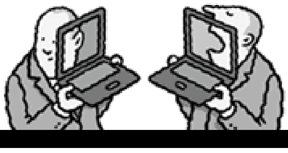


SIE file format

Version 4B – 30 Sept. 2008

Contents

1. Introduction

Background

1.1. There is an ever increasing need for the transfer of data between different types of programs. Standards have been prepared in a number of areas, either through standardization organizations or through the file format of a program producer becoming the de-facto standard.

1.2. There are however no standards in the financial reporting program world. Those wanting to transfer financial reporting data between programs from different suppliers have often been unable to obtain any assistance with this. Some program suppliers have implemented import or export routines which use another supplier's internal file format. This approach however results in substantial limitations.

1.3. A more practical alternative is where a common import/export file format is agreed by program suppliers. The SIE format is the result of this work.

Goals

1.4. The goals of the standard are as follows:

- Any doubt around whether two programs can communicate can be quickly laid to rest by determining whether both handle SIE files of a specific type. If both can handle SIE files of a specific type, then you can be certain transfer is possible.
- The file format should be able to be expanded without export files based on previous versions of the standard becoming unreadable.
- It should be easy to write export and import programs in most programming languages.
- The files should have a simple structure.
- The file format should be easy to read without the use of special software being required. The use of plaintext in the files allows them to be easily examined using a standard text editor or word processor. This is an important aspect and one that allows the functioning of the systems which generate the files to be simply verified.
- File format is to be as general as possible, so that the majority of variations in exportable data which arise in the different financial reporting programs can be handled. The majority of variations in data requirements in importing programs are similarly accommodated.

2. Compliance with the standard

2.1. A program supplier that chooses to export data in accordance with the SIE format specification is to support the entire file specification in their implementation. The program cannot be considered to comply with a SIE format if only some of the compulsory item types are exported.

2.2. An importing program is to be able to assume that a specific type of SIE file contains the items specified for the file type. Exporting only the item types you believe are required by a recipient program is absurd. This type of program does not follow the SIE standard and is not to be marketed as such.

2.3. The marketing of programs which generate SIE files is to state which type of SIE files can be generated. If a type is not specified, then type 1 specification should at least be followed. Producers of SIE input programs should similarly specify which types of files are required. File type is to be specified where SIE-gruppen's logotype is used.

2.4. These 'goods declaration requirements' have been set to allow those who acquire a program which contains SIE export and another which contains SIE import to be sure that data can be transferred between the two.

3. Test of SIE file format

List of approved programs

3.1. SIE-gruppen has prepared a list of approved programs which is based on testing and the results of this testing. The list is used in SIE-gruppen's information activities. To what extent the program can handle specific optional SIE format information is also specified in the list.

Approval duration

3.2. SIE-gruppen's list specifies from which program version SIE file support is available. The program supplier is to notify SIE-gruppen if the registered SIE file support partially or fully ceases.

3.3. Approvals can be revoked with immediate effect if it becomes apparent that a program's SIE file support is no longer in agreement with the information submitted in association with this approval.

3.4. Program suppliers who submit misleading or grossly incorrect information can be subject to disciplinary proceedings.

Use of SIE-gruppen's logotype

3.5. The approval gives the program supplier the right to use SIE-gruppen's logotype in the marketing of approved programs. The file types which the program is approved for are to be specified immediately below the logotype. An L is to be suffixed to the type number if the program is approved for reading a specific file type.

Example:

1, 2, 4, 4L	The program writes files in accordance with type 1, 2 and 4 and reads files in accordance with type 4.

Writing SIE files

3.6. Test files and a completed notification form are to be submitted to SIE-gruppen.

3.7. The test files are to contain a fully sufficient test material. Files submitted for testing which only contain a few items will be rejected, even though the file is correct. The file is to contain balances for the previous year, budget values and corresponding items to be considered to be fully sufficient.

Input

3.8. Programs are required to input the test files available on SIE-Gruppen's web site error free to be approved for input of a specific file type and the program supplier is to certify that this is the case.

3.9. A program which manages to input information from type 1 SIE files is of course to also be able to read this information from a type 2 or 3 file. The program is required to assimilate additional file information to be approved for type 2 or 3 input.

4. The different types of SIE files

General on the different file types.

4.1. The extent of the need to export information from financial reporting programs to post-processing systems can vary. The extent and comprehensiveness of information in financial reporting can also vary depending on financial reporting program options. The amount of information in a exported file, which is dependent on the software company's level of ambition, is therefore permitted to vary.

4.2. File types 1, 2 and 3 represent different levels of balance export from financial reporting programs. Type 4 specifies a file format for the import and export of verifications.

Type 1 - Export of closing balances

4.3. This type contains the item types which are required to export year end balances to different post-processing systems. Year end balances are often sufficient for income tax return or annual reporting programs.

4.4. File type 1 contains relatively little information and is permitted primarily for historical purposes. We recommend that all new SIE export implementations include minimum type 2.

Type 2 - Export of period end balances

4.5. Type 2 is an expansion of type 1 and should therefore contain all type 1 items. Period end balances and period end budget balances are also to be included in files of this type. The contents of this file type are intended used by different types of analysis program.

Type 3 - Export of object balances

4.6. This type of export format includes the export of balances at the object level. Programs from different producers handle objects in very different ways. An attempt has therefore been made to generalise the concept.

Type 4 - Import/Export of transactions

4.7. This file format is intended to be used to import and export transactions to/from financial reporting programs. A file which is generated by a pre-processing system such as an invoicing system, is an example of an import file. Using the SIE format allows a pre-processing system from one program producer to be used with a financial reporting program from another program producer. This is particularly relevant, as many pre-processing systems are custom made industry solutions. Pre-processing systems that support the SIE format provide a great deal more freedom when selecting a financial reporting program.

4.8. Transactions are also imported when special accounts closing programs are used. The transactions generated by the accounts closing program can then simply be returned to the financial reporting program.

4.9. The export of transactions is perhaps not as widespread. Computer supported audit does however have a great need for transactions export. An audit program can carry out various automatic checks on transactions exported from financial reporting programs.

4.10. Different import and export file formats is not desirable. Import and export therefore uses the same file format specification. Which items and fields are compulsory does vary.

