For Official Use **CTPA/CFA(2006)25/ADD3**

Organisation de Coopération et de Développement Economiques
Organisation for Economic Co-operation and Development **02-Dec-2005**

English - Or. English

CENTRE FOR TAX POLICY AND ADMINISTRATION
COMMITTEE ON FISCAL AFFAIRS

OECD MANUAL ON THE IMPLEMENTATION OF EXCHANGE OF INFORMATION FOR TAX PURPOSES - AUTOMATIC (OR ROUTINE) EXCHANGE OF INFORMATION

Module 3 - Automatic (or Routine) Exchange of Information

This document was approved by delegates to Working Party No.8.

The Manual is now submitted to the CFA for APPROVAL under the written procedure. Please submit your comments no later than January 9, 2006 to Ms. Suzanne Pedron (suzanne.pedron@oecd.org). If you can approve the Manual as it is you do not need to respond and the absence of comments will be taken to mean approval.

For further information please contact: Mrs. Martine Milliet-Einbinder: Tel: (33 1) 45 24 78 42; Fax: (33 1) 44 30 63 21; Email: martine.milliet-einbinder@oecd.org

JT00195476

CTPA/CFA(2006)25/ADD3
For Official Use

English - Or. English

TABLE OF CONTENTS

OECD MANUAL ON THE IMPLEMENTATION OF EXCHANGE OF INFORMATION FOR TAX PURPOSES

MODULE ON AUTOMATIC (OR ROUTINE) EXCHANGE OF INFORMATION

1. What is automatic exchange?

1. Automatic exchange of information (also called routine exchange by some countries) involves the systematic and periodic transmission of "bulk" taxpayer information by the source country to the residence country concerning various categories of income (e.g. dividends, interest, royalties, salaries, pensions, etc). This information is obtained on a routine basis in the source country (generally through reporting of the payments by the payer (financial institution, employer etc). Automatic exchange can also be used to transmit other useful types of information such as changes of residence, the purchase or disposition of immovable property, VAT refunds[1], etc. The country of residence tax authority can then check its tax records to verify that resident taxpayers have reported their foreign source income. In addition, information concerning the acquisition of significant assets may be used to evaluate the net worth of an individual, to see if the reported income reasonably supports the transaction. There are an increasing number of countries involved in automatic exchange using different types of media: tapes, diskettes, CD Roms but also paper.

2. Benefits of automatic exchange

2. The foreign source information received on magnetic media or in digital form can be input into the recipient tax data base (often using bridging programs to capture the relevant information) and automatically matched against the income reported by the taxpayer. This is the most cost effective way to process the information. For example the Australian Tax Office's 2004-05 Compliance Program states that 1171 foreign source income data matching audits were completed during the 2003-04 tax year, raising over AUD$3 million in liabilities**.** The foreign source information received on magnetic media or in digital form can also be matched manually, as a general procedure or when it could not be matched automatically. The automatic exchange of information on magnetic media also provides opportunities for more effective and efficient distribution of the exchanged information to local tax offices if needed and also for instance for feeding the information into data bases for purposes of risk analysis.

3. Legal basis

3. Automatic exchange can be based on:

i. The exchange of information article of the bilateral income tax convention between two countries; or

[1] These other types are not at present supported by the standard transmission formats (SMF/STF) developed by the OECD.

ii. Article 6 of the Joint Council of Europe/OECD Convention on Mutual Administrative Assistance in Tax Matters; or

iii. Article 3 of the EU Council Directive 77/799/EEC on Mutual Assistance as last amended or

iv. The EU Savings Directive 2003/48/EC; or

v. Article 17 of EU Council Regulation on administrative cooperation in the field of VAT 1798/2003; or

vi. Council Regulation of 16 November 2004 on administrative co-operation in the field of excise due; or

vii. Article 4, paragraph 3 of the CIAT Model Agreement on the Exchange of Tax Information.

4. Agreements or Memoranda of Understanding on automatic exchange

4. The exchange of information article of an income tax convention or automatic exchange article of a mutual assistance instrument constitutes the legal basis for automatic exchange of information. In addition, countries may agree to enter into a special working agreement or memorandum of understanding (MOU) setting forth the terms and conditions of the proposed automatic exchange. Such an agreement or MOU typically sets forth the types of information to be exchanged automatically, details about the procedures of sending and receiving the information, the appropriate format to use, and provision of TINs. These agreements or MOU may be published officially and may have a deterrent effect on potential tax evaders and are usually reviewed periodically.

5. The OECD has designed a Model Memorandum of Understanding between Competent Authorities on Automatic Exchange of Information for Tax Purposes C(2001)21/FINAL that can be used as a basis for an operational working agreement between tax administrations.

6. The OECD Model MOU provides a list of information that can be exchanged automatically, including:

- change in place of residence from one State to the other State;
- ownership of and income from immovable property;
- dividends;
- interest;
- royalties;
- capital gains;
- salaries, wages and other similar remuneration in respect of an employment,;
- directors' fees and other similar payments;
- income derived by artists and sportsmen, pensions and other similar remuneration, salaries, wages and other similar remuneration for government services, other income such as proceeds

from gambling, other items including items on indirect taxes such as VAT/sales tax and excise duties and social security payments; and

- commissions and other similar payments.

7. The OECD Model MOU recommends using the OECD Standard Magnetic Format for automatic exchange (or any further updated format recommended by the OECD Council) as well as providing Tax Identification Numbers (TINs) if available as they facilitate the processing and matching of the information received. It might also be a helpful reference when further inquiries to the other contracting party are necessary. In that respect the OECD Council recommended the use of TINs in the international context in 1997, see C(97)29/FINAL. The OECD Council recommends "that Member countries should encourage non residents recipients of income to disclose their residence country TIN. Member countries should consider making this disclosure mandatory." In the case of automatic exchange of information on income paid to non residents, having information on the residence country TIN would greatly facilitate the matching of information received by the residence country with the income reported by its own taxpayers. The table in the annex below gives an overview of the use of TINs for domestic purposes and international use of TINs.

5. Implementation

5.1 Standardisation of transmission formats and use of new media

8. Automatic exchange of information requires the standardisation of formats in order to be efficient. In 1981 the OECD designed a paper-based form for automatic exchange which introduced the standardisation of certain pieces of information C(81)39/FINAL. In 1992, taking advantage of technological developments, the OECD then designed the Standard Magnetic Format (SMF) for the transmission of taxpayer information on magnetic tape. Based on country experiences the SMF was revised in 1997 to further improve countries' capacity to match information received automatically with information reported by its taxpayers. Use of the revised format was recommended by the OECD Council in 1997 (see C(97)30/FINAL). The record layout of the Standard includes fields allocated to the:

- recipient beneficial owner, his agent or intermediary, to the actual payer of the income, the payer's agent or intermediary. For each series of fields the same pattern is followed to provide information on the TIN (both residence country TIN and source country TIN), name, alias or other name, date of birth (where applicable) and address; and

- income (tax year, date, type of payment, currency, gross and net amount, tax withheld, refund etc).

9. Fields are allocated to residence country TINs and source country TINs. The SMF is used by OECD member countries involved in automatic exchange and increasingly by non member countries. A multilingual electronic user manual is also available to provide guidance for the implementation of the Standard. It is available on CD ROM which can be obtained from the OECD Secretariat. In 2002 the European Union Council agreed on a standard format (FISC 39) for the implementation of the Savings Directive which is to be based on the OECD SMF. In 2008 EU countries will start using an STF-like format (FISC 73).

10. The OECD has also designed a "new generation" transmission format for automatic exchange to eventually replace the SMF. The new format is called the Standard Transmission Format (STF) and is

based on extensible mark-up language (XML[2]), a document mark-up language widely used in today's information technology for its many advantages (e.g. separation of the content of a message from any display structure, readability both by humans and machines, modularity and flexibility, ability to check the conformance of documents with the "contract" about its structure, etc). A multilingual electronic STF user manual is available to provide guidance for the implementation of the STF. It is available from the OECD Secretariat or on the OECD secure site for competent authorities. As the SMF and STF will coexist for the foreseeable future, bridging programmes have been developed to achieve conversion between the two formats, thus enabling treaty partners to engage in bilateral automatic exchange notwithstanding that they might each use a different standard format.

5.2 Security: Encryption and alternative methods

11. It is desirable that information contained in magnetic media exchanged automatically be transmitted in a secure manner and be encrypted whenever feasible. Such information is now transmitted on diskettes or, in particular, CD ROMs, both of which can be easily encrypted by the sending country using encryption software. A pilot has tested encryption software and a few countries now provide information on encrypted CD ROMs. The tested encryption software GNU PG[3] or equivalent commercial software has been found adequate to ensure the security of automatic exchange. If not encrypted, the diskettes tapes or CD ROMs should be exchanged via a secure mail system, such as a diplomatic pouch.

6. Importance of feedback from receiving country

12. Feedback to the sending country is essential to improve the efficiency of automatic exchange of information. Feedback from the receiving country on information exchanged automatically (not purely from an IT perspective) is crucial to make better use of what is exchanged: knowing what the source is of data exchanged, the common errors identified, etc. Feedback may also be useful to tax administrations for justifying resources for exchange of information. More generally, a systematic survey of the use of the OECD standard magnetic format is carried out on a regular basis by the OECD. Feedback includes comments on the accessibility, accuracy, and completeness of the data received as well as comments on the usefulness of the data, such as summary results of taxpayer reviews and audits.

7. Future developments: Risk analysis tools

13. Countries are now starting to use foreign source information received automatically not only for matching purposes but also in a more strategic manner for risk assessment purposes, and in particular to select cases for more in depth tax examinations.

[2] XML: a technical language for describing documents containing structured information. The term "extensible" refers to a system that can be enlarged by addition rather than by complete replacement.

[3] GNU Privacy Guard is a suite of programmes developed by the Free Software Foundation that provides security solutions for protecting and encrypting data (see www.gnupgp.org). It uses the Pretty Good Privacy (PGP) standard and is compatible with commercial PGP products. PGP is a widely used encryption programme designed to provide high-security encoding algorithms.

ANNEXES (ALSO AVAILABLE ON THE SECURE WEBSITE FOR EXCHANGE OF INFORMATION STAFF)

OECD Recommendation for Memorandum of Understanding on automatic exchange (see www.oecd.org/taxation)

OECD Standard Magnetic Format (SMF)

User Guide for OECD Standard Magnetic Format

OECD Standard Transmission Format (STF)

User Guide for Standard Transmission Format

Bridging programme from SMF to STF

Bridging programme from STF to SMF

Reference Guide on Sources of Information from Abroad (see www.oecd.org/taxation)

1997 REVISED OECD STANDARD MAGNETIC FORMAT FOR AUTOMATIC EXCHANGE OF INFORMATION

Technical introduction

The revised OECD Standard Magnetic Format is an improved version of the 1992 Standard Magnetic Format.
The revised OECD Standard Magnetic Format includes the Standard for the:

1 Record lay out and two tables
2 An OECD Foreign Data Exchange Magnetic Media Specifications
3 Physical label attached to the magnetic media

1. Record Lay out

The international standards (ISO codes) established by the International Organization for Standardization are used as codes for country names, currencies, dates , definition of characters used, file structure and labelling of the media. Two tables indicate the code for type of recipient and type of payer as well as the code for income.

The Record lay out includes 104 fields with filler fields at the end:

Fields 2 to 40 are allocated to the recipient Beneficial Owner,
Fields 41 to 55 are allocated to the recipient beneficial owner Agent or Intermediary
Fields 56 to 71 are allocated to the actual payer of the income
Fields 72 to 86 are allocated to the payer's agent or intermediary

For each series of fields the same pattern is followed to provide information on the Tax Identification Number, name, alias or other name and address.
Fields 87 to 100 are allocated to the income (tax year, date, type of payment, currency, gross and net amount, tax withheld, refund etc.).
Fields 101 and 102 are allocated to references
Fields 103 and 104 are allocated to fillers to provide additional information

For each field are provided the numbering of the field, the starting position of the field, its length, the data type: N= numeric, AN=alphanumeric, B=blank.

A user manual will be provided to give general guidance and country specifications.

Revised OECD Standard Magnetic Format Lay out

	Field Name	Start	Length	Data Type	Remarks
1	Data Type Indicator	1	1	N	0 - Repeat Data 1 - New Data 2 - Correction
2	Recipient Beneficial Owner Residence Country Code	2	2	A	ISO 3166 2 Alpha or Blank
3	Recipient Beneficial Owner Residence Country TIN	4	20	AN	Residence Country Tax Identification Number or Blank if data not available
4	Recipient Beneficial Owner Originating Country Code	24	2	A	ISO 3166 2 Alpha or Blank
5	Recipient Beneficial Owner Originating Country TIN	26	20	AN	Originating Country Tax Identification Number or Blank if data not available
6	Recipient Beneficial Owner OECD Recipient Type	46	2	AN	See Table 1.
7	Recipient Beneficial Owner Date of Birth	48	8	AN	ISO 8601: CCYYMMDD or CCYYMM or CCYY or Blank if unknown
8	Recipient Beneficial Owner Name Format Type	56	1	N	0 - Fixed 1 - Free If Free, all available name details will be given as a left justified string in bytes 57 through 266.
9	Recipient Beneficial Owner Keyname	57	70	AN	Keyname is the family name for individuals and the business name for legal entities.
10	Recipient	127	70	AN	Other Names includes First

	Beneficial Owner Other Names				Name, Middle Name and/or initials.
11	Recipient Beneficial Owner Title	197	35	AN	Title includes: Mr, Mrs, Doctor, Sir, Professor etc.
12	Recipient Beneficial Owner Suffix	232	35	AN	Suffix includes: Esquire, Senior, Junior etc.
13	Recipient Beneficial Owner Gender	267	1	AN	F - Female M - Male N - Non Individual U - Unknown
14	Recipient Beneficial Owner Birth City	268	35	AN	
15	Recipient Beneficial Owner Birth City Sub Entity	303	35	AN	
16	Recipient Beneficial Owner Birth Country Code	338	2	A	ISO 3166 2 Alpha or Blank
17	Recipient Beneficial Owner Alias or Other Name Format Type	340	1	N	0 - Fixed 1 - Free If Free, all available name details will be given as a left justified string in bytes 341 through 550
18	Recipient Beneficial Owner Alias or Other Keyname	341	70	AN	Keyname is the family name for individuals and the business name for legal entities.
19	Recipient Beneficial Owner Alias or Other Other Names	411	70	AN	Other Names includes First Name, Middle Name and/or initials.

