH=./:/path/to/jdbcdriver:/path/to/log4jjar
```

The import is started by calling the Java class DataAdministrator in the dbimport package:

```
/path/to/java dbimport.DataAdministrator [/path/to/project]
[number of cluster.dat] [part of database] [check details j/n]
[tabnr to start (optionally)] [>name of log file]
```

The import is specified by the parameters:

- Path to the project directory.
- Number of the cluster.dat file. With a single call the data tables only from one cluster file can be read.
- Information regarding the sub-database: default `iceadmin`,
- Whether additional detailed data validation shall be performed. For this, the values of the current stock are compared to the similarly encrypted values of a previous point in time. The validation is very resource-consuming.
- The Id for the data table in the database is assigned consecutively. Sometimes there is a need for a data table to be assigned a particular Id. For this special case the desired consecutive number is given. This specification is optional.
- Finally the console output may be redirected into a text file for archiving purposes.

After the import is started the program checks if the files required are available. First the cluster.dat file is read, where the rows are checked, whether the specified z-file was already saved in the database. In such a case there is the corresponding entry in the database table `icetabellen`. If this is not the case, the meta information from the cluster.dat and cluster.son files are added to several database tables (see below), and subsequently the z-file is read into a new data table:

- `icetabellen`: contains name of z-file under a determined disposable 8-digit table id number.
- `tabellenimportiert`: contains the name of z-file as well as the ids of the tables newly imported
- `datenbestand`: contains control keys for the data stock as well as number of decimal places
- `tabverwaltungX`: contains to the table ids applicable characteristics with `from` and `to`-attributes (incl. Flag whether the attributes are missing in initial stock)
- `sonderfehlX`: contains keys missing in data stock
- `sonderzusX`: contains in data stock combined keys

- `spaltenmerkmale`: contains the column characteristic number in z-file
- `sondatentabs`: listed all data stock values being not numeric (e.g. 'k.A.')
- `datentab_XXXXXXXX`: contains the sum values incl. It's ICE-key-assignment.

By importing z-file the program verifies, where possible, if the data file and its meta information match. The result of this verification is recorded in the import log.

2.4 Evaluating the import log

For every import a log file `warnings_x.log` is created in the output folder, whereas x means the time and date of the import operation. The information of the import run is written there if the meta information for the data file was incomplete or erroneous, respectively:

- keys entered as missing in `cluster.son`, although they appear in the data file;
- keys missing in `cluster.son`, although they are not present in the data file.

The import program tries to correct the information or, respectively, writes into the log file what to check or add. After each import run the protocol shall be reviewed and processed, if necessary.

2.5 Deleting the data stock

If the files required for the database import are not found, or if there are problems with reading them or with writing into the database table, the import will be cancelled, and the error will be output on the console. After error correction the import may be done once more. One should make sure beforehand that the failed import did not leave any database entries behind. The simplest way to do this is to delete the data table, of which the import was cancelled. The required table-Id may be taken from the console output.

The `ImportRueckgaengig` java class in the `dbimport` package is for deleting database entries to the imported data tables, whereas files of the last import attempt or merely a single data table may be deleted.

The program will be started from the ICE-server home directory:

```
cd %TOMCAT_HOME%/webapps/iceproject/WEB-INF/classes
```

Prior to the start the path of the classes must be set:

```
export CLASSPATH=./path/to/jdbcdriver:/path/to/log4jjar
```

The call for the particular data table is:

```
/path/to/java dbimport.ImportRueckgaengig [tabid] [part of database]
```

where `tabid` is the 8-digit Id the data file got in the data base. The Id may be determined from the database table `icetabellen`.

If it is desired to remove all data tables read with the import operation executed last, the call is:

```
/path/to/java dbimport.ImportRueckgaengig all [part of database]
```

With this, all the data tables appearing in the database relation `tabellenimportiert` will be removed.

3 Adding the metadata in the data base

3.1 Adding the notes to the stock

For each imported data table additional text information may be added. This text information relates to the data itself, e.g. to refer to gaps or implausibilities, or relates to the encryption/boundaries of the data. In most cases it is intended to ensure the comparability of the figures from the data table to similarly coded figures with the notes.

At present the notes relate to a key appearing in the data table and being restricted to a temporal frame. They are filed in the `hinweise` database relation.

The notes may be entered into the data base using the `NotesAdmin.jar` tool.

3.1.1 Installing the notes administration tool

The notes administration tool requires a Java Runtime Environment (JRE) Version 1.5 or higher. It runs with all operating systems where Java is executed. The `NotesAdmin.jar` program may be installed on any computer by copying it into any directory of the file system. To ease the call from the prompt the best way is to copy it into a directory not too deep in the file system.

The program communicates via the socket connection with the ICE-application-server, which is launched as daemon on the server side and listens to client requests. The information regarding the database connection the application server takes from the configuration file `config/ICETab.cfg`.

To start the application server one changes in a console window to the directory with the ICE-server:

```
cd %TOMCAT_HOME%/webapps/iceproject/WEB-INF/classes
```

Prior to the start the path of the classes must be set:

```
export CLASSPATH=./path/to/jdbcdriver:/path/to/log4jjar
```

The daemon will be launched as root:

```
/path/to/java server.TabServer [portnr]
```

and by default will run on port 50.

3.1.2 Starting the notes administration tool

In a console window one changes to the directory with `NotesAdmin.jar` and starts the program with:

```
/path/to/java -jar NotesAdmin.jar
```

At first the socket connection will be established. For this, the server name (and port as appropriate) where the application server is running as well as sub-database and the password will be entered. With a click of the button `Connect`, the socket connection will be established. Thereupon the IDs of all the imported data tables will be loaded.

By multiple clicks the individual data tables may be selected or deselected, respectively. With `Insert note` the note entered in the text field with its key assignment (`Characteristic (prim. map.), Attribute`) and its time information (`Characteristic (sec. map.), From, To`) for the selected data table included will be inserted into the database. By entering „-1“ into the attribute field the note will be entered

for all attributes of the specified characteristic.

3.2 Updating the virtual stocks

3.2.1 *Virtual vs. physical stocks*

The data can be stored in the database in a number of different data tables called “physical stocks”. For a better overview they can be grouped to so called “virtual stocks”. So a virtual stock can represent a time series, while the data for the single time points are stored in different data tables.

A further example could be the case when there are several data tables of a topic area, which have different structure, but it's intersection of characteristics and attributes is of particular interest, e.g. a data stock for students broken down by fields of study and gender or the one for graduates broken down by fields of study and nationality. This intersection can be defined as an additional virtual stock in this case, e.g. as a stock for students and graduates broken down by fields of study.

While defining the request the user of the information system deals with the virtual stocks and does not realize, in which physical tables the data are actually stored. It is managed by the system.

3.2.2 *Original files with virtual stocks*

The list of virtual stocks is created based on the `cluster.dat.x` and `cluster.son.x` cluster files. In the simplest case the cluster files are copied to the files with the extension `.HC`. Each row of the `cluster.dat.x.HC` file represents the single virtual stock.

The virtual stock description in the HC - file is similar to the one of the physical stock in `cluster.dat` file. Firstly every virtual stock is described with one key of each of three so called Managing Characteristics: Data Source, Presentation Form and Data Quality. Then the further characteristics follow (Topic Area Characteristic and Time Characteristic among them). For each relevant characteristic (except Managing Characteristics) only from- and to- attributes are declared. In case the attributes between from- and to- are missing in the converted file, they are entered by the program into the `cluster.son.x.HC` file. In `cluster.dat` file the flag “1” (instead of “0”) will be set straight after the characteristic number in this case. The corresponding row numbers of the `cluster.son.x.HC` file are stated in the end of a `cluster.son.x.HC` row. The names of z- and i- files as well as the number of decimal places are not relevant for a virtual stock. The order of characteristics and the information about characteristic being in columns can differ from ones in physical stocks as well. Finally only the information which characteristics and attributes appear in a virtual stock is of importance.

In addition, every virtual stock gets an unique number that is entered into the `cluster.dat.x.HC` file (columns 388 – 393). Moreover, the flag ' 0' must be entered into the columns 394 to 399.

To create a new virtual stock just copy and paste a suitable cluster - row and adjust the characteristic and attributes entries afterwards: add a new year time point, for instance, if you would like to extend the time series. You can also delete a characteristic and its from- and to- attributes in case they should not be offered in a virtual stock.

3.2.3 cluster.dat.x.HC File

Virtual stocks are either newly created or updated based on the data being imported. If the data being imported defines a new year time point for existing data, then the relevant existing virtual stock is updated with the new time point attribute. If the data being imported are belonging to a new topic area or subject, then a new virtual stock is created.

Until the virtual stocks are created or updated, the new data can not be queried, using the flexible report generator.

To define a new virtual stock;

- 1) indicate the new virtual stock number (columns 388 – 393). Usually, this is the last virtual stock number for the topic area, incremented by one. If the last virtual stock number is 24, then the new stock number is 25.

eg: ... data_2008.i 25 ...

- 2) indicate the line numbers of the missing attributes contained in cluster.son.x.HC file.

eg: ... (i2,i5,i2,i2) ... 11 1202

cluster.son.x.HC file

11 -101 3 0 0 0

12 -1995 0 0 0 0

- 3) set the number within columns 394-399 to zero.

eg: 0(i2,i5,i2,i2)

To update a existing virtual stock;