5. The file format

File name and positioning

5.1. There are no restrictions on file naming. The following is however recommended:

- Type 1, 2, 3 and 4E files are given the file suffix .SE when exporting from financial reporting programs. Files which are designed to be imported by financial reporting programs, so called type 4I transaction files, are given the file suffix .SI. (See point 6. Compulsory and optional SIE file items).
- The files are placed in a sub-directory named \SIE.

5.2. The recommended file names for the transfer of transactions from the most common pre-processing systems are as follows

• Invoicing	FAKT.SI
• Accounts payable ledger	LEVR.SI
• Wages program	LÖN.SI

Read more about file names under point 7.4 Use of the flag item.

File content

5.3. Each file contains a number of items which can have different contents. Each item begins with a 'label' which marks item content. All labels begin with the character #. Examples of labels are #PSALDO which prefaces a period end balance item and #KONTO which prefaces an account item.

5.4. Some items can contain sub-entries. An example of a sub-entry are entry rows for a verification. These sub-entries are specified in braces ({ or }) immediately after the main item. These braces are to stand alone on their own rows.

5.5. Each item is to be ended with a line feed (ASCII 10). A carriage return (ASCII 13) before the end of the row is permitted but is not required.

5.6. Empty rows are permitted anywhere in the file and are to be ignored by the importing program.

5.7. The different fields within each item are to be delimited by one or more spaces. Tabulators (ASCII 9) are also accepted as being spaces. All fields are to be in quotation marks (ASCII 34). Quotation marks are however not a requirement and are only required when the field contains spaces. Quotation marks in export fields are to be preceded by a backslash (ASCII 92). There are to be no control characters in text strings. ASCII 0 up to and including ASCII 31 and ASCII 127 are control characters.

5.8. The character set used in the file is to be IBM PC 8-bits extended ASCII (Codepage 437)

5.9. Amounts are to be written using a point (ASCII 46) as the decimal delimiter. If no cents are specified, the amount is to be specified without decimals. Maximum two decimals are to be specified. Any minus signs are to be specified before the amount. Plus signs are not to be used. There are no amount size upper limits. The importing program should therefore check that its maximum limit is not exceeded.

5.10. Dates are always specified using the format YYYYMMDD.

General on SIE file structure

5.11. The items in SIE files are subdivided into groups.

5.12. Items are to occur in the file in the following sequence:

1 Flag item

2 Identification items

3 Chart of accounts information

4 Balance items/Verification items

5.13. There is no specified sequence for items in each group unless this is specified in the item specifications below.

5.14. All items marked as compulsory in the below table for the item type are to be found in the file. Special compulsory requirements apply to some item types. These are marked in the table and are described below.

The difference between a compulsory item and a compulsory field

5.15. It has been specified for each item type whether the item is compulsory in a specific type of SIE file. A summary of compulsory and optional items is provided in the table below. It is specified in the specification which fields are compulsory and which are optional for each item type. For example, the item type #BKOD is optional. The industry code field within the item is however a compulsory field. If a #BKOD item type is written in the SIE file, then an industry code must also be specified. If industry code information is not available when writing to files, then no #BKOD items are to be written.

Opening and closing balances

5.16. Both #IB and #UB item types are marked in the summary as being compulsory for both the current and previous year. The precise requirements for these items are as follows:

- #UB for the current year is to always be present.

- Either #UB for the previous year or #IB for the current year is to be present. We however recommend that both item types are written where the values are available.
- #IB for the previous year is completely optional. We however recommend that these items are written where the values are available.

When there is no balance

5.17. Balance items which are marked in the summary above as being compulsory items can under some circumstances be omitted. It is specified in the summary for example that #IB is a compulsory item. It is also specified that zero amount items do not need to be written in the SIE file. For example, a recently started company that does not have any opening balances will therefore not have any#IB items in the SIE file. This is of course quite in order. Similarly #PBUDGET items can be present in type 2 or 3 SIE files. If a budget has not been registered in a financial reporting program, then no budget items will be written.

5.18. The compulsory designation for these items should be interpreted as meaning that <u>if there are values which can be written, then these are to be written </u>.

6. Compulsory and optional items in the SIE file

Identification items:

		1	*2*	*3*	*4I*	*4E*
#FLAGGA	Flag item which specifies whether the file has been received by the recipient.	●	●	●	●	●
#PROGRAM	Which program generated the file.	●	●	●	●	●
#FORMAT	Which character set was used.	●	●	●	●	●
#GEN	When and who generated the file.	●	●	●	●	●
#SIETYP	Which type of SIE format the file follows.	○	●	●	●	●
#PROSA	Free comment text on file content.	○	○	○	○	○
#FTYP	Company type.	○	○	○	○	○
#FNR	The financial reporting program's internal code for an exported company.	○	○	○	○	○
#ORGNR	CIN of the exported company.	○	○	○	○	○
#BKOD	Exported company industry.	○	○	○	-	○
#ADRESS	Company address information.	○	○	○	○	○
#FNAMN	Complete name of the	●	●	●	●	●

	exported company.					
#RAR	Financial year from which the exported data is retrieved.	●†	●†	●†	○	●†
#TAXAR	Taxation year for income tax return information (SRU codes).	○	○	○	○	○
#OMFATTN	Dates of the period for the period end balances.	-	●	●	-	○
#KPTYP	Chart of accounts type.	○	○	○	○	○
#VALUTA	Financial reporting currency.	○	○	○	○	○

Chart of accounts information:

		1	*2*	*3*	*4I*	*4E*
#KONTO	Account information.	●	●	●	○	●
#KTYP	Account type.	○	○	○	○	○
#ENHET	Unit used in quantity reporting.	○	○	○	○	○
#SRU	RSV code for standardized accounts extracts.	●	●	○	○	○
#DIM	Dimension.	-	-	●	○	○
#UNDERDIM	Sub-dimension.	-	-	●	○	○
#OBJEKT	Object.	-	-	●	○	○

Balance items/Verification items:

#IB	Opening balance of a balance sheet account.	●†	●†	●†	-	●†
#UB	Closing balance of a balance sheet account.	●†	●†	●†	-	●†
#OIB	Opening balance of a balance sheet account.	-	-	●†	-	○
#OUB	Closing balance of a balance sheet account.	-	-	●†	-	○
#RES	Balance item for a profit and loss account.	●†	●†	●†	-	●†
#PSALDO	Period end balance for a specified account.	-	●	●	-	○
#PBUDGET	Period budget for a specified account.	-	●	●	-	○
#VER	Verification item.	-	-	-	○	○