20	Recipient Beneficial Owner Alias or Other Title	481	35	AN	Title includes: Mr, Mrs, Doctor, Sir, Professor etc.
21	Recipient Beneficial Owner Alias or Other Suffix	516	35	AN	Suffix includes: Esquire, Senior, Junior etc.
22	Recipient Beneficial Owner In Care of Name Format Type	551	1	N	0 - Fixed 1 - Free If Free, all available name details will be given as a left justified string in bytes 552 through 761. .
23	Recipient Beneficial Owner In Care Of Keyname	552	70	AN	Keyname is the family name for individuals and the business name for legal entities.
24	Recipient Beneficial Owner In Care of Other Names	622	70	AN	Other Names includes First Name, Middle Name and/or initials.
25	Recipient Beneficial Owner In Care of Title	692	35	AN	Title includes: Mr, Mrs, Doctor, Sir, Professor etc.
26	Recipient Beneficial Owner In Care of Suffix	727	35	AN	Suffix includes: Esquire, Senior, Junior etc.
27	Recipient Beneficial Owner Address Type	762	1	N	0 - Residential or Business 1 - Registered Office 2 - Other or Unknown
28	Recipient Beneficial Owner Address Format Type	763	1	N	0 - Fixed 1 - Free If Free, all available address details will be given as a left justified string in bytes 764 through 912

29	Recipient Beneficial Owner Address Street	764	70	AN	
30	Recipient Beneficial Owner Address City	834	35	AN	
31	Recipient Beneficial Owner Address Country Sub-entity	869	35	AN	State, Province etc.
32	Recipient Beneficial Owner Address Postal Code	904	9	AN	
33	Recipient Beneficial Owner Address Country Code	913	2	A	ISO 3166 2 Alpha or Blank
34	Recipient Beneficial Owner Other Address Type	915	1	N	0 - Residential or Business 1 - Registered Office 2 - Other or Unknown
35	Recipient Beneficial Owner Other Address Format Type	916	1	N	0 - Fixed 1 - Free If Free, all available address details will be given as a left justified string in bytes 917 through 1065
36	Recipient Beneficial Owner Other Address Street	917	70	AN	
37	Recipient Beneficial Owner Other Address City	987	35	AN	
38	Recipient Beneficial Owner Other Address Country Sub-entity	1022	35	AN	State, Province etc.

39	Recipient Beneficial Owner Other Address Postal Code	1057	9	AN	
40	Recipient Beneficial Owner Other Address Country Code	1066	2	AN	ISO 3166 2 Alpha or Blank
41	Recipient Agent or Intermediary Country Code for TIN 1	1068	2	A	ISO 3166 2 Alpha or Blank
42	Recipient Agent or Intermediary TIN 1	1070	20	AN	Recipient Agent or Intermediary Tax Identification Number or Blank if data not available
43	Recipient Agent or Intermediary Country Code for TIN 2	1090	2	A	ISO 3166 2 Alpha or Blank
44	Recipient Agent or Intermediary TIN 2	1092	20	AN	Recipient Agent or Intermediary Tax Identification Number or Blank if data not available
45	Recipient Agent or Intermediary Name Format Type	1112	1	N	0 - Fixed 1 - Free If Free, all available name details will be given as a left justified string in bytes 1113 through 1322
46	Recipient Agent or Intermediary Keyname	1113	70	AN	Keyname is the family name for individuals and the business name for legal entities.
47	Recipient Agent or Intermediary Other Names	1183	70	AN	Other Names includes First Name, Middle Name and/or initials.
48	Recipient Agent or Intermediary Title	1253	35	AN	Title includes: Mr, Mrs, Doctor, Sir, Professor etc.
49	Recipient	1288	35	AN	Suffix includes:

	Agent or Intermediary Suffix				Esquire, Senior, Junior etc.
50	Recipient Agent or Intermediary Address Format Type	1323	1	N	0 - Fixed 1 - Free If Free, all available address details will be given as a left justified string in bytes 1324 through 1472
51	Recipient Agent or Intermediary Address Street	1324	70	AN	
52	Recipient Agent or Intermediary Address City	1394	35	AN	
53	Recipient Agent or Intermediary Address Country Sub-entity	1429	35	AN	State, Province etc.
54	Recipient Agent or Intermediary Address Postal Code	1464	9	AN	
55	Recipient Agent or Intermediary Address Country Code	1473	2	AN	ISO 3166 2 Alpha or Blank
56	Actual Payer Country Code for TIN 1	1475	2	A	ISO 3166 2 Alpha or Blank

57	Actual Payer TIN 1	1477	20	AN	Actual Payer Agent or Intermediary Tax Identification Number or Blank if data not available
58	Actual Payer Country Code for TIN 2	1497	2	A	ISO 3166 2 Alpha or Blank
59	Actual Payer TIN 2	1499	20	AN	Actual Payer Agent or Intermediary Tax Identification Number or Blank if data not available
60	Actual Payer OECD Payer Code	1519	2	AN	See Table 1
61	Actual Payer Name Format Type	1521	1	N	0 - Fixed 1 - Free If Free, all available address details will be given as a left justified string in bytes 1521 through 1731
62	Actual Payer Keyname	1522	70	AN	Keyname is the family name for individuals and the business name for legal entities.
63	Actual Payer Other Names	1592	70	AN	Other Names includes First Name, Middle Name and/or initials.
64	Actual Payer Title	1662	35	AN	Title includes: Mr, Mrs, Doctor, Sir, Professor etc.
65	Actual Payer Suffix	1697	35	AN	Suffix includes: Esquire , Senior, Junior etc.
66	Actual Payer Address Format Type	1732	1	N	0 - Fixed 1 - Free If Free, all available address details will be given as a left justified string in bytes 1733 through 1881

67	Actual Payer Address Street	1733	70	AN	
68	Actual Payer Address City	1803	35	AN	
69	Actual Payer Address Country Sub entity	1838	35	AN	State, Province etc.
70	Actual Payer Address Postal Code	1873	9	AN	
71	Actual Payer Address Country Code	1882	2	A	ISO 3166 2 Alpha or Blank
72	Payer Agent or Intermediary Country Code for TIN 1	1884	2	A	ISO 3166 2 Alpha or Blank
73	Payer Agent or Intermediary TIN 1	1886	20	AN	Agent or Intermediary Tax Identification Number or Blank if data not available
74	Payer Agent or Intermediary Country Code for TIN 2	1906	2	A	ISO 3166 2 Alpha or Blank
75	Payer Agent or Intermediary TIN 2	1908	20	AN	Agent or Intermediary Tax Identification Number or Blank if data not available
76	Payer Agent or Intermediary Name Format Type	1928	1	N	0 - Fixed 1 - Free If Free, all available address details will be given as a left justified string in bytes 1929 through 2138
77	Payer Agent or Intermediary Keyname	1929	70	AN	Keyname is the family name for individuals and the business name for legal entities.
78	Payer Agent or Intermediary Other Names	1999	70	AN	Other Names includes First Name, Middle Name and/or initials.
79	Payer	2069	35	AN	Title includes:

	Agent or Intermediary Title				Mr, Mrs, Doctor, Sir, Professor etc.
80	Payer Agent or Intermediary Suffix	2104	35	AN	Suffix includes: Esquire, Senior, Junior etc.
81	Payer Agent or Intermediary Address Format Type	2139	1	N	0 - Fixed 1 - Free If Free, all available name details will be given as a left justified string in bytes 2140 through 2289
82	Payer Agent or Intermediary Address Street	2140	70	AN	
83	Payer Agent or Intermediary Address City	2210	35	AN	
84	Payer Agent or Intermediary Address Country Sub entity	2245	35	AN	State, Province etc.
85	Payer Agent or Intermediary Address Postal Code	2380	9	AN	
86	Payer Agent or Intermediary Address Country Code	2289	2	A	ISO 3166 2 Alpha or Blank

87	Sending Country Tax Year End	2291	8	AN	ISO 8601 CCYYMMDD or CCYYMM or CCYY or Blank if unknown
88	Date of Payment	2299	8	AN	ISO 8601 CCYYMMDD or CCYYMM or CCYY or Blank if unknown
89	OECD Payment Type	2307	4	AN	See Table 2
90	Sending Country Payment Type	2311	4	AN	
91	Gross Income Paid Currency Code	2315	3	A	ISO 4217 3 Alpha
92	Gross Income Paid Amount Major Unit of Currency	2318	18	N	Major units of currency only
93	Net Income Paid Currency Code	2336	3	A	ISO 4217 3 Alpha
94	Net Income Paid Amount Major Unit of Currency	2339	18	N	Major units of currency only
95	Tax Withheld Currency Code	2357	3	A	ISO 4217 3 Alpha
96	Tax Withheld Amount Major Unit of Currency	2360	18	N	Major units of currency only
97	Tax Rate	2378	4	N	nn.nn decimal point virtual Blank if no data
98	Tax Refund Currency Code	2382	3	A	ISO 4217 3 Alpha

99	Tax Refund Amount Major Unit of Currency	2385	18	N	Major units of currency only
100	Date of Refund or Date of First Refund	2403	8	AN	ISO 8601 CCYYMMDD or CCYYMM or CCYY or Blank if unknown
101	Sender's Reference	2411	70	AN	Sender's number for any queries relating to the particular record
102	Correction Reference	2481	70	AN	Relates to the Sender's number where this record is a correction record. (see field 1). Otherwise Blank
103	Filler General Use	2551	105	AN	Blank unless used. Details to be provided in technical notes accompanying the transmittal.
104	Filler Specific Arrangements	2656	105	AN	Blank unless used. Details agreed between sending country and recipient country and specified in technical notes accompanying the transmittal.

Table 1: Type of recipient and type of payer

01 Individual
02 Corporation
03 Partnership
04 Business organisation other than corporation and partnership
05 Government or international organisation
06 Other (specify in filler)
07 Unknown

Table 2: Type of income code corresponding to the numbering of the Article of the OECD Model Convention on Income and Capital

6-- Income from immovable property
7-- Business profits
10--Dividends
11-- Interest
12-- Royalties
13-- Capital Gains
14 --Income from Independent personal services
15-- Income from dependent personal services
15a--Gross amount (including fringe benefits)
15b--Money amount only but additional information on fringe benefits to follow in fillers according to bilateral arrangements
15c--Money amount only
16-- Directors' fees
17 --Income derived from activities of an artist or sportsman
18-- Pensions
19-- Income from government services and public pensions
20-- Payments to students for education and training
21-- Other income

2. Standard for an OECD Foreign Data Exchange Magnetic Media Specifications

<table>
<tr><td colspan="3">ORGANISATION FOR ECONOMIC CO-OPERATION AND DEVELOPMENT

Foreign Tax Data Exchange
Magnetic Media Specifications</td></tr>
<tr><td rowspan="3">Identity Information</td><td>Sending Country Name</td><td></td></tr>
<tr><td>Receiving Country Name</td><td></td></tr>
<tr><td>Sending Country Contact Person’s Name/ Phone Number</td><td></td></tr>
<tr><td rowspan="7">Media Information</td><td>Total Number of Media being sent</td><td></td></tr>
<tr><td>File Name(s)
File Name Format: xxyyzznn
xx= Sending Country
(Code ISO 3166 2 Alpha)
yy= Receiving Country
(Code ISO 3166 2 Alpha)
zz= Year of Sending
nn= Sequence Number of File
(nn-th File from xx to yy in zz)</td><td></td></tr>
<tr><td>Record Length</td><td></td></tr>
<tr><td>Block Size</td><td></td></tr>
<tr><td>Density</td><td></td></tr>
<tr><td>Creation Date</td><td></td></tr>
<tr><td>Tax Year(s) of Data</td><td></td></tr>
<tr><td colspan="3">Comments_____
_____
_____</td></tr>
<tr><td>Signature of Competent Authority</td><td>Title</td><td>Date</td></tr>
</table>

3. Standard for physical label attached to the magnetic media

ORGANISATION FOR ECONOMIC CO-OPERATION AND DEVELOPMENT

Foreign Tax Data Exchange

Physical Label Attached to Magnetic Media

Foreign Tax Data Exchanged in OECD Standard Magnetic Format

Sending Country Name:	**Filename:**
Receiving Country Name:	**Tax Year(s) of Data:**
Record Length:	**Block size:**
Number of Records on Media:	**Density:**
Sequence Number of Media in this Transmission (n of m: n/m):	

N.B.: The Tax Data provided are to be used according to the provisions of the international instrument under which they are exchanged.