- 1) Update the time attribute(s) to represent the union of time series data from the beginning time point year to the end time point year)

eg: The admission data exists from year 2004 to year 2006. It is needed to update the same virtual stock with year 2007 data. The existing definition contains attributes from year 2004 to year 2006. Therefore we need to change it from year 2004 to year 2007.

- 2) Check missing attribute(s).

eg: 2004 – 2006 } faculty engineering is missing
2007 } faculty engineering is available.

To make the engineering faculty visible in all years remove the missing attribute for faculty engineering from cluster.son.x.HC and cluster.dat.x.HC. Then the faculty will also be visible for the previous years, without any data values.

3.2.4 Format of the cluster.dat.x.HC file

From	To	Description
------	----	-------------

1	12	Deprecated
13	27	administrative characteristics
13	15	Common Characteristic number for the source
16	17	Common Attribute number for the source characteristic
18	20	Common Characteristic number for the notation
21	22	Common Attribute number for the notation characteristic
23	25	Common Characteristic number for the data quality
26	27	Common Attribute number for the data quality characteristic
28	347	further characteristics – row and column characteristics
...		
28	32	Common Characteristic number (negative characteristic number for column characteristics)
33	33	<p>Flag for missing values (0 for no missing values, otherwise 1) If any of the physical stocks is not having missing values, this flag must be 0. eg: 3011 0 12 60502004020040 (stock 1) 3010 0 12 60502007020070 (stock 2) Suppose stock 1 is missing the attribute 9. Then the virtual stock should be defined as 3010 0 12 6051200420070 The missing two years must be indicated in the cluster.son.x.HC file and the flag must be set to 1.</p>
34	38	<p>From-attribute describing the union of the sets of characteristics in each physical stock eg1: 60502005020050 60502006020060 ----> 60502005020070 60502007020070</p>
39	43	<p>To-attribute describing the union of the sets of characteristics in each physical stock eg1: 60502005020050 60502006020060 ----> 60502005020070 60502007020070</p>
...		
348	367	Name of the Z-file (Not Important)
368	387	Name of the I-file (Not Important)
388	393	Number of virtual stock

394	399	Must be ' 0'
400	429	Reading format for the keys in the i-file or z-file (Not Important)
		<p>For each physical stock containing missing values, its relevant lines are copied from cluster.son.x and appended at the end of the cluster.son.x.HC file and then the new starting row number and the ending row number are indicated here.</p> <p>Check if missing values are already available in other physical stocks, before indicating them as missing in the new virtual stock.</p> <p>Eg: an attribute missing in one physical stock may be available in another physical stock to be combined as a virtual stock. So it must not be indicated as missing in the cluster.son.x.HC file.</p>
430	434	From-row in cluster.son.x.HC
435	439	To-row in cluster.son.x.HC
440	440	Number of decimal places for the sums in z-file (Not Important)
441	441	Not important

3.2.5 Format of the cluster.son.x.HC File

From	To	Description
1	3	Characteristic number of the missing attribute(s)
4	4	Minus (-) indicating missing
5	29	Attribute number for each missing attribute (5 digits). Placeholder value is 0.

3.2.3 Reading the virtual stocks

The created HC files describe in keyed form the virtual stocks with the aggregated data. These are the stocks the user deals with while defining it's requests. Special Java tools are used for importing virtual stocks into the database.

The program for importing the virtual stocks requires a Java Runtime Environment (JRE) Version 1.5 or higher. It runs on all operating systems where Java runs (Windows, Linux etc.). JDBC drivers are used for the database connection.

For the installation the Java classes of the ICE-server have to be copied to any directory of the file system. The best way is to use the classes already filed with the ICE-installation into the directory %TOMCAT-HOME%/webapps/iceproject/WEB-INF/classes. The file `config/ICETab.cfg` containing the information regarding the database connection has to be customized.

To import the virtual stocks, the prepared HC files (cluster.dat.x.HC and cluster.son.x.HC) are copied to any directory.

Then change to the directory with the ICE-server:

```
cd %TOMCAT_HOME%/webapps/iceproject/WEB-INF/classes
```

Prior to the start the classpath must be set:

```
export CLASSPATH=./path/to/jdbcdriver:/path/to/log4jjar
```

The actual reading is done in two steps. The first step is to create two new files, BestandKlartext and BestandSchluessel, from the HC files:

```
/path/to/java tools.VirtualStockGenerator [/path/to/stock  
overviews] [part of database]
```

In this step the formatting of the HC file will be checked and it will be verified, whether the specified keys are available in the database. Should an error occur with this the processing will be cancelled, and the error message will appear on the console.

If the above mentioned text files have been created successfully, they may be read into the database in the second step:

```
/path/to/java tools.VirtualStockReader [/path/to/stock overviews]  
[part of database]
```

In doing so all the virtual stocks for the aggregated data are fully deleted from database before being read from the newly created BestandSchluessel file. Should an error occur with this the processing will be cancelled, and the error message will appear on the console.