#TRANS	Transaction item.	-	-	-	○	○
#RTRANS	Supplementary transaction item.	-	-	-	○	○
#BTRANS	Removed transaction item.	-	-	-	○	○
Control total items:						
#KSUMMA	Start of control summation/total.	○	○	○	○	○

Symbol key:

4I	=	4I refers to SIE type 4 which is intended to be used for import to a financial reporting program.
4E	=	4E refers to export from a financial reporting program as specified in SIE type 4.
○	=	Optional item.
●	=	Compulsory item. Zero amount balance items can however be omitted. Refer to the comments above.
†	=	Items for both current and previous financial years are to be present. Previous year items can be omitted where these figures are not available. Refer also to the comments below.
❖	=	Dimensions and sub-dimensions do not need to be declared if they coincide with the standard dimensions specified for SIE type 3.
-	=	The item is not to occur in this file type.

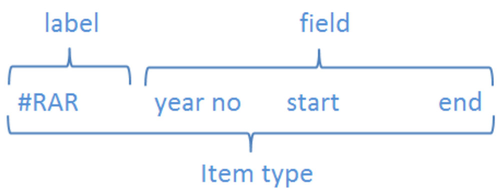
As declaring both accounts and objects in type 4I files is optional, it is assumed in principle that these are set up in an importing program before import, so that the contents of the file can be input in a correct way. Accounts and objects do not need to be set up in an importing company before the import is carried out if these items are declared in the file.

It is therefore recommended that the accounts and objects which are used in a SIE file are always declared under #KONTO and #OBJEKT even though this is optional.

7. To take into consideration when reading SIE files

Expandability

7.1. The file format is expandable by the specification being expanded with new labels. A reading program does not need to support all labels that occur.



For example, a post-processing system which does not need budget balances can disregard any budget items in the SIE file. An importing program is to permit the occurrence of unknown labels. Unknown item types should then be ignored.

7.2. Only labels specified in the SIE standard are to be found in a SIE file. This is to avoid problems arising when new labels are added to the SIE standard which others have previously used.

7.3. The format can also be expanded within current item types. An importing program is to therefore allow the occurrence of 'unknown' fields at the end of the item. This ensures that all software which reads the file format will be able to read future files in which the format has been expanded, without it being able to assimilate future information.

Use of the flag item

7.4. The flag in the flag item is used to receive a confirmation that a SIE file has been correctly transferred from a pre-processing system to a financial reporting program. This also functions as a crude reconfirmation channel. A certain degree of discipline of both the exporting pre-processing system and the importing financial reporting program is required for this to function. The following applies:

- The importing financial reporting program is not to remove input SIE files. The SIE files are deleted by the pre-processing system after the pre-processing system has established via the flag item that the transactions have been transferred correctly.
- The importing financial reporting program must signal that input has taken place through setting the flag in the flag item.
- A pre-processing system should use a predefined file name, so that this file name can be specified using receiving program system settings. Read more under point 5.1 File name and positioning.
- A pre-processing system can alternatively use a variable file name so that an optional number of SIE files can be exported to an 'Out tray'. The receiving program can then read this folder as its SIE files'In tray'and determine using the flag item which files can be imported. The exporting pre-processing system and the importing receipt program use suitable system settings to assign folders to the 'Out tray'and 'In tray'.

Immunity to extreme data

7.5. The SIE format specifies very few limitations on the size of data in individual fields. For example, the standard specifies no limits on the sizes of amounts which can be handled. With the exception of a few fields, no limitations on field lengths are specified.

7.6. Most input programs set system limitations on, for example, amount size or string length. The input program should therefore have special handling to handle the cases where an input file contains data which exceeds these limitations.

8. Object reporting in SIE files (type 3 and 4)

Declaration of dimensions

8.1. A description of the dimensions must be present in the file to allow the importing program to know the content of the file's object balances. Each dimension is assigned a dimension number here. Dimension relationships are also to be specified in the declaration, for example whether they are hierarchical. An input program can therefore use these declarations to determine how object balances are to be used and which balances are of interest.

8.2. A declaration can, for example, be as follows :

```
#DIM          1   "Cost centre"
#UNDERDIM     2   "Costs bearer"    1
#DIM          6   "Project"
```

8.3. The declaration of dimensions is in principle unrestricted. However some dimension numbers are reserved for a number of object types that normally occur.

8.4. Dimensions do not need to be declared where only reserved dimension numbers are used.

Universal dimensions

8.5. The content of the different dimensions are vary variable. A special 'content agreement' must often therefore be agreed between the sending and receiving program. There are however a number of commonly used dimensions (such as cost centre, costs bearer and project) which normally have a more universal content. Reserving specific dimension numbers for these dimensions allows a file format to be achieved in which object balances in most contexts can be transferred without 'special agreements' needing to be set up.

8.6. The SIE format's reserved dimension numbers are specified later in this document.

Single dimensions

8.7. In the simplest case, the dimensions have no relationship with each other. The dimensions are in this case declared as follows:

```
#DIM        7   'Employee number"
```

8.8. Objects within dimensions are exported in the following way:

```
#OBJEKT      7   "23"   "Sven Svensson"
```

8.9. Balances are exported as follows:

```
#PSALDO         0   200801   7010   {7 "23"}    200.00
```

Hierarchical dimensions

8.10. Some objects are a pure subdivision of the overlying objects. The code for the underlying object does not have any meaning if the overlying object is not specified. Department 01 can for example have three sub-departments (01, 02 and 03) while department 02 has four sub-departments (01, 02, 03 and 04). Only specifying the sub-department code (for example 02) does not provide complete information on which department is being referred to.

8.11. A sub-dimension is declared in the same way as a normal dimension is declared. Information on which dimension is highest in the hierarchy is however added. Example:

```
#DIM            20   "Department"
#UNDERDIM       21   "Sub-department"   20
```

8.12. When hierarchical objects are exported, balances for the overlying objects are exported individually and the balances for the underlying objects are exported individually. The codes for overlying and underlying objects are concatenated when exporting sub-objects.