USER GUIDE FOR OECD STANDARD MAGNETIC FORMAT (SMF)

English Edition (Both Record Layout Tables and Explanations in English)

The revised OECD Standard Magnetic Format (SMF 1997) is an improved version of the 1992 Standard Magnetic Format. The revised OECD Standard Magnetic Format includes the standards for (1) the Record Layout and two codetables, (2) OECD Foreign Data Exchange Magnetic Media Specifications - Transmittal Document -, (3) Physical label to be attached to the magnetic media.

Record Layout Version 1997

The international standards established by the International Organization for Standardization (ISO codes) are used as codes for country names, currencies, dates, definition of characters used, file structure and labelling of the media. Two tables indicate the code for type of recipient and type of payer as well as the code for income. The record layout includes 104 fields, including filler fields at the end: fields 2 to 40 are allocated to the recipient beneficial owner, fields 41 to 55 are allocated to the recipient beneficial owner's agent or intermediary, fields 56 to 71 are allocated to the actual payer of the income, fields 72 to 86 are allocated to the payer's agent or intermediary. For each series of fields the same pattern is followed to provide information on the Tax Identification Number (TIN), name, alias or other name and address. Fields 87 to 100 are allocated to the income (tax year, date, type of payment, currency, gross and net amount, tax withheld, refund etc.). Fields 101 and 102 are allocated to record references. Fields 103 and 104 are allocated to fillers to provide additional information. For each field are provided: (1) the number of the field, (2) the starting position of the field, (3) its length, (4) the data type: A=alphabetic, N= numeric, AN=alphanumeric, B=blank. Entries to all fields marked as alphabetic or alphanumeric are to be left-justified. Entries to fields marked as numeric are to be right-justified. If the content of a field does not fill the entire space provided for this field, the rest of the field shall be filled with blanks; in the case of a numeric field it is also acceptable to fill with leading zeros.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F001	***Document Type Indicator	1	1	n	"0" repeated "1" new "2" correction

This field is used to indicate whether the following data are new, a replication of data already sent (possibly not received) or corrections to data sent. The field applies only to the one record in which it is included. In the case of repeated or corrected data field #102 must contain the identifier (field #101) of the record it refers to. Whenever the reference to a replicated record is not found by the receiving country it will be assumed to be

new data. If the reference to a correcting record is not found, the correction is treated as replication. Records shall be transmitted and, what is even more important, processed in the following order: repeated - new - correction. In the case of a correction also the unchanged fields shall be transmitted again (i.e., repeated) - except for field #101.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
RBO	***Recipient Beneficial Owner				

Fields #2 to #40 contain data about the Recipient Beneficial Owner. The recipient beneficial owner is the person (legal person or individual), resident of a contracting State, that is entitled to the income for tax purposes and has the benefit thereof, taking into account the economic, legal, factual, and other relevant circumstances (e.g. the relevant double taxation treaty) under which the income is received.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F002	**ResidenceCountryCode	2	2	a	ISO 3166 ; blank if unknown

Enter the appropriate country code for the state of which the recipient beneficial owner is a resident from the ISO 3166 two-byte alpha version. Leave blank if no data are available. Most likely, however, the "Unknown" case will not happen in a transmission as you would not know where to transmit the data.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F003	**ResidenceCountryTIN	4	20	an	Residence Country Tax Identification Number or blank if data not available

Enter the Tax Identification Number (TIN) of the recipient beneficial owner in the state of which he is a resident for treaty purposes. Leave blank if no data are available. If in the residence country a TIN properly said does not exist but another identifier is used regularly in that country for tax purposes it can be entered here instead of a TIN. Other (general) identifiers can be helpful for matching; they shall, however, be entered in field #104 together with an explanation of their type. The country specific information of the SMF documentation or the technical note accompanying the transmission also may contain further advice.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F004	**OriginatingCountryCode	24	2	a	ISO 3166 ; blank if unknown

Enter the appropriate country code for the state out of which the income is paid from the ISO 3166 two-byte alpha version. Leave blank if no data are available. As a rule, however, this will be the code for the country transmitting the information, so the "Unknown" case is unlikely.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F005	**OriginatingCountryTIN	26	20	an	Originating Country Tax Identification Number or blank if data not available

Enter the Tax Identification Number (TIN) of the recipient beneficial owner assigned by the state out of which the income is paid. This instruction applies in the case of countries - like the United States - that also assign TINs to non-residents. The type of such number might be explained in field #104. The country-specific explanation of the SMF documentation also may contain further information which may help the treaty partner decide, whether it can make use of such "TIN". Leave blank if no data are available.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F006	**OECDRecipientType	46	2	an	See Codelist Table 1

Enter the appropriate type of the recipient from codelist #1.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F007	**DateOfBirth	48	8	an	ISO 8601: CCYYMMDD or CCYYMM or CCYY or Blank if unknown

Enter the date of birth of the recipient beneficial owner as indicated in the ISO 8601 standard, i.e., CCYYMMDD; or CCYYMM; or CCYY (C = Century; Y = Year; M = Month; D = Day). Leave blank if no data are available.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
N_RBO	**(name)				

The name of the recipient beneficial owner as indicated to the sending country by payer documentation or official records. In the fields #8 to #12, the user must enter data about the name of the recipient beneficial owner. Refer to those fields for more information. If the recipient beneficial owner is also known under another name ("alias"), insert that data in fields #17 to #21. Refer to those fields for more information.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F008	*NameFormatType	56	1	n	0 fixed; 1 free; If free, all available name details will be given as a left justified string in bytes 57 through 266

For fields #9 to #12, the user has the option to enter the data about the name of the recipient beneficial owner either as one long left-justified field of 210 bytes in positions 57 through 266, or to spread the data over 4 fields. If the user chooses the option to enter the data required in fields #9 to

#12 in one field, enter "1". This will be the common case with legal entities. All available name details should then be entered as one left-justified string of 210 bytes, blanks serving as delimiter between name parts. If the user chooses the option to enter the data required in fields #9 to #12 in separate fields, enter "0" (zero). The free format should be used if there is any doubt about the name details (which is keyname, which is first name etc.) as the receiving country may be in a better position to make a correct judgment. Note that "keyname" is intended to be a culturally neutral notion for what otherwise might be called "surname", "last name", "family name" etc.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FreeForm1	***(name free format)**	57	210	an	

If the user has entered "1" in field #8 he has opted to enter the data about the name of the recipient beneficial owner in one long left-justified field of 210 bytes. For individuals enter the family name, the first name, middle name or initials (if available), the title (if available, e.g. Mr., Mrs.), and/or the suffix (if available, e.g., Esquire, Senior). For legal entities (corporations, partnerships, etc.), enter the registered or documented business name. Use blank as a delimiter between parts of the name.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FixForm1	***(name fixed format)**				The details of the subdivision in fields are given below.

If the user has entered "0" (zero) in field #8, he has opted to enter the data about the name of the recipient beneficial owner in four separate fields.In fields #9 to #12, enter for individuals the family name (#9), the first name, middle name and/or initials (if available) (#10), the title (if available, e.g. Mr., Mrs.) (#11) and/or the suffix (if available, e.g. Esquire, Senior) (#12). Note that "keyname" is intended to be a culturally neutral notion for what otherwise might be called "surname", "last name", "family name" etc. For legal entities (corporations, partnerships, etc.) if you do not use free form as recommended, enter the complete business name in field #9.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F009	Keyname	57	70	an	Keyname is the family name for individuals and the business name for legal entities

If the user has entered "0" (zero) in field #8, he has opted to enter the data about the name of the recipient beneficial owner in four separate fields.In field #9 enter the family name for individuals or the business name for legal entities (if you do not use free form as recommended for legal entities).

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F010	OtherNames	127	70	an	Other name includes first name, middle name and/or initials

If the user has entered "0" (zero) in field #8, he has opted to enter the data about the name of the recipient beneficial owner in four separate fields.In field #10, enter the first name, the middle name and/or the initials. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F011	Title	197	35	an	Title includes: Mr, Mrs, Doctor, Sir, Professor etc.

If the user has entered "0" (zero) in field #8, he has opted to enter the data about the name of the recipient beneficial owner in four separate fields.In field #11, enter the title. Titles may include Mr; Mrs; Doctor, Sir, Professor, etc. Abbreviations may be used as customary. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F012	Suffix	232	35	an	Suffix includes: Esquire, Senior, Junior etc.

If the user has entered "0" (zero) in field #8, he has opted to enter the data about the name of the recipient beneficial owner in four separate fields.In field #12, enter the suffix. Suffixes may include Esquire, Senior, Junior, etc. Abbreviations may be used as customary. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F013	****Gender**	267	1	a	F - Female; M - Male; N - Non individual; U - Unknown

The Recipient Beneficial Owner of the income may be an individual, a company, a partnership, or other business organisation or a permanent establishment. For all cases except individuals enter "N". Note that this includes partnerships. Example: a partnership of two male partners will be denoted "N". Individuals are identified as "F" or "M" or "U" as the case may be. The transmitting country shall use upper case; however, the country receiving the information shall be able to process lower case as well as upper case.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F014	****BirthCity**	268	35	an	

Enter the name of the city where the recipient beneficial owner was born. This field may serve to better identify the beneficial owner. It applies only for natural persons.This field may serve to better identify the beneficial owner. It applies only for natural persons. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks

F015	****BirthCitySubentity**	303	35	an	

Enter the name of the sub-entity of the city where the recipient beneficial owner was born.This field may serve to better identify the beneficial owner. It applies only for natural persons. Leave blank if no data are available or in the case of a legal entity. This field will generally be used only in the case of large cities that are subdivided, e.g., Manhattan (New York), Kreuzberg (Berlin).

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F016	****BirthCountryCode**	338	2	a	ISO 3166 2 Alpha or Blank

Enter the appropriate country code for the state in which the city of birth of the recipient beneficial owner is situated from the ISO 3166 two-byte alpha version. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
ALI	****(AliasOrOther)**				

Sometimes the recipient beneficial owner is also known under another name (an "alias"). For legal entities (corporations, partnerships etc.) an alias might be a short name of the entity or a name that is used for public acquaintance instead of the official business name ("doing business as -DBA", "trading as - TA"; examples: in the United States, DaimlerChrysler is still known simply as Chrysler, Dr. William Black DAB Quality Pedriatics, Inc.). For such alias names enter available data in fields #17 to #21. Leave blank if no data are available (which will probably be the most common case). Instructions for these fields will more or less duplicate those for fields #8 to #12. However, fields that are numeric in the "name"-group are defined alphanumeric here to allow for a blank value when no Alias is known.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F017	***NameFormatType**	340	1	an	0 fixed; 1 free; Blank if no alias given. If free, all available name details will be given as a left justified string in bytes 341 through 550.

For fields #18 through #21, the user has the option to enter the data about the "alias" name of the recipient beneficial owner either as one long left-justified field of 210 bytes in positions 341 through 550, or to spread the data over 4 fields. If the user chooses the option to enter the data required in fields #18 through #21 in one field, enter "1". This will be the common case with legal entities. All available name details should then be entered as one left-justified string of 210 bytes, blanks serving as delimiter between name parts. If the user chooses the option to enter the data required in fields #18 through #21 in separate fields, enter "0" (zero). The free format should be used if there is any doubt about the name details (which is keyname, which is first name etc.) as the receiving country may be in a better position to make a correct judgment. Note that "keyname" is intended to be a culturally neutral notion for what otherwise might be called "surname", "last name", "family name" etc. To allow for the whole "alias" group

of fields to be left blank if no such information is known, this field is defined alphanumeric in contrast to the "name" group of fields and a blank is a valid entry in the case mentioned.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FreeForm2	***(name free format)**	341	210	an	

If the user has entered "1" in field #17 he has opted to enter the data about the alias of the recipient beneficial owner in one long left-justified field of 210 bytes. For individuals enter the family name, the first name, middle name or initials (if available), the title (if available, e.g. Mr., Mrs.), and/or the suffix (if available, e.g., Esquire, Senior). For legal entities (corporations, partnerships, etc.), enter the registered or documented business name. Use blank as a delimiter between parts of the name.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FixForm2	***(name fixed format)**				The details of the subdivision in fields are given below.

If the user has entered "0" (zero) in field #17, he has opted to enter the data about the alias of the recipient beneficial owner in four separate fields.In fields #18 to #21, enter for individuals the family name (#18), the first name, middle name and/or initials (if available) (#19), the title (if available, e.g. Mr., Mrs.) (#20) and/or the suffix (if available, e.g. Esquire, Senior) (#21). Note that "keyname" is intended to be a culturally neutral notion for what otherwise might be called "surname", "last name", "family name" etc. For legal entities (corporations, partnerships, etc.) if you do not use free form as recommended, enter the complete business name in field #18.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F018	Keyname	341	70	an	Keyname is the family name for individuals and the business name for legal entities

If the user has entered "0" (zero) in field #17, he has opted to enter the data about the alias of the recipient beneficial owner in four separate fields.In field #18 enter the family name for individuals or the business name for legal entities (if you do not use free form as recommended for legal entities).

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F019	OtherNames	411	70	an	Other name includes first name, middle name and/or initials

If the user has entered "0" (zero) in field #17, he has opted to enter the data about the alias of the recipient beneficial owner in four separate fields.In field #19, enter the first name, the middle name and/or the initials. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F020	Title	481	35	an	Title includes: Mr, Mrs, Doctor, Sir, Professor etc.

If the user has entered "0" (zero) in field #17, he has opted to enter the data about the alias of the recipient beneficial owner in four separate fields.In field #20, enter the title. Titles may include Mr; Mrs; Doctor, Sir, Professor, etc. Abbreviations may be used as customary. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F021	Suffix	516	35	an	Suffix includes: Esquire, Senior, Junior etc.

If the user has entered "0" (zero) in field #17, he has opted to enter the data about the alias of the recipient beneficial owner in four separate fields.In field #21, enter the suffix. Suffixes may include Esquire, Senior, Junior, etc. Abbreviations may be used as customary. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
ICO	****(InCareOfPerson)**				

A person that is acting for the recipient beneficial owner in some capacity (e.g., legal, administrative, or functional). This can be something as simple as the use of another person's post office box for the receipt of mail and does not necessarily imply the creation of any legal obligation or responsibility.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F022	***NameFormatType**	551	1	an	0 fixed; 1 free; Blank if no in-care-of given. If free, all available name details will be given as a left justified string in bytes 552 through 761.

For fields #23 to #26, the user has the option to enter the data about the name of the in-care-of person either as one long left-justified field of 210 bytes in positions 552 through 761, or to spread the data over 4 fields. If the user chooses the option to enter the data required in fields #23 to #26 in one field, enter "1". This will be the common case with legal entities. All available name details should then be entered as one left-justified string of 210 bytes, blanks serving as delimiter between name parts. If the user chooses the option to enter the data required in fields #23 to #26 in separate fields, enter "0" (zero). The free format should be used if there is any doubt about the name details (which is keyname, which is first name etc.) as the receiving country may be in a better position to make a correct judgment. Note that "keyname" is intended to be a culturally neutral notion for what otherwise might be called "surname", "last name", "family name" etc. To allow for the whole "in-care-of" group of fields to be left blank if no

such information is known, this field is defined alphanumeric in contrast to the "name" group of fields and a blank is a valid entry in the case mentioned.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FreeForm3	***(name free format)**	552	210	an	

If the user has entered "1" in field #22 he has opted to enter the data about the in-care-of person in one long left-justified field of 210 bytes. For individuals enter the family name, the first name, middle name or initials (if available), the title (if available, e.g. Mr., Mrs.), and/or the suffix (if available, e.g., Esquire, Senior). For legal entities (corporations, partnerships, etc.), enter the registered or documented business name. Use blank as a delimiter between parts of the name.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FixForm3	***(name fixed format)**				The details of the subdivision in fields are given below.

If the user has entered "0" (zero) in field #22, he has opted to enter the data about the in-care-of person in four separate fields.In fields #23 to #26, enter for individuals the family name (#23), the first name, middle name and/or initials (if available) (#24), the title (if available, e.g. Mr., Mrs.) (#25) and/or the suffix (if available, e.g. Esquire, Senior) (#26). Note that "keyname" is intended to be a culturally neutral notion for what otherwise might be called "surname", "last name", "family name" etc. For legal entities (corporations, partnerships, etc.) if you do not use free form as recommended, enter the complete business name in field #23.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F023	Keyname	552	70	an	Keyname is the family name for individuals and the business name for legal entities

If the user has entered "0" (zero) in field #22, he has opted to enter the data about the in-care-of person in four separate fields.In field #23 enter the family name for individuals or the business name for legal entities (if you do not use free form as recommended for legal entities).

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F024	OtherNames	622	70	an	Other name includes first name, middle name and/or initials

If the user has entered "0" (zero) in field #22, he has opted to enter the data about the in-care-of person in four separate fields.In field #24, enter the first name, the middle name and/or the initials. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F025	Title	692	35	an	Title includes: Mr, Mrs, Doctor, Sir, Professor etc.

If the user has entered "0" (zero) in field #22, he has opted to enter the data about the in-care-of person in four separate fields.In field #25, enter the title. Titles may include Mr; Mrs; Doctor, Sir, Professor, etc. Abbreviations may be used as customary. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F026	Suffix	727	35	an	Suffix includes: Esquire, Senior, Junior etc.

If the user has entered "0" (zero) in field #22, he has opted to enter the data about the in-care-of person in four separate fields.In field #26, enter the suffix. Suffixes may include Esquire, Senior, Junior, etc. Abbreviations may be used as customary. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
A_RBO	****(Address)**				

Fields #27 to #33 contain the primary or usual address of the recipient beneficial owner as documented to the paying entity or sending administration. The main aim of including this address is for identification of the recipient beneficial owner and to give the means of sending mail. The location of this address does not necessarily imply any tax relevance. The address can basically be situated in any country. The sending country may also inform the receiving country of another address it has identified in the next field group (fields #34 through #40).

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F027	***AddressType**	762	1	n	0 - Residential or Business; 1 - Registered Office; 2 -Other or Unknown

Enter the appropriate type of the address of the recipient beneficial owner by selecting from the given possibilities: 0 - Residential or Business; 1 - Registered Office; 2 -Other or Unknown. Due to the fact that the address usually relates to a resident of another state, the sending state may not be certain of the type of the address. Therefore, when in doubt, the user should enter "2".

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F028	***AddrFormatType**	763	1	n	0 fixed; 1 free; If free, all available address details will be given as a left justified string in bytes 764 through 912

For fields #29 through #32, the user has the option to enter the data about the address of the recipient beneficial owner either as one long field or to spread the data over four fields. If the user chooses the option to enter the data required in fields #29 through #32 in separate fields, enter "0" (zero). If the user chooses the option to enter the data required in a less structured way enter "1". All available details on the address of the recipient beneficial owner should then be presented as one left-justified string of 149 bytes, blank or "/" (slash) used as a delimiter between parts of the address. PLEASE NOTE that the address country code is outside this area of choosable format. The use of the fixed form is recommended as a rule to allow easy matching. However, the use of the free form is recommended if the sending state cannot reliably identify and distinguish the different parts of the adress.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FreeForm4	***(address free format)**	764	149	an	

If the user has entered "1" in field #28, he has opted to enter the data about the address of the recipient beneficial owner in one long left-justified field of 149 bytes. Enter the street and number, the city, the country sub-entity - if applicable (state, province or other political subunit) - and the postal (zip) code. Use blank as a delimiter between parts of the address. You may also use "/" (slash) as a delimiter to designate a new line in the printed form of the address. Do not enter the address country code in this field.It is strongly recommended that the address information be presented in the way the state where the recipient beneficial owner resides or is located usually structures the address. Refer to the structure information to which the link "ad-strct" points - depending on implementation.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FixForm4	***(address fixed format)**				The details of the subdivision in fields are given below.

If the user has entered "0" (zero) in field #28, he has opted to enter the data about the address of the recipient beneficial owner in four separate fields (#29 through #32).

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F029	Street	764	70	an	

If the user has entered "0" (zero) in field #28, he has opted to enter the data about the address of the recipient beneficial owner in four separate fields (#29 through #32). In field #29, enter the complete name of the street and the building number where the recipient beneficial owner resides or

is situated.It is strongly recommended that the address information be presented in the way the state where the recipient beneficial owner resides or is located usually structures the address. Refer to the structure information to which the link "ad-strct" points - depending on implementation.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F030	City	834	35	an	

If the user has entered "0" (zero) in field #28, he has opted to enter the data about the address of the recipient beneficial owner in four separate fields (#29 through #32). In field #30, enter the name of the city or municipality where the recipient beneficial owner resides or is situated.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F031	CountrySubentity	869	35	an	State, Province etc

If the user has entered "0" (zero) in field #28, he has opted to enter the data about the address of the recipient beneficial owner in four separate fields (#29 through #32). In field #31, enter the name of the country sub-entity if applicable in which the city where the recipient beneficial owner resides or is situated. Sub-entities may include states, regions, laender, provinces, or other political subunits. Leave blank if no data are available.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F032	PostalCode	904	9	an	

If the user has entered "0" (zero) in field #28, he has opted to enter the data about the address of the recipient beneficial owner in four separate fields (#29 through #32). In field #32, enter the postal (zip) code of the address of the recipient beneficial owner.It is strongly recommended that the address information be presented in the way the state where the recipient beneficial owner resides or is located usually structures the address. Refer to the structure information to which the link "ad-strct" points - depending on implementation.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F033	***CountryCode**	913	2	a	ISO 3166 ; blank if unknown

Enter the ISO 3166 two-byte alpha country code of the address entered in bytes 764 through 912. This field is not included in the alternatively fixed or free area in order to provide a unique location for this information.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
OA_RBO	****(OtherAddress)**				

This field group (fields #35 through #39) offers the opportunity to inform the information receiving country of a second address for the beneficial owner of which the information sending country has knowledge. The structure of this field group is identical to the primary address field group.

This field group may be left empty (blank), if no other address is known. Therefore some fields that are numeric only in the primary address field group are defined alphanumeric here to allow for a blank value.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F034	***OtherAddressType**	915	1	an	0 - Residential or Business; 1 - Registered Office; 2 -Other or Unknown; blank is allowed if the whole field group is empty.

Enter the appropriate type of the address of the recipient beneficial owner by selecting from the given possibilities: 0 - Residential or Business; 1 - Registered Office; 2 -Other or Unknown. Due to the fact that the address usually relates to a resident of another state, the sending state may not be certain of the type of the address. Therefore, when in doubt, the user should enter "2".

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F035	***OtherAddrFormatType**	916	1	an	0 fixed; 1 free; blank if no other address given. If free, all available address details will be given as a left justified string in bytes 917 through 1065

For fields #36 through #39, the user has the option to enter the data about the other address of the recipient beneficial owner either as one long field or to spread the data over four fields. If the user chooses the option to enter the data required in fields #36 through #39 in separate fields, enter "0" (zero). If the user chooses the option to enter the data required in a less structured way enter "1". All available details on the other address of the recipient beneficial owner should then be presented as one left-justified string of 149 bytes, blank or "/" (slash) used as a delimiter between parts of the address. PLEASE NOTE that the address country code is outside this area of choosable format. The use of the fixed form is recommended as a rule to allow easy matching. However, the use of the free form is recommended if the sending state cannot reliably identify and distinguish the different parts of the adress.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
FreeForm5	***(other address free format)**	917	149	an	

If the user has entered "1" in field #35, he has opted to enter the data about the other address of the recipient beneficial owner in one long left-justified field of 149 bytes. Enter the street and number, the city, the country sub-entity - if applicable (state, province or other political subunit) - and the postal (zip) code. Use blank as a delimiter between parts of the address. You may also use "/" (slash) as a delimiter to designate a new line in the printed form of the address. Do not enter the address country code in this field.It is strongly recommended that the address information be presented in the way the state where the recipient beneficial owner resides or is located usually structures the address. Refer to the structure information to which the link "ad-strct" points - depending on implementation.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FixForm5	***(other address fixed format)**				The details of the subdivision in fields are given below.

If the user has entered "0" (zero) in field #35, he has opted to enter the data about the other address of the recipient beneficial owner in four separate fields (#36 through #39).

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F036	Street	917	70	an	

If the user has entered "0" (zero) in field #35, he has opted to enter the data about the other address of the recipient beneficial owner in four separate fields (#36 through #39). In field #36, enter the complete name of the street and the building number where the recipient beneficial owner resides or is situated.It is strongly recommended that the address information be presented in the way the state where the recipient beneficial owner resides or is located usually structures the address. Refer to the structure information to which the link "ad-strct" points - depending on implementation.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F037	City	987	35	an	

If the user has entered "0" (zero) in field #35, he has opted to enter the data about the other address of the recipient beneficial owner in four separate fields (#36 through #39). In field #37, enter the name of the city or municipality where the recipient beneficial owner resides or is situated.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F038	CountrySubentity	1022	35	an	State, Province etc

If the user has entered "0" (zero) in field #35, he has opted to enter the data about the other address of the recipient beneficial owner in four separate fields (#36 through #39). In field #38, enter the name of the country sub-entity if applicable in which the city where the recipient beneficial owner resides or is situated. Sub-entities may include states, regions, laender, provinces, or other political subunits. Leave blank if no data are available.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F039	PostalCode	1057	9	an	

If the user has entered "0" (zero) in field #35, he has opted to enter the data about the other address of the recipient beneficial owner in four separate fields (#36 through #39). In field #39, enter the postal (zip) code of the other address of the recipient beneficial owner.It is strongly recommended that the address information be presented in the way the state where the recipient beneficial owner resides or is located usually structures the address. Refer to the structure information to which the link "ad-strct" points - depending on implementation.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F040	***CountryCode**	1066	2	a	ISO 3166 ; blank if unknown

Enter the ISO 3166 two-byte alpha country code of the address entered in bytes 917 through 1065. This field is not included in the alternatively fixed or free area in order to provide a unique location for this information.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
RAI	***RecipientAgent Or Intermediary**				

This is a party that received the payment in question but is known not to be the recipient beneficial owner (e.g., an intermediary such as a financial institution). If no such party is identified the field group should be left blank - this may even apply to those fields which would have to be numeric in the "existence" case.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F041	**CountryCodeForTIN1	1068	2	a	ISO 3166 ; blank if unknown

If a TIN is entered in field #42, enter the ISO 3166 two-byte alpha country code of the state that issued the TIN. Note that the agent or intermediary may reside or do business anywhere, therefore this can be any country.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F042	**TIN1	1070	20	an	Agent or Intermediary Tax Identification Number or Blank if data not available

If applicable, enter the TIN that the country identified in country code field #41 has issued to the agent or intermediary.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F043	**CountryCodeForTIN2	1090	2	a	ISO 3166 ; blank if unknown

If a TIN is entered in field #44, enter the ISO 3166 two-byte alpha country code of the state that issued the TIN. Note that the agent or intermediary may reside or do business anywhere, therefore this can be any country.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F044	**TIN2	1092	20	an	Agent or Intermediary Tax Identification Number or Blank if data not available

If applicable, enter the TIN that the country identified in country code field #43 has issued to the agent or intermediary. There is no rule whatsoever, if TINs assigned by two countries to the same recipient agent or intermediary are known, which TIN to enter as TIN1 and which as TIN2. However, the two numbers should not be identical (i.e., do not enter the same number both in fields #42 and #44).