8.13. Example:

```
#DIM            20   "Department"
#UNDERDIM       21   "Sub-department"   20

#OBJEKT      20   "01"     "Children's dept"
#OBJEKT      20   "02"     "Youth dept"

#OBJEKT      21   "0101"   "Infants"
#OBJEKT      21   "0102"   "Children 1-3 years"
#OBJEKT      21   "0103"   "Children 4-6 years"
#OBJEKT      21   "0201"   "Secondary school youth"
#OBJEKT      21   "0202"   "Sixth form college youth"

#PSALDO      0   200801   4010   {20 "01"}       49655.00
#PSALDO      0   200801   4010   {20 "02"}         200.00
#PSALDO      0   200801   4010   {21 "0101"}     13200.00
#PSALDO      0   200801   4010   {21 "0102"}      7800.00
#PSALDO      0   200801   4010   {21 "0103"}     28655.00
#PSALDO      0   200801   4010   {21 "0201"}       200.00
```

8.14. Uniform code string length is assumed. If the length of the code string is permitted to vary, concatenation of code and sub-codes can give non-unique codes. Example: dept 21 sub-dept 410 and dept 214 sub-dept 10 gives the same concatenation code (21410). Exporting programs which allow varying code lengths can solve this problem by adding a space (or other fill character). The example above gives the concatenation codes 21¤¤¤410¤¤ and214¤¤10¤¤¤ where filled out with five ¤ characters.

Sub-accounts

8.15. Sub-accounts often occur where further specification of a specific account (or a specific account group) is required. A type example is subsidiary ledger accounts which can be subdivided using customer or supplier numbers (or invoice numbers).

8.16. The sub-accounts are declared in the same way as single dimensions.

Reserved dimension numbers

8.17. The following dimension numbers are reserved:

1	=	Cost centre / result unit.
2	=	Cost bearer (is to be sub-dimension of 1).
3-5	=	Reserved for future expansion of the standard.
6	=	Project.
7	=	Employee.
8	=	Customer.
9	=	Supplier.
10	=	Invoice.
11-19	=	Reserved for future expansion of the standard.
20-	=	Available funds.

Use of dimensions and objects in type 4 import files

8.18. Transactions with an object specification are to of course only be used where this is justified. This applies to import where entries have already been specified in the pre-processing system using some form of object.

8.19. For example, a salary system where the generated entries can be specified using employee numbers. The wages system, when it generates the SIE-file, declares that it uses dimension 7 to specify employee number.

```
#DIM        7    "Employee number"
```

8.20. Using the predefined dimension number for employee number (number 7) allows the importing financial reporting program to know that the object specifications specified in the file are just employee numbers. If a financial reporting program cannot handle this information, it can choose to disregard the object specification.

Transaction object list

8.21. A transaction, in addition to account and amount, also includes information on which object the transaction relates to. This is achieved through a list of object numbers. This list contains paired dimension numbers and object numbers. Only the object numbers which are relevant to the transaction in question need to be specified.

8.22. Example:

```
#DIM        7    "Employee number"
#DIM        1    "Department"
#DIM        6    "Project"

#VER        A    567    20081216    "Cash salary"

{
        #TRANS    7010    {"1" "456" "7" "47"}     13200.00
        #TRANS    1910    {}                      -13200.00
}
```

8.23. Where hierarchical dimensions are imported, only the sub-dimension's code is to be specified in the sub-object's position in the object list. The overlying object must in these cases always be specified in the object list.

9. Quantity reporting in SIE files

General on quantity reporting

9.1. Some financial reporting programs have functions that allow quantities to be registered in transactions with amounts. These are balanced in the same way as amounts and provide the opportunity for example to extract cost/profit per unit. Some industries use quantity reporting to a much greater extent than others, for example agriculture.

9.2. All balance items and transaction items in the SIE format therefore have an optional field to facilitate transfer of this quantity reporting to different post-processing systems. These items are: #IB, #OIB, #OUB, #UB, #RES, #PSALDO and #PBUDGET.

9.3. The transaction items #TRANS, #RTRANS and #BTRANS in the SIE format have also been provided with an optional field for quantities.

9.4. A special item type #ENHET can be used to declare the units of the quantities specified for a specific account. Quantity designations can however be specified irrespective of whether a unit has been declared for the account.

10. Control summation

Purpose

10.1. Control summation is above all used to protect the user of SIE files from errors which are due to the corruption of data due to faults on storage media or errors in data communication. The proposed algorithm gives a very high probability of detecting these types of errors.

10.2. Control summation also provides a certain degree of opportunity for detecting deliberate manipulation of the SIE files. It should however be noted that control summation is not primarily intended as a protection against deliberate data manipulation.

Practical introduction to the SIE standard

10.3. The control total is an optional supplement to the SIE standard. An importing program which supports control summation is to therefore accept all input files which do not have a control total and is to check all input files which do have control totals.

Introduction of the control total in the SIE file

10.4. An SIE file which is control summated when generated is to contain two items of type #KSUMMA as specified in the following:

```
#FLAGGA 0
#KSUMMA


…
  Other items
…


#KSUMMA 1234567890
```

10.5. The first item of type #KSUMMA signals to the input program that control totals have been calculated and that a control total will be presented at the end of the file.

10.6. The importing program should refuse to import the file if the importing program detected the introductory control summation signal but could not find any final control total item. The file is defective (truncated).

Algorithm for control summation

10.7. An algorithm CRC-32, which is used within telecommunication and fax transmissions, is used for control summation.

10.8. The algorithm is based on each bit in the text (the message) being considered to be coefficients in an polynomial. This polynomial is divided by a generator polynomial. The remainder from this division is used as a control total. The technique provides a very high degree of safety, as each bit in the message affects the control total. The statistical distribution of the control totals generated by this algorithm is in addition very even, which is a hallmark of a good algorithm.

10.9. Polynomial division is a relatively complicated operation. Algorithms have therefore been developed in which the calculation is simplified through searching in a generated table. This type of algorithm has been attached to this document.

10.10. We have chosen to not describe the technical details of the algorithm here. Those interested can obtain additional information from the following articles:

- "A Tutorial on CRC Computations.", IEEE Micro, August 1998.
- "File verification using CRC", Dr. Dobb´s Journal, May 1992.

Parameters for control total algorithms

10.11. The generator polynomial which is used in control summation of SIE files, is to have the coefficients EDB88320H.

10.12. The control total is before calculation to be set to FFFFFFFFH (pre-conditioning).

10.13. The control total is to be bit inverted after the calculation has been completed (post-conditioning).

Which parts of the file contents are to be control summated?