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
N_RAI	**(name)				

This is the name of the beneficial owner's agent or intermediary as indicated to the sending country by appropriate documentation or official records. In fields #45 through #49, the user should enter data about the name of the recipient beneficial owner's agent or intermediary if applicable. Refer to those fields for more information.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F045	*NameFormatType	1112	1	an	0 fixed; 1 free; blank acceptable if no agent or intermediary given. If free, all available name details will be given as a left justified string in bytes 1113 through 1322

For fields #46 through #49, the user has the option to enter the data about the name of the recipient beneficial owner's agent or intermediary either as one long left-justified field of 210 bytes in positions 1113 through 1322, or to spread the data over 4 fields. If the user chooses the option to enter the data required in fields #46 through #49 in one field, enter "1". This will be the common case with legal entities. All available name details should then be entered as one left-justified string of 210 bytes, blanks serving as delimiter between name parts. If the user chooses the option to enter the data required in fields #46 through #49 in separate fields, enter "0" (zero). The free format should be used if there is any doubt about the name details (which is keyname, which is first name etc.) as the receiving country may be in a better position to make a correct judgment. Note that "keyname" is intended to be a culturally neutral notion for what otherwise might be called "surname", "last name", "family name" etc. To allow for the whole "agent or intermediary" group of fields to be left blank if no such information is known, this field is defined alphanumeric in contrast to the "name" group of fields and a blank is a valid entry in the case mentioned.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FreeForm6	*(name free format)	1113	210	an	

If the user has entered "1" in field #45 he has opted to enter the data about the name of the recipient beneficial owner's agent or intermediary in one long left-justified field of 210 bytes. For individuals enter the family name, the first name, middle name or initials (if available), the title (if available, e.g. Mr., Mrs.), and/or the suffix (if available, e.g., Esquire, Senior). For legal entities (corporations, partnerships, etc.), enter the registered or documented business name. Use blank as a delimiter between parts of the name. Leave blank if no data are available.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
FixForm6	***(name fixed format)**				The details of the subdivision in fields are given below.

If the user has entered "0" (zero) in field #45, he has opted to enter the data about the name of the recipient beneficial owner's agent or intermediary in four separate fields.In fields #46 to #49, enter for individuals the family name (#46), the first name, middle name and/or initials (if available) (#47), the title (if available, e.g. Mr., Mrs.) (#48) and/or the suffix (if available, e.g. Esquire, Senior) (#49). Note that "keyname" is intended to be a culturally neutral notion for what otherwise might be called "surname", "last name", "family name" etc. For legal entities (corporations, partnerships, etc.) if you do not use free form as recommended, enter the complete business name in field #46.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F046	Keyname	1113	70	an	Keyname is the family name for individuals and the business name for legal entities

If the user has entered "0" (zero) in field #45, he has opted to enter the data about the name of the recipient beneficial owner's agent or intermediary in four separate fields.In field #46 enter the family name for individuals or the business name for legal entities (if you do not use free form as recommended for legal entities).

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F047	OtherNames	1183	70	an	Other name includes first name, middle name and/or initials

If the user has entered "0" (zero) in field #45, he has opted to enter the data about the name of the recipient beneficial owner's agent or intermediary in four separate fields.In field #47, enter the first name, the middle name and/or the initials. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F048	Title	1253	35	an	Title includes: Mr, Mrs, Doctor, Sir, Professor etc.

If the user has entered "0" (zero) in field #45, he has opted to enter the data about the name of the recipient beneficial owner's agent or intermediary in four separate fields.In field #48, enter the title. Titles may include Mr; Mrs; Doctor, Sir, Professor, etc. Abbreviations may be used as customary. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F049	Suffix	1288	35	an	Suffix includes: Esquire, Senior, Junior etc.

If the user has entered "0" (zero) in field #45, he has opted to enter the data about the name of the recipient beneficial owner's agent or intermediary in four separate fields.In field #49, enter the suffix. Suffixes may include Esquire, Senior, Junior, etc. Abbreviations may be used as customary. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
A_RAI	****(Address)**				

This group of fields is used to enter address information about the recipient beneficial owner's agent or intermediary.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F050	***AddrFormatType**	1323	1	an	0 fixed; 1 free; blank acceptable if no agent or intermediary given. If free, all available address details will be given as a left justified string in bytes 1324 through 1472.

For fields #51 through #54, the user has the option to enter the data about the address of the recipient beneficial owner's agent or intermediary either as one long field or to spread the data over four fields. If the user chooses the option to enter the data required in fields #51 through #54 in separate fields, enter "0" (zero). If the user chooses the option to enter the data required in a less structured way enter "1". All available details on the address of the recipient beneficial owner's agent or intermediary should then be presented as one left-justified string of 149 bytes, blank or "/" (slash) used as a delimiter between parts of the address. PLEASE NOTE that the address country code is outside this area of choosable format. The use of the fixed form is recommended as a rule to allow easy matching. However, the use of the free form is recommended if the sending state cannot reliably identify and distinguish the different parts of the adress.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FreeForm7	***(address free format)**	1324	149	an	

If the user has entered "1" in field #50, he has opted to enter the data about the address of the recipient beneficial owner's agent or intermediary in one long left-justified field of 149 bytes. Enter the street and number, the city, the country sub-entity - if applicable (state, province or other political subunit) - and the postal (zip) code. Use blank as a delimiter between parts of the address. You may also use "/" (slash) as a delimiter to designate a new line in the printed form of the address. Do not enter the address country code in this field.It is strongly recommended that the address information be presented in the way the state where the recipient beneficial owner's agent or intermediary resides or is located usually

structures the address. Refer to the structure information to which the link "ad-strct" points - depending on implementation. Leave blank if not applicable or no data are available.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FixForm7	***(address fixed format)**				The details of the subdivision in fields are given below.

If the user has entered "0" (zero) in field #50, he has opted to enter the data about the address of the recipient beneficial owner's agent or intermediary in four separate fields (#51 through #54). Leave blank if not applicable or no data are available.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F051	Street	1324	70	an	

If the user has entered "0" (zero) in field #50, he has opted to enter the data about the address of the recipient beneficial owner's agent or intermediary in four separate fields (#51 through #54). In field #51, enter the complete name of the street and the building number where the recipient beneficial owner's agent or intermediary resides or is situated.It is strongly recommended that the address information be presented in the way the state where the recipient beneficial owner's agent or intermediary resides or is located usually structures the address. Refer to the structure information to which the link "ad-strct" points - depending on implementation.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F052	City	1394	35	an	

If the user has entered "0" (zero) in field #50, he has opted to enter the data about the address of the recipient beneficial owner's agent or intermediary in four separate fields (#51 through #54). In field #52, enter the name of the city or municipality where the recipient beneficial owner's agent or intermediary resides or is situated.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F053	CountrySubentity	1429	35	an	

If the user has entered "0" (zero) in field #50, he has opted to enter the data about the address of the recipient beneficial owner's agent or intermediary in four separate fields (#51 through #54). In field #53, enter the name of the country sub-entity if applicable in which the city where the recipient beneficial owner's agent or intermediary resides or is situated. Sub-entities may include states, regions, laender, provinces, or other political subunits. Leave blank if not applicable or no data are available.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F054	PostalCode	1464	9	an	

If the user has entered "0" (zero) in field #50, he has opted to enter the data about the address of the recipient beneficial owner's agent or intermediary in four separate fields (#51 through #54). In field #54, enter the postal (zip) code of the address of the recipient beneficial owner's agent or intermediary.It is strongly recommended that the address information be presented in the way the state where the recipient beneficial owner's agent or intermediary resides or is located usually structures the address. Refer to the structure information to which the link "ad-strct" points - depending on implementation. Leave blank if not applicable or no data are available.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F055	***CountryCode**	1473	2	a	ISO 3166 ; blank if unknown

Enter the ISO 3166 two-byte alpha country code of the address entered in bytes 1324 through 1472. This field is not included in the alternatively fixed or free area in order to provide a unique location for this information.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
APR	*****ActualPayer**				

This is the source of the payment that is described in this record, as determined by or documented to the sending country.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F056	****CountryCodeForTIN1**	1475	2	a	ISO 3166 ; blank if unknown

If a TIN is entered in field #57, enter the ISO 3166 two-byte alpha country code of the state that issued the TIN. The country code of the sending country is the most likely entry here, although any country code may be indicated.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F057	****TIN1**	1477	20	an	Actual Payer Tax Identification Number or Blank if data not available

If applicable, enter the TIN that the country identified in country code field #56 has issued to the actual payer.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F058	****CountryCodeForTIN2**	1497	2	a	ISO 3166 ; blank if unknown

If a TIN is entered in field #59, enter the ISO 3166 two-byte alpha country code of the state that issued the TIN.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F059	**TIN2	1499	20	an	Actual Payer Tax Identification Number or Blank if data not available

If applicable, enter the TIN that the country identified in country code field #58 has issued to the actual payer. There is no rule whatsoever, if TINs assigned by two countries to the same recipient agent or intermediary are known, which TIN to enter as TIN1 and which as TIN2. However, the two numbers should not be identical (i.e., do not enter the same number both in fields #57 and #59).

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F060	**OECDPayerCode	1519	2	an	See Codelist Table 1

Enter the appropriate type of the payer by choosing out of the codelist #1.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
N_APR	**(name)				

This is the documented name of the actual payer.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F061	*NameFormatType	1521	1	n	0 fixed; 1 free; If free, all available name details will be given as a left justified string in bytes 1522 through 1731

For fields #62 through #65, the user has the option to enter the data about the name of the actual payer either as one long left-justified field of 210 bytes in positions 1522 through 1731, or to spread the data over 4 fields. If the user chooses the option to enter the data required in fields #62 through #65 in one field, enter "1". This will be the common case with legal entities. All available name details should then be entered as one left-justified string of 210 bytes, blanks serving as delimiter between name parts. If the user chooses the option to enter the data required in fields #62 through #65 in separate fields, enter "0" (zero). The free format should be used if there is any doubt about the name details (which is keyname, which is first name etc.) as the receiving country may be in a better position to make a correct judgment. Note that "keyname" is intended to be a culturally neutral notion for what otherwise might be called "surname", "last name", "family name" etc.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FreeForm8	***(name free format)**	1522	210	an	

If the user has entered "1" in field #61 he has opted to enter the data about the name of the actual payer in one long left-justified field of 210 bytes. For individuals enter the family name, the first name, middle name or initials (if available), the title (if available, e.g. Mr., Mrs.), and/or the suffix (if available, e.g., Esquire, Senior). For legal entities (corporations, partnerships, etc.), enter the registered or documented business name. Use blank as a delimiter between parts of the name.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FixForm8	***(name fixed format)**				The details of the subdivision in fields are given below.

If the user has entered "0" (zero) in field #61, he has opted to enter the data about the name of the actual payer in four separate fields.In fields #62 to #65, enter for individuals the family name (#62), the first name, middle name and/or initials (if available) (#63), the title (if available, e.g. Mr., Mrs.) (#64) and/or the suffix (if available, e.g. Esquire, Senior) (#65). Note that "keyname" is intended to be a culturally neutral notion for what otherwise might be called "surname", "last name", "family name" etc. For legal entities (corporations, partnerships, etc.) if you do not use free form as recommended, enter the complete business name in field #62.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F062	Keyname	1522	70	an	Keyname is the family name for individuals and the business name for legal entities

If the user has entered "0" (zero) in field #61, he has opted to enter the data about the name of the actual payer in four separate fields.In field #62 enter the family name for individuals or the business name for legal entities (if you do not use free form as recommended for legal entities).

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F063	OtherNames	1592	70	an	Other name includes first name, middle name and/or initials

If the user has entered "0" (zero) in field #61, he has opted to enter the data about the name of the actual payer in four separate fields.In field #63, enter the first name, the middle name and/or the initials. Leave blank in the case of a legal entity.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F064	Title	1662	35	an	Title includes: Mr, Mrs, Doctor, Sir, Professor etc.

If the user has entered "0" (zero) in field #61, he has opted to enter the data about the name of the actual payer in four separate fields.In field #64, enter the title. Titles may include Mr; Mrs; Doctor, Sir, Professor, etc. Abbreviations may be used as customary. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F065	Suffix	1697	35	an	Suffix includes: Esquire, Senior, Junior etc.

If the user has entered "0" (zero) in field #61, he has opted to enter the data about the name of the actual payer in four separate fields.In field #65, enter the suffix. Suffixes may include Esquire, Senior, Junior, etc. Abbreviations may be used as customary. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
A_APR	****(Address)**				

In this group of fields enter the address information of the actual payer.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F066	***AddrFormatType**	1732	1	n	0 fixed; 1 free; If free, all available address details will be given as a left justified string in bytes 1733 through 1881

For fields #67 through #70, the user has the option to enter the data about the address of the actual payer either as one long field or to spread the data over four fields. If the user chooses the option to enter the data required in fields #67 through #70 in separate fields, enter "0" (zero). If the user chooses the option to enter the data required in a less structured way enter "1". All available details on the address of the actual payer should then be presented as one left-justified string of 149 bytes, blank or "/" (slash) used as a delimiter between parts of the address. PLEASE NOTE that the address country code is outside this area of choosable format. The use of the fixed form is recommended as a rule to allow easy matching. However, the use of the free form is recommended if the sending state cannot reliably identify and distinguish the different parts of the adress.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
FreeForm9	***(address free format)**	1733	149	an	

If the user has entered "1" in field #66, he has opted to enter the data about the address of the actual payer in one long left-justified field of 149 bytes. Enter the street and number, the city, the country sub-entity - if applicable (state, province or other political subunit) - and the postal (zip)

code. Use blank as a delimiter between parts of the address. You may also use "/" (slash) as a delimiter to designate a new line in the printed form of the address. Do not enter the address country code in this field.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FixForm9	***(address fixed format)**				The details of the subdivision in fields are given below.

If the user has entered "0" (zero) in field #66, he has opted to enter the data about the address of the actual payer in four separate fields (#67 through #70).

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F067	Street	1733	70	an	

If the user has entered "0" (zero) in field #66, he has opted to enter the data about the address of the actual payer in four separate fields (#67 through #70). In field #67, enter the complete name of the street and the building number where the actual payer resides or is situated.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F068	City	1803	35	an	

If the user has entered "0" (zero) in field #66, he has opted to enter the data about the address of the actual payer in four separate fields (#67 through #70). In field #68, enter the name of the city or municipality where the actual payer resides or is situated.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F069	CountrySubentity	1838	35	an	

If the user has entered "0" (zero) in field #66, he has opted to enter the data about the address of the actual payer in four separate fields (#67 through #70). In field #69, enter the name of the country sub-entity if applicable in which the city where the actual payer resides or is situated. Sub-entities may include states, regions, laender, provinces, or other political subunits. Leave blank if no data are available.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F070	PostalCode	1873	9	an	

If the user has entered "0" (zero) in field #66, he has opted to enter the data about the address of the actual payer in four separate fields (#67 through #70). In field #70, enter the postal (zip) code of the address of the actual payer. Leave blank if no data are available.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F071	***CountryCode**	1882	2	a	ISO 3166 ; blank if unknown

Enter the ISO 3166 two-byte alpha country code of the address entered in bytes 1733 through 1881. This field is not included in the alternatively fixed or free area in order to provide a unique location for this information.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
PAI	*****Payer Agent Or Intermediary**				

This is a party through which the actual payer has effected the payment. If no such party is identified, the entire field group should be left blank. This even applies to those fields that would have to be numeric in the "existence" case.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F072	****CountryCodeForTIN1**	1884	2	a	ISO 3166 ; blank if unknown

If a TIN is entered in field #73, enter the ISO 3166 two-byte alpha country code of the state that issued the TIN. Note that the agent or intermediary may reside or do business anywhere, therefore this can be any country.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F073	****TIN1**	1886	20	an	If applicable enter a TIN that the country with country code field#72 has issued for the agent or intermediary.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F074	****CountryCodeForTIN2**	1906	2	a	ISO 3166 ; blank if unknown

If a TIN is entered in field #75, enter the ISO 3166 two-byte alpha country code of the state that issued the TIN. Note that the agent or intermediary may reside or do business anywhere, therefore this can be any country.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F075	**TIN2	1908	20	an	Agent or Intermediary Tax Identification Number or Blank if data not available

If applicable, enter the TIN that the country identified in country code field #74 has issued to the actual payer's agent or intermediary.. There is no rule whatsoever, if TINs assigned by two countries to the same recipient agent or intermediary are known, which TIN to enter as TIN1 and which as TIN2. However, the two numbers should not be identical (i.e., do not enter the same number both in fields #73 and #75).

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
N_PAI	**(name)				

This is the documented name of the actual payer's agent or intermediary. In fields #76 through #80, the user should enter data about the name of the actual payer's agent or intermediary if applicable. Refer to those fields for more information.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F076	*NameFormatType	1928	1	an	0 fixed; 1 free; If free, all available name details will be given as a left justified string in bytes 1929 through 2138

For fields #77 to #80, the user has the option to enter the data about the name of the actual payer's agent or intermediary either as one long left-justified field of 210 bytes in positions 1929 through 2138, or to spread the data over 4 fields. If the user chooses the option to enter the data required in fields #77 to #80 in one field, enter "1". This will be the common case with legal entities. All available name details should then be entered as one left-justified string of 210 bytes, blanks serving as delimiter between name parts. If the user chooses the option to enter the data required in fields #77 to #80 in separate fields, enter "0" (zero). The free format should be used if there is any doubt about the name details (which is keyname, which is first name etc.) as the receiving country may be in a better position to make a correct judgment. Note that "keyname" is intended to be a culturally neutral notion for what otherwise might be called "surname", "last name", "family name" etc. To allow for the whole "payer's agent or intermediary" group of fields to be left blank if no such information is known, this field is defined alphanumeric in contrast to the "name" group of fields and a blank is a valid entry in the case mentioned.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FreeForm10	*(name free format)	1929	210	an	

If the user has entered "1" in field #76 he has opted to enter the data about the name of the actual payer's agent or intermediary in one long left-justified field of 210 bytes. For individuals enter the family name, the first name, middle name or initials (if available), the title (if available, e.g.

Mr., Mrs.), and/or the suffix (if available, e.g., Esquire, Senior). For legal entities (corporations, partnerships, etc.), enter the registered or documented business name. Use blank as a delimiter between parts of the name. Leave blank if no data are available.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
FixForm10	***(name fixed format)**				The details of the subdivision in fields are given below.

If the user has entered "0" (zero) in field #76, he has opted to enter the data about the name of the actual payer's agent or intermediary in four separate fields.In fields #77 to #80, enter for individuals the family name (#77), the first name, middle name and/or initials (if available) (#78), the title (if available, e.g. Mr., Mrs.) (#79) and/or the suffix (if available, e.g. Esquire, Senior) (#80). Note that "keyname" is intended to be a culturally neutral notion for what otherwise might be called "surname", "last name", "family name" etc. For legal entities (corporations, partnerships, etc.) if you do not use free form as recommended, enter the complete business name in field #77.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F077	Keyname	1929	70	an	Keyname is the family name for individuals and the business name for legal entities

If the user has entered "0" (zero) in field #76, he has opted to enter the data about the name of the actual payer's agent or intermediary in four separate fields.In field #77 enter the family name for individuals or the business name for legal entities (if you do not use free form as recommended for legal entities).

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F078	OtherNames	1999	70	an	Other name includes first name, middle name and/or initials

If the user has entered "0" (zero) in field #76, he has opted to enter the data about the name of the actual payer's agent or intermediary in four separate fields.In field #78, enter the first name, the middle name and/or the initials. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F079	Title	2069	35	an	Title includes: Mr, Mrs, Doctor, Sir, Professor etc.

If the user has entered "0" (zero) in field #76, he has opted to enter the data about the name of the actual payer's agent or intermediary in four separate fields.In field #79, enter the title. Titles may include Mr; Mrs; Doctor, Sir, Professor, etc. Abbreviations may be used as customary. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F080	Suffix	2104	35	an	Suffix includes: Esquire, Senior, Junior etc.

If the user has entered "0" (zero) in field #76, he has opted to enter the data about the name of the actual payer's agent or intermediary in four separate fields.In field #80, enter the suffix. Suffixes may include Esquire, Senior, Junior, etc. Abbreviations may be used as customary. Leave blank if no data are available or in the case of a legal entity.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
A_PAI	****(Address)**				

In this group of fields, enter address information about the actual payer's agent or intermediary.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F081	***AddrFormatType**	2139	1	an	0 fixed; 1 free; blank acceptable if no agent or intermediary given. If free, all available address details will be given as a left justified string in bytes 2140 through 2289.

For fields #82 through #85, the user has the option to enter the data about the address of the actual payer's agent or intermediary either as one long field or to spread the data over four fields. If the user chooses the option to enter the data required in fields #82 through #85 in separate fields, enter "0" (zero). If the user chooses the option to enter the data required in a less structured way enter "1". All available details on the address of the actual payer's agent or intermediary should then be presented as one left-justified string of 149 bytes, blank or "/" (slash) used as a delimiter between parts of the address. PLEASE NOTE that the address country code is outside this area of choosable format. The use of the fixed form is recommended as a rule to allow easy matching. However, the use of the free form is recommended if the sending state cannot reliably identify and distinguish the different parts of the adress.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FreeForm11	***(address free format)**	2140	149	an	

If the user has entered "1" in field #81, he has opted to enter the data about the address of the actual payer's agent or intermediary in one long left-justified field of 149 bytes. Enter the street and number, the city, the country sub-entity - if applicable (state, province or other political subunit) - and the postal (zip) code. Use blank as a delimiter between parts of the address. You may also use "/" (slash) as a delimiter to designate a new line in the printed form of the address. Do not enter the address country code in this field.It is strongly recommended that the address information be

presented in the way the state where the actual payer's agent or intermediary resides or is located usually structures the address. Refer to the structure information to which the link "ad-strct" points - depending on implementation. Leave blank if not applicable or no data are available.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
FixForm11	***(address fixed format)**				The details of the subdivision in fields are given below.

If the user has entered "0" (zero) in field #81, he has opted to enter the data about the address of the actual payer's agent or intermediary in four separate fields (#82 through #85). Leave blank if not applicable or no data are available.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F082	Street	2140	70	an	

If the user has entered "0" (zero) in field #81, he has opted to enter the data about the address of the actual payer's agent or intermediary in four separate fields (#82 through #85). In field #82, enter the complete name of the street and the building number where the actual payer's agent or intermediary resides or is situated.It is strongly recommended that the address information be presented in the way the state where the actual payer's agent or intermediary resides or is located usually structures the address. Refer to the structure information to which the link "ad-strct" points - depending on implementation. Leave blank if not applicable or no data are available.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F083	City	2210	35	an	

If the user has entered "0" (zero) in field #81, he has opted to enter the data about the address of the actual payer's agent or intermediary in four separate fields (#82 through #85). In field #83, enter the name of the city or municipality where the actual payer's agent or intermediary resides or is situated. Leave blank if not applicable or no data are available.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F084	CountrySubentity	2245	35	an	

If the user has entered "0" (zero) in field #81, he has opted to enter the data about the address of the actual payer's agent or intermediary in four separate fields (#82 through #85). In field #84, enter the name of the country sub-entity if applicable in which the city where the actual payer's agent or intermediary resides or is situated. Sub-entities may include states, regions, laender, provinces, or other political subunits. Leave blank if not applicable or no data are available.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F085	PostalCode	2380	9	an	

If the user has entered "0" (zero) in field #81, he has opted to enter the data about the address of the actual payer's agent or intermediary in four separate fields (#82 through #85). In field #85, enter the postal (zip) code of the address of the actual payer's agent or intermediary.It is strongly recommended that the address information be presented in the way the state where the actual payer's agent or intermediary resides or is located usually structures the address. Refer to the structure information to which the link "ad-strct" points - depending on implementation. Leave blank if not applicable or no data are available.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F086	***CountryCode**	2289	2	a	ISO 3166 ; blank if unknown

Enter the ISO 3166 two-byte alpha country code of the address entered in bytes 2140 through 2288. This field is not included in the alternatively fixed or free area in order to provide a unique location for this information.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F087	***Sending Country TaxYearEnd**	2291	8	an	ISO 8601: CCYYMMDD or CCYYMM or CCYY or Blank if unknown

This is the end of the tax year in the source country. The date to be entered should refer to the tax year in which the income payment was effected. The reason for the inclusion of this field into the record is mainly that in some countries the tax year does not coincide with the calendar year. The data must be provided left-justified with four digits for the year and optionally two digits for the month and another two digits for the day of the month, refer to ISO 8601.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
PD	***(PaymentData)**				

This group of fields contains the data about the payment that is the subject of this record. The term "payment" embodiesthe concept of the legal obligation to put funds at the disposal of the recipient beneficial owner of the income in the manner required by contract or by custom ("constructive receipt of income"). Therefore the interpretation of this term should not be restricted to the actual physical payment of the income (cf.:comment on Model Tax Convention Art.10 and Art.11) As a rule the following equations should hold: field#94 (NIP) = field#92 (GIP) - field#96 (TWH); field#96 (TWH) = field#92 (GIP) * field#97 (TR). Amounts that can be calculated from the others by virtue of these equations do not necessarily have to be entered. If amounts are entered for which the above equations do not hold, an explanation should be provided in the country specific part of this manual, the technical notes accompanying the file and/or in field#104, depending on the scope of the exception. It is assumed that field#99 (tax refund) is not bound to the other amounts by an equation of the above kind.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F088	**DateOfPayment	2299	8	an	ISO 8601: CCYYMMDD or CCYYMM or CCYY or Blank if unknown

Enter the actual date the income was paid. Note the remark on "payment" made in the introduction to this field group (identifier PD). If necessary specific information concerning the date of payment can be included in the country specific part of this manual, the technical notes accompanying the file and/or in field#104. Enter the data left-justified with four digits for the year and optionally two digits for the month and another two digits for the day of the month, refer to ISO 8601. Leave blank if no data are available.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F089	**OECDPaymentType	2307	4	an	See Codelist Table 2