10.14. The following instructions must be followed precisely if writing and reading programs are to arrive at the same control totals:

- Control summation begins with the first item after the introductory #KSUMMA and includes all items up to (but not including) the final #KSUMMA.
- The control total calculation includes the item labels which preface each item and the fields within the item.
- Spaces and tabulators between the fields are not to be included in the control total calculation.
- Quotation marks which preface and end fields are not to be included in the control total calculation.
- Braces around object lists or entry items are not to be included in the control total calculation.
- The control total is calculated using the character values specified in IBM PC-8 character set (codepage 437).
- Quotation marks within fields are marked with a 'backslash'. However, only the quotation marks are to be included in the calculation of the control total.

Example

10.15. Characters which are included in the control total are underlined below in the following example

#KONTO 1915 "Kassa \"special\""

Code example of control summation

```
#define CRC32_POLYNOMIAL 0xEDB88320L


unsigned long CRCTable[ 256 ];
unsigned long crc;                  // Global variable to accumulate CRC

// This routine is to be called to initiate the lookup-table
// before the calculation is begun

void CRC_skapa_tabell()
{
    int i;
    int j;
    unsigned long crc;

    for ( i = 0; i <= 255 ; i++ )
    {
        crc = i;
        for ( j = 8 ; j > 0; j-- )
        {
            if ( crc & 1 )
            {
                crc = ( crc >> 1 ) ^ CRC32_POLYNOMIAL;
            }
            else
            {
                crc >>= 1;
            }
        }

        CRCTable[ i ] = crc;
    }
}

// This routine begins the CRC calculation

void CRC_start(void)
{
    // Begin the CRC calculation by setting the global
    // CRC accumulator for the pre-conditioning value.
```

```
        crc = 0xFFFFFFFFL;
}

// This routine is called for each text section which is to be included
// in the control total

void CRC_ackumulera(void *buffer, unsigned int count)
{
        unsigned char *p;
        unsigned long temp1;
        unsigned long temp2;

        p = (unsigned char*) buffer;

        while ( count-- != 0 )
        {
                temp1 = ( crc >> 8 ) & 0x00FFFFFFL;
                temp2 = CRCTable[ ( (int) crc ^ *p++ ) & 0xff ];
                crc = temp1 ^ temp2;
        }
}

// This routine ends the CRC calculation and returns the
// resulting CRC value.

unsigned long CRC_retur(void)
{
        // Carries out post-conditioning of the cumulative CRC value by
        // bit inverting in accordance with post-conditioning mask

        return( crc ^ 0xFFFFFFFFL );
}
```

11. Description of all item types

#ADRESS Address information for the exporting company

Format:

```
#ADRESS    contact    distribution address    postal address    tel
```

Description:

1. Address information for exported company.

Example:

```
#ADRESS    "Sven Svensson"    "Box 21"    "211 20 MALMÖ"    "040-123 45"
```

#BKOD Industry code for the exported company

Format:

```
#BKOD    SNI code
```

Description:

1. Exported company industry. Industry is specified using an SNI code. This can be used in an analysis program for comparisons with industry average values.

Example:

```
#BKOD    82300
```

#DIM Single dimension

Format:

```
#DIM    dimension no    name
```

Description:

1. Declares a single dimension. The reserved dimension number should be used if the dimension you intend to declare is one of the universal dimensions. A dimension number within the unrestricted range should otherwise be specified.

Example:

```
#DIM    1    "Department"
#DIM    6    "Project"
```

#ENHET Unit for quantity reporting

Format:

```
#ENHET    account no    unit
```

Description:

1. Specifies units for the quantities reported on the specified account. The account must have been declared previously in the file using an item of type #KONTO.
2. Specifying units is optional. Quantities can be specified in balance or transaction items for accounts which do not have unit items.

Example:

```
#ENHET          4010     litre
```

#FLAGGA Import flag

Format:

```
#FLAGGA         x
```

Description:

1. The first item in each file is a flag item, which indicates whether the file has been input or not. It is set to 0 by the program which writes the file. A one is written in the zero position where it is successfully input. This prevents the same file being imported twice. The program which generates import files can, using the same method, check that previous files have been input before a new file is written. This is particularly important in pre-processing systems which generate verifications (files of type 4I), as these are not to be overlooked nor be imported twice.
2. This flag only needs to be handled for accumulated import (for example import of verifications). Programs in which the import of the file twice has no effect do not need to take note of this flag.

Example:

```
#FLAGGA         0
```

#FNAMN Complete name of the company which is exported.

Format:

```
#FNAMN          company name
```

Description:

1. The company's legal name.

Example:

```
#FNAMN          "Målerifirman Axelsson & Johnsson AB"
```

#FNR Financial reporting program's internal code for the company which is exported

Format:

```
#FNR        company id
```

Description:

1. Financial reporting system's identity term for the company. Different types of financial reporting systems can use very different company identifiers. This code is important for example in an accounts closing program which is to re-export accounts closing transactions back to the financial reporting program.

   Example:

```
#FNR        Kalles
```

#FORMAT Specifies which character set is used

Format:

```
#FORMAT        PC8
```

Description:

1. Until further notice, the standard only permits IBM Extended 8-bit ASCII (PC8 - codepage 437).

   Example:

```
#FORMAT        PC8
```

#FTYP Type of company

Format:

```
#FTYP        Company type
```

Description:

1. Company type is used to identify the type of company. Is for example used by importing programs to specify which SRU codes set up is to be used.
2. The company types in the list are Bolagsverket's definition of company types.

   X i.e. Other company type is to be specified for foreign company types or company types which are not found in the list.

AB	=	Limited liability company.
E	=	Sole proprietor.
HB	=	Trading company.
KB	=	Limited partnership.
EK	=	Economic association.
KHF	=	Co-operative tenancy association.
BRF	=	Tenant-owner's association.
BF	=	Housing association
SF	=	Allotment cooperative.
I	=	Non-profit association which operates business activities
S	=	Foundation which operates business activities.

FL	=	Foreign company branch.
BAB	=	Joint-stock bank.
MB	=	Members' bank.
SB	=	Savings bank.
BFL	=	Branch of foreign bank.
FAB	=	Limited liability insurance company.
OFB	=	Mutual insurance company.
SE	=	Euro company.
SCE	=	European co-operative association.
TSF	=	Religious community.
X	=	Other company types.

Example:

```
#FTYP         AB
```

#GEN Specifies when and who generated the file

Format:

```
#GEN          date      sign
```

Description:

3. Dates are specified using the YYYYMMDD format and is compulsory information. Signature can be the name, signature or user id of the person or process that generated the printout. Signature can be omitted.

Example:

```
#GEN          20080518   AN
```

#IB Opening balance for balance sheet account

Format:

```
#IB         year no    account    balance    quantity
```

Description:

1. Credit balance is designated by a negative amount.
2. Year no is specified using 0 for the current year and -1 for the previous year. If you would like to export additional comparison years, these can be added as years -2, -3 etc.
3. The quantity field is optional. The quantity is to normally be specified using the same characters as the balance amount, i.e. plus for debit and minus for credit.

Example:

```
#IB         0    1930    23780.78
```

#KONTO Account information

Format:

```
#KONTO          account no     account name
```

Description:

1. Specifies the account name of an existing account.
2. Account number is to be numeric.
3. When exporting, all used accounts are to be exported.
4. For import, it is assumed that the accounts which are used in the transactions have been set up in the financial reporting program's chart of accounts. The import file is to otherwise to contain suitable account items.
5. The importing program sets up a new account for account items found in the import file but not found in the financial reporting program's chart of accounts.
6. The importing program should make the user aware of any account items in the import file that have a name which does not match the name of the corresponding account in the financial reporting program's chart of accounts.

Example:

```
#KONTO          1510    "Accounts receivable"
```

#KPTYP Type of chart of accounts

Format:

```
#KPTYP          type
```

Description:

1. Specifies which charts of accounts type the exported chart of accounts is based on. The item is optional. If this item is missing, an input program should assume that the chart of accounts follows BAS 95.
2. BAS95, BAS96, EUBAS97 or NE2007 can be specified as type.
3. NE2007 relates to BAS 2007 for sole proprietors (K1).
4. SIE-gruppen will only introduce new permitted charts of accounts types if the structure of these deviate from any of the above. If for example BAS-gruppen publishes a EU- BAS 98 which has a structure which corresponds to EU-BAS 97, EUBAS97 will continue to be specified in the SIE file.
5. If the chart of accounts type begins with BAS2, i.e. BAS2008, this is to be handled as a chart of accounts of type EUBAS97.

Example:

```
#KPTYP          EUBAS97
```

#KTYP Account type

Format:

```
#KTYP        account no      account type
```

Description:

1. Account type is specified as T, S, K or I (asset, debt, cost or income). The equivalent account item must have occurred previously in the file for an account type to be specified.
2. The item type is optional. If an account type is not specified, it is assumed that the account type is as specified by the BAS standard.

Example:

```
#KTYP        1510   T
```

#OBJEKT Object

Format:

```
#OBJEKT        dimension no    object no    object name
```

Description:

1. Used for exporting existing objects within a dimension.
2. The importing program may choose to not input/set up the object if an object is not set up in the importing program and is not declared under #OBJEKT, but still occurs on #TRANS rows in the import file (files of type 4I).

Example:

```
#OBJEKT        1    "0123"    "Service department"
#OBJEKT        1    "0124"    "Sales department"
#OBJEKT        1    "0125"    "Development department"
```

#OIB Opening balance for object

Format:

```
#OIB        year no    account    {dimension no object no}    balance    quantity
```

Description:

1. Credit balance is designated by a negative amount.
2. Year no is specified using 0 for the current year and -1 for the previous year. If you would like to export additional comparison years, these can be added as years -2, -3 etc.

3. The quantity field is optional. The quantity is to normally be specified using the same characters as the balance amount, i.e. plus for debit and minus for credit. Account unit is defined under #ENHET.

Example:

```
#OIB        0    1510    {8 "12345"}     23780.78
```

#OMFATTN Date for period of period end balance

Format:

```
#OMFATTN        date
```

Description:

1. Specifies period (date up to and including) of the balances exported in the file. This is specified to inform those who read the file that the file's balances only are for parts of the year. The date for last period accounts close or last closing period is normally specified. The item must be present for export of period end balances.

Example:

```
#OMFATTN        20080630
```

#ORGNR Specifies corporate identification number of the company which is exported

Format:

```
#ORGNR        CIN      acq no      act no
```

Description:

1. Specifies corporate identification number of the exported company. The corporate identification number is to contain a hyphen after the sixth digit.
2. Acquisition numbers and activity numbers are not compulsory and are only specified when the information is available. Acquisition number is used to differentiate companies with the same corporate identification number (Arises when a person operates several individual companies).
3. From the 1995 taxation year, companies are no longer subdivided in income tax returns by acquisition numbers and activity numbers. A serial number is given to companies which have the same corporate identification number. If this serial number can be registered in a financial reporting program, then this should be exported in the field acq no.

Example:

```
#ORGNR        556334-3689    1
```

#OUB Closing balance for object

Format:

```
#OUB    year no    account    {dimension no object no}    balance    quantity
```

Description:

1. Credit balance is designated by a negative amount.
2. Year no is specified using 0 for the current year and -1 for the previous year. If you would like to export additional comparison years, these can be added as years -2, -3 etc.
3. The quantity field is optional. The quantity is to normally be specified using the same characters as the balance amount, i.e. plus for debit and minus for credit. Account unit is defined under #ENHET.

Example:

```
#OUB        0    1510    {8 "12345"}      23780.78
```

#PBUDGET Period budget item.

Format:

```
#PBUDGET    year no    period    account    {dimension no object no}    balance    quantity
```

Description:

1. The item specifies the change on the account in the period. Credit balance is designated by a negative amount.
2. Year no is specified using 0 for the current year and -1 for the previous year. If you would like to export additional comparison years, these can be added as years -2, -3 etc. The financial year is defined under #RAR.
3. Period is specified as YYYYMM (where YYYY refers to calendar year and MM refers to calendar month).
4. Object specification (dimension no and object no) is omitted when exporting period end balances in accordance with SIE type 2. Programs which input SIE type 2 are to disregard items which have object specifications which are not empty.
5. Items are to be specified both for the account as a whole (i.e. without object specification) and for the object when exporting in accordance with SIE type 3.

6. The quantity field is optional. The quantity is to normally be specified using the same characters as the balance amount, i.e. plus for debit and minus for credit.