Codelist 2 contains the codes for the various types of income that may be paid to a recipient beneficial owner. Only these codes should be used, as they correspond to the articles of the OECD Model Double Taxation Convention and must be chosen in accordance with the definition given in those articles. Enter the code corresponding to the type of income represented by the amount in field #92. Note that code #15, corresponding to "dependent personal services" has been extended to allow a better description of this type of income. Additional information necessary should be entered in the filler fields at the end of the record (preferably field #103). At present, this standard does not impose rules on how to do this, but leaves this implementation detail for agreement between the countries involved. If a sending countr y feels that no code listed in codelist 2 corresponds to the type of income being reported, use code #21 (other income) and provide an explanation in field #104.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F090	**SendingCountryPaymentType	2311	4	an	

The OECD standard provides common definitions for most income types, thus, deviation from use of the OECD codes given in codelist 2 is discouraged. However, if a sending country is unable to use the OECD codes to describe the income type, it may use its own categories. In this case, the sending country should identify country-specific payment types in this field. The definition of such country-specific payment types must be conveyed to the receiving country by adequate means, preferably in the country-specific part of this manual. Field #90 can also be used in addition to an entry in field #89 in order to make a better distinction. In this case, too, separate information as to the meaning of the codes used is necessary.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
GIP	**GrossIncomePaid				

Fields #91and #92 contain data about the Gross Income Paid.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F091	***CurrencyCode**	2315	3	a	ISO 4217-3 Alpha

Enter the three-byte ISO code for the currency of the gross income. This should be the currency in which the payment has actually been effected. Leave blank to indicate that you do not provide a meaningful amount in field #92 (this allows you to enter a meaningless 0 - zero - and not violate the numeric field type). Never leave blank if there is an amount in field #92, as the information is meaningless without this code.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F092	***Amount-MajorUnitOfCurrency**	2318	18	n	Major units of currency only

This entry must be strictly numerical. Examples: $4000 would be formally wrong as the currency has to be identified by the ISO code in field #91. 4000.71 would be formally wrong as it contains a decimal point and minor currency units. 798,000 would be formally wrong as no editing characters are allowed in a strictly numerical field. 40050 for an amount of 400 dollars and 50 cents would be semantically wrong as it would have to be interpretated as 40050 dollars according to the definition of this standard. The decision to round or truncate to a major currency unit is left to the discretion of the sending country, information about the procedure followed should be included in the country specific part of this manual.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
NIP	****NetIncomePaid**				

Fields #93 to #94 contain data about the Net Income Paid, that is the amount of income after reduction for tax withheld.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F093	***CurrencyCode**	2336	3	a	ISO 4217-3 Alpha

Enter the three-byte ISO code for the currency of the net income paid. This should be the currency in which the payment has actually been effected. Leave blank to indicate that you do not provide a meaningful amount in field #96 (this allows you to enter a meaningless 0 - zero - and not violate the numeric field type). Never leave blank if there is an amount in field #96, as the information is meaningless without this code.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F094	***Amount-MajorUnitOfCurrency**	2339	18	n	Major units of currency only

This entry must be strictly numerical. Examples: $4000 would be formally wrong as the currency has to be identified by the ISO code in field #91. 4000.71 would be formally wrong as it contains a decimal point and minor currency units. 798,000 would be formally wrong as no editing characters are allowed in a strictly numerical field. 40050 for an amount of 400 dollars and 50 cents would be semantically wrong as it would have

to be interpretated as 40050 dollars according to the definition of this standard. The decision to round or truncate to a major currency unit is left to the discretion of the sending country, information about the procedure followed should be included in the country specific part of this manual.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
TWH	****TaxWithheld**				

Fields #95 and #96 contain data about the tax withheld.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F095	***CurrencyCode**	2357	3	a	ISO 4217-3 Alpha

Enter the three-byte ISO code for the currency of the tax withheld. This should be the currency in which the payment has actually been effected. Leave blank to indicate that you do not provide a meaningful amount in field #96 (this allows you to enter a meaningless 0 - zero - and not violate the numeric field type). Never leave blank if there is an amount in field #96, as the information is meaningless without this code.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F096	***Amount-MajorUnitOfCurrency**	2360	18	n	Major units of currency only

This entry must be strictly numerical. Examples: $4000 would be formally wrong as the currency has to be identified by the ISO code in field #91. 4000.71 would be formally wrong as it contains a decimal point and minor currency units. 798,000 would be formally wrong as no editing characters are allowed in a strictly numerical field. 40050 for an amount of 400 dollars and 50 cents would be semantically wrong as it would have to be interpretated as 40050 dollars according to the definition of this standard. The decision to round or truncate to a major currency unit is left to the discretion of the sending country, information about the procedure followed should be included in the country specific part of this manual.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F097	****TaxRate**	2378	4	n or blank	nn.nn decimal point virtual, Blank if no data

Enter the withholding tax rate actually applied to the payment. If used, this must be strictly numerical. The digits entered are always interpreted as hundredth of percents. E.g., 5 percent will be given as 0500, 1550 is a tax rate of 15.5 percent. Blank is a legal entry (and the only legal non-numeric entry) denoting the fact that no tax rate is given.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
TRF	****TaxRefund**				

Enter tax refunded, if any, after the amount entered in field #96 was withheld. If more than one refund is effected with respect to the income of this record enter the sum of these refunds.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F098	***CurrencyCode**	2382	3	a	ISO 4217-3 Alpha

Enter the three-byte ISO code for the currency of the tax refunded. This should be the currency in which the payment has actually been effected. Leave blank to indicate that you do not provide a meaningful amount in field #99 (this allows you to enter a meaningless 0 - zero - and not violate the numeric field type). Never leave blank if there is an amount in field #99, as the information is meaningless without this code.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F099	***Amount-MajorUnitOfCurrency**	2385	18	n	Major units of currency only

This entry must be strictly numerical. Examples: $4000 would be formally wrong as the currency has to be identified by the ISO code in field #91. 4000.71 would be formally wrong as it contains a decimal point and minor currency units. 798,000 would be formally wrong as no editing characters are allowed in a strictly numerical field. 40050 for an amount of 400 dollars and 50 cents would be semantically wrong as it would have to be interpretated as 40050 dollars according to the definition of this standard. The decision to round or truncate to a major currency unit is left to the discretion of the sending country, information about the procedure followed should be included in the country specific part of this manual.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F100	***Date Of Refund Or First Refund**	2403	8	an	ISO 8601: CCYYMMDD or CCYYMM or CCYY or Blank if unknown

Enter the actual date the refund was made. If the amount entered in field #99 reflects more than one refund, enter only the date of the first refund. Enter the data left-justified with four digits for the year and optionally two digits for month and another two digits for the day of the month. Refer to ISO 8601. Leave blank if no data are available.

Field/Field_Group	Field/Group_Name	Start	Length	Data_Type	Remarks
F101	*****Sender´sReference**	2411	70	an	Sender´s number for any queries relating to the particular record

Enter in this field a unique reference "number" (not necessarily strictly numeric) identifying each record transmitted. Even in the case of a repetition (see field #1) this reference must be distinct from the one of the record repeated. All communication about the record should refer to the content of this field. It is best to make this reference "all time unique" for the sending country and not to rely on additional information (tax year,

date of transmission, etc.) not included in the reference to identify the record. It is also advisable to build the reference in a way that links it to the internal records of the sending country.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F102	***CorrectionReference	2481	70	an	Relates to the Sender´s number if this record is a correction or repetition record (see field 1). Otherwise blank.

Be sure to include this reference in the case of correction or repetition of a record sent before, as otherwise correct handling of the record will not be guaranteed or may even be impossible.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F103	***FillerGeneralUse	2551	105	an	Blank unless used. Details to be provided in technical notes accompanying the transmittal.

As a rule this filler field is reserved for changes decided upon by an OECD official procedure. You may also use this field in a way you describe in the technical notes you give to the receiving country at the time of the transmittal, but you should do so only if field #104 does not suffice. The use of field #103 is recommended in connection with the extended OECD payment type 15b, though there is no firm rule how to do this and a bilateral agreement or documentation in technical notes is therefore necessary.

Field/Field_Group	_____Field/Group_Name_____	Start	Length	Data_Type	Remarks
F104	***Filler Specific Arrangements	2656	105	an	Blank unless used. Details agreed between sending country and recipient country and specified in technical notes accompanying the transmittal.

The use of this field is completely at the discretion of the sending country as long as it is made clear to the receiving country what information is reflected.

Codelist1: Type of recipient and type of payer

Code	Meaning	Explanation
01	Individual	
02	Corporation	
03	Partnership	
04	Business organisation other than corporation or partnership	
05	Government or international organisation	
06	Other (specify in filler)	
07	Unknown	Whenever the type of a party is unknown, code 07 has to be specified. Do not leave the field blank.

Codelist2: Type of income code corresponding to the numbering of the Articles of the OECD Model Convention of Income and Capital

Code	Meaning	Explanation
06	Income from immovable property	
07	Business profits	
10	Dividends	
11	Interest	
12	Royalties	
13	Capital Gains	
14	Income from independent personal services	
15	Income from dependent personal services	
15a	Gross amount (including fringe benefits)	
15b	Money amount only but additional information on fringe benefits to follow in fillers according to	

	bilateral agreements	
15c	Money amount only	
16	Directors' fees	
17	Income derived from activities of an artist or sportsman	
18	Pensions	
19	Income from government services and public pensions	
20	Payments to students for education and training	
21	Other income	

OECD STANDARDISED TRANSMISSION FORMAT FOR AUTOMATIC EXCHANGE OF INFORMATION FOR TAX PURPOSES

Background

1. Article 26 of the OECD Model Tax Convention allows for tax information to be exchanged by competent authorities in three ways: on-request, automatically and spontaneously. Information suitable for automatic exchange is typically bulk information comprising many individual cases of the same type, usually consisting of details of income arising from sources in the supplying state where such information is available periodically under that state's own system and can be transmitted automatically on a routine basis. Automatic exchange of information requires standardisation of formats in order to be efficient.

2. The OECD's first step towards standardising the presentation of this type of information resulted in the design, in 1981, of a paper-based form for use in automatic (routine) exchange. The subsequent proliferation of electronic data processing capabilities within tax administrations led to the development of the OECD Standard Magnetic Format (SMF) for automatic exchange of information in 1992, followed by a revised version in 1997[4] (the version currently in use). The SMF contains similar information to the paper-based form but is designed for transmission using electronic media. This enhances data matching in the receiving country and facilitates the exchange of information concerning a large number of taxpayers in a single transmission. The SMF record layout includes fields allocated to the:

- recipient beneficial owner, his agent or intermediary;
- payer of the income, his agent or intermediary;
- residence and source country TINs;
- income derived (tax year, date paid, type of payment, currency, gross and net amount paid, tax withheld, amount refunded etc.).

The Taxation Information Exchange Systems (TIES) Sub-Group has been monitoring the implementation of the SMF since its inception.[5]

Issues

3. The SMF was originally designed for the transmission of information on magnetic tapes. However, magnetic tapes are no longer universally used and for some years countries have been using diskettes and CD-ROMs without formal background to transmit taxpayer information automatically. Moreover the conventional record layout is somewhat inflexible and not a suitable base for future modifications and extensions.

4. In this regard, the TIES Sub-Group has developed a "new generation" transmission format for automatic exchange to replace the SMF. The new format is called the Standard Transmission Format (STF)

4 Refer to Council Recommendations C(92)50/FINAL (23 July 1992) and C(97)30/FINAL (10 July 1997).

5 Refer to the 2002 Survey on the Implementation and use of the OECD SMF (DAFFE/CFA/WP8/TIES(2002)5/CONF) for further information.

and is based on extensible markup language (XML[6]), a document markup language widely used in today's information technology for its many advantages, e.g.:

- separation of the content of a message from any display structure;
- readability both by humans and machines;
- modularity and flexibility;
- ability to check the conformance of documents with the "contract" about its structure;
- availability of a host of tools.

Annex 2 sets out the schemas of the STF (this information is primarily for the benefit of staff involved with the information technology aspects of exchange of information).

5. The STF is intended to co-exist with the SMF rather than to replace it in the near future. The SMF remains an effective tool and the TIES Sub-Group will continue to support its use for the foreseeable future. Migrating to the STF in the short term is not a practical option for most countries that currently use the SMF. However, the majority of OECD countries currently do not use the SMF and the adoption of the STF would allow those countries to move straight to the best available technology when developing the capacity to exchange information automatically. In this regard bridging programmes have been developed to achieve conversion between the two formats, thus enabling treaty partners to engage in bilateral automatic exchange notwithstanding that they might each use a different standard format. Features of the STF that exceed the capabilities of the SMF will not be bridgeable. However, there are currently only a few such features (of minor importance) because compatibility with the SMF was a key design goal for the first version of the STF. Thus the impact for countries that currently use the SMF will be minimal.

6. The STF is ultimately intended to form part (Level 1: Transmission Media) of a framework known as *OECD Standards on Exchange of Information in Taxation* (SEIT). The TIES Sub-Group is presently developing this framework (see Annex I) in relation to three main practical aspects of exchange of information:

- Transmission (how information is physically sent and received by competent authorities: refer to CTPA/CFA/WP8/TIES(2004)1/CONF for a discussion of the electronic transmission of encrypted taxpayer information)
- Security (the protection of confidentiality)
- Format (how information is presented)

7. The Committee on Fiscal Affairs agreed to Working Party No. 8's proposal that the STF be accepted as the new OECD standard format for automatic exchange of information, subject to the following conditions:

- Countries that currently use the SMF can migrate to the STF but for the foreseeable future will not be required to migrate to the STF;
- The TIES Sub-Group will continue to monitor and support the use of the SMF for the foreseeable future;
- The bridging programmes will enable the two formats to operate in parallel.