Example:

```
#PBUDGET     0    200801    3011    {} -1243.50             -415
#PBUDGET     0    200801    5010    {1 "0123"}           3411.80
```

#PROGRAM Specifies the program which exported the file

Format:

```
#PROGRAM          program name     version
```

Description:

1. Specifies which program exported the file.

Example:

```
#PROGRAM          "Visma Compact"          5.1
```

#PROSA Free comment text on file content.

Format:

```
#PROSA          Text
```

Description:

1. Can for example specify who or which program generated the file.

Example:

```
#PROSA          "Exported using Visma Compact 080512"
```

#PSALDO Period end balance item

Format:

```
#PSALDO   year no   period   account   {dimension no object no}   balance   quantity
```

Description:

1. The item specifies the change on the account in the period. Credit balance is designated by a negative amount.
2. Year no is specified using 0 for the current year and -1 for the previous year. If you would like to export additional comparison years, these can be added as years -2, -3 etc. The financial year is defined under #RAR.
3. Period is specified as YYYYMM (where YYYY refers to calendar year and MM refers to calendar month).
4. Object specification (dimension no and object no) is omitted when exporting period end balances in accordance with SIE type 2. Programs which input SIE type 2 are to disregard items which have object specifications which are not empty.
5. Items are to be specified both for the account as a whole (i.e. without object specification) and for the object when exporting in accordance with SIE type 3.

6. The quantity field is optional. The quantity is to normally be specified using the same characters as the balance amount, i.e. plus for debit and minus for credit.

Example:

```
#PSALDO        0    200808    1910    { }           1243.50   123
#PSALDO        0    200809    5010    {1 '0123'}    3411.80
```

#RAR Financial year from which the exported data is retrieved.

Format:

```
#RAR        year no    start    end
```

Description:

1. The financial year start and end date is specified in the format YYYYMMDD. Year no is set to 0 for the current year and -1 for the previous year.
2. If you would like to export additional comparison years, these can be added as years -2, -3 etc.
3. Note that the SIE file only contains one chart of accounts (that for year 0). All data for the comparison year must be standardized to this chart of accounts.

Example:

```
#RAR        0     20080101    20081231
#RAR        -1    20070101    20071231
```

#RES Balance for profit and loss account

Format:

```
#RES        year's   account   balance   quantity
```

Description:

1. Credit balance is designated by a negative amount. Year no is specified using 0 for the current year and -1 for the previous year. If you would like to export additional comparison years, these can be added as years -2, -3 etc. The financial year is defined under #RAR.
2. The quantity field is optional. The quantity is to normally be specified using the same characters as the balance amount, i.e. plus for debit and minus for credit. Account unit is defined under #ENHET.

Example:

```
#RES        0    3011    -23780.78
```

#SIETYP Specifies which file type within the SIE format the file follows

Format:

```
#SIETYP       type no
```

Description:

1. Specifies which file type within the SIE format the file follows The importing program can assume that the file follows SIE type 1 where file type is not specified.

Example:

```
#SIETYP    2
```

#SRU SRU code for transfer of account balances to an income tax return

Format:

```
#SRU    account    SRU code
```

Description:

1. Specifies for a specified account where the account's balance is to be positioned on the form for standardised accounts extracts. Accounts balances can be required entered under several headings on the form or on several income tax return forms. The file in this case is to contain several items with the same account number (one for each SRU code).
2. An amount can in some cases be reported under different SRU codes, depending on the amount's sign. In this case, only the single SRU code is to be specified in the SIE file using item type #SRU (that which it is equal to). The receiving program (the income tax return program) has already knowledge of the alternative codes and can transfer the amount to the code which agrees with the amount's sign.

Example:

```
#SRU    1910    7214
```

#TAXAR Taxation year which SRU codes relate to

Format:

```
#TAXAR    year
```

Description:

1. Income tax returns are changed each taxation year, which means that the SRU codes entered in the SIE file only apply to a specific taxation year. The exporting program uses this item to specify which year's income tax return the SRU codes relate to. An importing program should check this item to determine whether the SRU codes are correct for the specific income tax return. Year is specified in the form YYYY.
2. The item is optional.

Example:

```
#TAXAR    2008
```

#TRANS Transaction item

Format:

```
#TRANS  account no  {object list}  amount  transdate  transtext  quantity  sign
```

Description:

1. Transdate and transtext do not need to be specified.
2. If transdate is not specified, it is assumed that the transaction date is the date specified for verification date.
3. Transaction items are only to occur as #VER sub-entries.
4. Verifications are to balance. The total of all transaction amounts within a verification is to therefore be zero.
5. The quantity field is optional. The quantity is to normally be specified using the same characters as the transaction amount i.e. plus for debit and minus for credit.
6. Sign can be the name, signature or user id of the person or process which generated the transaction item. Signature can be omitted.

Example:

```
#VER    " "       20080101    "Postage"
{
        #TRANS    1910 { }    -1000.00
        #TRANS    2640 { }    200.00
        #TRANS    6250 { }    800.00
}
```

#RTRANS Supplementary transaction item

Format:

```
#RTRANS  account no  {object list}  amount  transdate  transtext  quantity  sign
```

Description:

1. Transdate and transtext do not need to be specified.
2. If transdate is not specified, it is assumed that the transaction date is the date specified for verification date.
3. Supplementary transaction items are only to occur as #VER sub-entries.
4. This item is a supplement in the SIE standard. There is to always be a row of type #RTRANS immediately followed by an identical row of type #TRANS so that backward compatibility with the old SIE formats is maintained.

Example:

```
#RTRANS     1910 { }     200.00
#TRANS      1910 { }     200.00
```

First comes the supplementary row. On the row immediately beneath the supplementary row is to be an identical row of the type #TRANS.

5. If items of the type #RTRANS and #BTRANS are not handled, these rows are to be ignored when importing a SIE file.
6. If items of the type #RTRANS are handled, the following row of type #TRANS is to be ignored when importing a SIE file.
7. The quantity field is optional. The quantity is to normally be specified using the same characters as the transaction amount i.e. plus for debit and minus for credit.
8. Signature can be the name, signature or user ID of the person or process which added the transaction item. Signature can be omitted.

Example:

```
#VER        " "         20080101      "Postage"
{
            #TRANS      1910 { }      -1200.00
            #TRANS      2640 { }      240.00
            #TRANS      6250 { }      960.00
            #RTRANS     1910 { }      200.00
            #TRANS      1910 { }      200.00
            #RTRANS     2640 { }      -40.00
            #TRANS      2640 { }      -40.00
            #RTRANS     6250 { }      -160.00
            #TRANS      6250 { }      -160.00
}
```

Example of how the verification can look after input, where items of the type #RTRANS are not handled:

Account	Text	Debit	Credit
1910	Postage		1.200,00
2640	Postage	240,00	
6250	Postage	960,00	
1910	Postage	200,00	
2640	Postage		40,00
6250	Postage		160,00
Balance sheet		1.400,00	1.400,00

total			

Example of how the verification can look after input, where items of the type #RTRANS are handled:

Account	**Text**	**Debit**	**Credit**
1910	Postage		1.200,00
2640	Postage	240,00	
6250	Postage	960,00	
1910	Postage	200,00	
2640	Postage		40,00
6250	Postage		160,00
Balance sheet total		1.400,00	1.400,00

#BTRANS Removed transaction item

Format:

```
#BTRANS   account no   {object list}   amount   transdate   transtext   quantity   sign
```

Description:

1. Transdate and transtext do not need to be specified.
2. If transdate is not specified, it is assumed that the transaction date is the date specified for verification date.
3. Removed transaction items are only to occur as #VER sub-entries.
4. If items of the type #BTRANS and #RTRANS are not handled, these rows are to be ignored when importing a SIE file.
5. The quantity field is optional. The quantity is to normally be specified using the same characters as the transaction amount i.e. plus for debit and minus for credit.
6. Sign can be the name, signature or user id of the person or process which removed (cancelled) the transaction item. Signature can be omitted.

Example:

```
#VER        " "        20080101    "Postage"
{

            #TRANS     1910 { }    -1000.00
            #TRANS     2640 { }    200.00
            #BTRANS    6110 { }    800.00
            #RTRANS    6250 { }    800.00
            #TRANS     6250 { }    800.00
```

}

Example of how the verification can look after input, where items of the type #BTRANS and #RTRANS are not handled:

Account	Text	Debit	Credit
1910	Postage		1.000,00
2640	Postage	200,00	
6250	Postage	800,00	
Balance sheet total		1.000,00	1.000,00

Example of how the verification can look after input, where items of the type #BTRANS and #RTRANS are handled:

Account	Text	Debit	Credit
1910	Postage		1.000,00
2640	Postage	200,00	
~~**6110**~~	~~Postage~~	~~800,00~~	
6250	Postage	800,00	
Balance sheet total		1.000,00	1.000,00

#UB Closing balance for balance sheet account

Format:

```
#UB        year no   account    balance   quantity
```

Description:

1. Credit balance is designated by a negative amount.
2. Year no is specified using 0 for the current year and -1 for the previous year.
3. The quantity field is optional. The quantity is to normally be specified using the same characters as the balance amount, i.e. plus for debit and minus for credit.

Example:

```
#UB         0    2440    -2380.39
```

#UNDERDIM Sub-dimension for hierarchical dimensions

Format:

```
#UNDERDIM      dimension no    name    superdimension
```

Description:

1. As #DIM above. Overlying dimensions are however also specified.

Example:

```
#UNDERDIM       21    "Sub-department"    1
```

#VALUTA Reporting currency

Format:

```
#VALUTA      currency code
```

Description:

1. The currency code applies to the entire contents of the SIE file.
2. Currency code is specified in accordance with ISO 4217. The item is optional. If this item is missing, an input program should assume that the currency code is SEK. The currency code is optional. This means that it cannot be assumed that an importing program reads this item even where it is specified in a SIE file.

Example:

```
#VALUTA      NOK
```

#VER Verification item

Format:

```
#VER      series   verno   verdate    vertext   regdate    sign
```

Description:

1. The verification item is to always be followed by a number of #TRANS items within braces.
2. Vertext does not need to be specified.
3. Regdate is the date the verification was generated/registered. This date is primarily used within history processing. The specification of a regdate is optional.
4. Series is specified using letters from A onwards, alternatively using numbers from 1 onwards. An importing program is to handle both variants.
5. Series can also be specified using an alphanumeric string, for example LEV1.
6. Series and/or verno can be submitted empty where the file format is used to input transactions (using files of type 4I) from a pre-processing system to a financial reporting program. The series or verification number is in this case set by the financial reporting program.
7. All numbered verifications within a single series are to be placed in the SIE file in ascending verification number order.
8. Sign can be the name, signature or user id of the person or process that generated the transaction item or last edited the transaction item. Signature can be omitted.

Example:

```
#VER        "     "     20081216     "Postage"
{
            …
}
```

12. Summary of all item types

All item types are listed below in alphabetic sequence. The fields which are compulsory are shown in bold. Note however that it is not certain that the item is compulsory.

Label	*Field*
#ADRESS	**contact distribution address postal address tel**
#BKOD	**SNI code**
#DIM	**dimension no name**
#ENHET	**account no unit**
#FLAGGA	**x**
#FNAMN	**company name**
#FNR	**company id**
#FORMAT	**PC8**
#FTYP	**company type**
#GEN	**date** sign
#IB	**year no account balance** quantity
#KONTO	**account no account name**
#KPTYP	**chart of accounts type**
#KTYP	**account no account type**
#OBJEKT	**dimension no object no object name**
#OIB	**year no account {dimension no object no} balance** quantity
#OMFATTN	**date**
#ORGNR	**CIN** acq no act no
#OUB	year no account {dimension no object no} balance quantity
#PBUDGET	**year no. period account {**dimension no object no**} balance** quantity
#PROGRAM	**program name version**
#PROSA	**text**
#PSALDO	**year no. period account {**dimension no object no**} balance** quantity
#RAR	**year no start end**

Label	*Field*
#RES	**year no account balance** quantity
#SIETYP	**type no**
#SRU	**account SRU code**
#TAXAR	**year**
#TRANS	**account no. {**object list**} amount** transdate transtext quantity sign
#RTRANS	**account no. {**object list**} amount** transdate transtext quantity sign
#BTRANS	**account no. {**object list**} amount** transdate transtext quantity sign
#UB	**year no account balance** quantity
#UNDERDIM	**dimension no name superdimension**
#VALUTA	**Currency code**
#VER	series verno **verdate** vertext regdate sign