[6] XML: a technical language for describing documents containing structured information. The term "extensible" refers to a system that can be enlarged by addition rather than by complete replacement.

ANNEX 1

OECD STANDARD FOR EXCHANGE OF INFORMATION IN TAXATION (SEIT)				
	On-request and spontaneous exchange		**Automatic exchange**	
	Physical exchange	**Electronic exchange**	**Physical exchange**	**Electronic exchange**
Level 1 **Transmission media**	Paper documents transmitted via normal mail, commercial courier, diplomatic bag etc.	Email attachment	Magnetic tape, diskette, CD Rom (or DVD[7]) transmitted via normal mail, commercial courier, diplomatic bag etc.	Email attachment[8] Electronic file transfer[9] SOAP[10]
Level 2 **Encryption**	No encryption of paper documents.	Standard Transmission Encryption[11] [see DAFFE/CFA/ WP8/TIES(2003)5/CONF for encryption and key management procedures]	Standard Transmission Encryption (see footnote 13) [see DAFFE/CFA/ WP8/TIES(2003)5/CONF for encryption and key management procedures]	Standard Transmission Encryption (see footnote 13) [see DAFFE/CFA/ WP8/TIES(2003)5/CONF for encryption and key management procedures]
Level 3 **Content Format**	N/A for paper documents.	PDF, JPG or TIFF files for scanned documents. RTF for electronically stored documents.[12]	Standard Magnetic Format (SMF); or Standard Transmission Format (STF)	Standard Magnetic Format (SMF); or Standard Transmission Format (STF)

7 DVDs are not generally used by competent authorities but can hold much more data than CD-ROMs.

8 Automatic exchange via email is possible (subject to file size) but is generally not considered necessary.

9 A potential future method of transmission.

10 A potential future method of transmission. SOAP (Simple Object Access Protocol) is a communications protocol that enables objects created under different systems to invoke each other's methods by exchanging plain text messages in an XML based format over an ordinary HTTP connection. For exchange of information purposes, SOAP can also serve just for the exchange of plain text messages (preferably in an XML format).

11 Gnu Privacy Guard was the encryption software used in the TIES Sub-Group pilot on electronic exchange of Category 3 information. It uses the Pretty Good Privacy (PGP) standard and is compatible with commercial PGP products.

12 PDF (Portable Document Format); JPG (Joint Photographic Group standard format); TIFF (Tagged Image File Format); RTF (Rich Text Format).

ANNEX 2: STF SCHEMAS

MAIN SCHEMA STFDIRECT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="urn:oecd:ties:stf:v1" xmlns="urn:oecd:ties:stf:v1" xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xsd:include schemaLocation="stftypes-1.0.xsd"/>
    <xsd:complexType name="STF_Direct_Type">
        <xsd:sequence>
            <xsd:element name="DocSpec" type="DocSpec_Type">
             <xsd:annotation>
                <xsd:documentation>General information concerning this document - document specification</xsd:documentation>
             </xsd:annotation>
            </xsd:element>
            <xsd:element name="RecipientBeneficialOwner" type="Party_Type">
             <xsd:annotation>
                <xsd:documentation>This is the Recipient Beneficial Owner party of stfdirect. The recipient beneficial owner is the person (legal person or individual), resident of a contracting State, that is entitled to the income for tax purposes and has the benefit thereof, taking into account the economic, legal, factual, and other relevant circumstances (e.g. the relevant double taxation treaty) under which the income is received.
</xsd:documentation>
             </xsd:annotation>
            </xsd:element>
            <xsd:element name="RecipientAgentOrIntermediary" type="Party_Type" minOccurs="0">
             <xsd:annotation>
                <xsd:documentation>Recipient Agent or Intermediary party of stfdirect. This is a party that received the payment in question but is known not to be the recipient beneficial owner (e.g., an intermediary such as a financial institution).
</xsd:documentation>
             </xsd:annotation>
            </xsd:element>
            <xsd:choice>
             <xsd:element name="PayerAgentOrIntermediary" type="Party_Type">
                <xsd:annotation>
                    <xsd:documentation>Payer - Agent or Intermediary party of stfdirect. This is a party through which the actual payer has effected the payment.</xsd:documentation>
                </xsd:annotation>
             </xsd:element>
             <xsd:sequence>
                <xsd:element name="ActualPayer" type="Party_Type">
                    <xsd:annotation>
                     <xsd:documentation>Actual Payer party of stfdirect. This is the source of the payment that is described in this STF-Direct element, as determined by or documented to the sending country.

</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="PayerAgentOrIntermediary" type="Party_Type" minOccurs="0">
                    <xsd:annotation>
                     <xsd:documentation>Payer - Agent or Intermediary party of stfdirect. This is a party through which the actual payer has effected the payment.</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
             </xsd:sequence>
            </xsd:choice>
            <xsd:element name="PaymentData" type="PaymentData_Type">
             <xsd:annotation>
                <xsd:documentation>The information about the payment that this STF-Direct element is to communicate to the receiving country. The term "payment" embodies the concept of the legal obligation to put funds at the disposal of the
```

recipient beneficial owner of the income in the manner required by contract or by custom ("constructive receipt of income"). Therefore the interpretation of this term should not be restricted to the actual physical payment of the income (cf.:comment on Model Tax Convention Art.10 and Art.11)
</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="OtherInfo" type="OtherInfo_Type">
<xsd:annotation>
<xsd:documentation>Any other information regarding this STF-Direct element, e.g. errors when transforming from SMF, Filler info from SMF-Records</xsd:documentation>
</xsd:annotation>
</xsd:element>
</xsd:sequence>
<xsd:attribute name="version" fixed="1.0"/>
</xsd:complexType>
<xsd:element name="STF_OECD">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="MessageSpec" type="MessageSpec_Type"/>
<xsd:element name="STF_DIRECT" type="STF_Direct_Type" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="version" fixed="1.0"/>
</xsd:complexType>
</xsd:element>
</xsd:schema>

Collection of general STF datatypes: Schema stftypes

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="urn:oecd:ties:stf:v1" xmlns="urn:oecd:ties:stf:v1" xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
<xsd:include schemaLocation="isotypes_v1.xsd"/>
<xsd:include schemaLocation="oecdtypes_v1.xsd"/>
<!-- -->
<!-- Simple Types for the Family of OECD STF documents ___ in alphabetical order -->
<!-- -->
<!-- -->
<!-- Technical Indication for Handling of the Document -->
<xsd:simpleType name="DocTypeIndic_Type">
<xsd:annotation>
<xsd:documentation xml:lang="en">This element is used to indicate whether the data in this document part of the message is new -1-, a replication of data already sent (possibly not received) -0- or corrections to data transmitted before -2-. The element applies only to the document part in which it is included. In the case of repeated or corrected data elements CorrMessageRefId and CorrDocRefId must contain the identifiers MessageRefId and DocRefId respectively for the data referred to. Whenever the data referenced by a replication or correction is not found, the transmitted data shall be treated as new. Documents shall be transmitted and, what is even more important, processed in the following order: repeated - new - correction. In the case of a correction the unchanged elements shall be transmitted again (i.e., repeated) - except for the element DocRefId.

</xsd:documentation>
</xsd:annotation>
<xsd:restriction base="xsd:byte">
<xsd:enumeration value="0"/>
<xsd:enumeration value="1"/>
<xsd:enumeration value="2"/>
</xsd:restriction>
</xsd:simpleType>
<!-- -->
<!-- Code for the gender of an individual person -->
<xsd:simpleType name="Gender_Type">
<xsd:annotation>
<xsd:documentation xml:lang="en">
This element can be used to indicate Gender for individuals. Unlike the SMF, this field is optional and can be omitted if the gender is unknown. There is no need to provide a gender element for non-individuals, as these are described by another data type.</xsd:documentation>

```
        </xsd:annotation>
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="M"/>
            <xsd:enumeration value="F"/>
        </xsd:restriction>
    </xsd:simpleType>
    <!-- International Bank Account Number -->
    <xsd:simpleType name="IBAN_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            The International Bank Account Number has to be given here for the account into which the payment in question has been made. Depending on the transmission type this element is optional. Its structure is:
Country code, 2 letters/Check digits, 2 digits/Basic Bank Account Number (BBAN), 10  to 30 alphanumeric characters
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[A-Z]{2}[0-9]{2}[0-9,A-Z]{10,30}"/>
        </xsd:restriction>
    </xsd:simpleType>
    <!-- -->
    <!-- International Securities Identification Number  -->
    <xsd:simpleType name="ISIN_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            The International Securities Identification Number relevant to the reported payment. Its structure is:
Country code, 2 letters/Main code, 9 alphanumeric characters/Check digit, 1 digit
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[A-Z]{2}[0-9,A-Z]{9}[0-9]{1}"/>
        </xsd:restriction>
    </xsd:simpleType>
    <!-- -->
    <!-- Type of the address considered from a legal point of view -->
    <xsd:simpleType name="legalAddressType_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            This is a datatype for an attribute to an address. It serves to indicate the legal character of that address (residential, business etc.)
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="residentialOrBusiness"/>
            <xsd:enumeration value="residential"/>
            <xsd:enumeration value="business"/>
            <xsd:enumeration value="registeredOffice"/>
            <xsd:enumeration value="unspecified"/>
        </xsd:restriction>
    </xsd:simpleType>
    <!-- -->
    <!-- Kind of Name  -->
    <xsd:simpleType name="nameType_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
             It is possible for stf documents to contain several names for the same party. This is a qualifier to indicate the type of a particular name.  Such types include nicknames ('nick'), names under which a party does business ('dba' a short name for the entity, or a name that is used for public acquaintance instead of the official business name) etc. Examples: in the United States, DaimlerChrysler is still known simply as Chrysler, Dr. William Black dba Quality Pedriatics, Inc. 'SMFAliasOrOther' should be chosen if the document is generated from a legacy SMF record, where no further distinction is possible.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="SMFAliasOrOther"/>
            <xsd:enumeration value="indiv"/>
            <xsd:enumeration value="alias"/>
            <xsd:enumeration value="nick"/>
```

```
            <xsd:enumeration value="aka"/>
            <xsd:enumeration value="dba"/>
            <xsd:enumeration value="legal"/>
            <xsd:enumeration value="atbirth"/>
        </xsd:restriction>
    </xsd:simpleType>
    <!--  -->
    <!--  Kind of the Identifier that is provided for a party  -->
    <xsd:simpleType name="partyIdType_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            This is to designate the kind of the identifier that is provided for a party. The party can be identified by a variety of identification numbers, codes etc. Preferably the partyIdType should be a TIN, nevertheless in the absence of a TIN other identifiers may be helpful, such as a tax file number -TFN. The element 'PartyId' that has an attribute 'partyIdType' of type 'partyIdType_Type' has another attribute to indicate the body that has issued the identifier.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="TIN"/>
            <xsd:enumeration value="IdNo"/>
            <xsd:enumeration value="TFN"/>
        </xsd:restriction>
    </xsd:simpleType>
    <!--  -->
    <!--  Kind of the Qualifier that describes a Payment  Type-->
    <xsd:simpleType name="paymentTypeQlf_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            Kind of the qualifier that describes a payment  type. A 'Payment' element is accompanied by up to two elements 'PaymentType' for the indication of the payment's type. 'paymentTypeQlf' is an attribute of 'PaymentType' indicating the codelist this payment type code is taken from.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="opt">
             <xsd:annotation>
                 <xsd:documentation xml:lang="en">OECD  payment type</xsd:documentation>
             </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="cpt">
             <xsd:annotation>
                 <xsd:documentation xml:lang="en">country specific payment type</xsd:documentation>
             </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="sd1">
             <xsd:annotation>
                 <xsd:documentation xml:lang="en">EU savings directive payment type kind 1</xsd:documentation>
             </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="sd2">
             <xsd:annotation>
                 <xsd:documentation xml:lang="en">EU savings directive payment type kind 2</xsd:documentation>
             </xsd:annotation>
            </xsd:enumeration>
        </xsd:restriction>
    </xsd:simpleType>
    <!--  -->
    <!--  Qualifier for a Payment:  Gross or Net Income, Tax Withheld or Refunded -->
    <xsd:simpleType name="paymentQlf_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            Qualifier for a Payment:  Gross or Net Income, Tax Withheld or Refunded
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="gip"/>
```

```
            <xsd:enumeration value="nip"/>
            <xsd:enumeration value="twh"/>
            <xsd:enumeration value="trf"/>
        </xsd:restriction>
    </xsd:simpleType>
    <!--  -->
    <xsd:simpleType name="SWIFT_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            Registration Authority for the Bank Identifier Code:
            Bank code, 4 alphanumeric characters/Country code, 2 letters/Location code, 2 alphanumeric characters /Branch code, 3 alphanumeric characters, optional
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[0-9,A-Z]{4}[A-Z]{2}[0-9,A-Z]{2}([0-9,A-Z]{3})?"/>
        </xsd:restriction>
    </xsd:simpleType>
    <!--  -->
    <!-- A list of entries to mark the tax years relevant to the document (part). The years are in the form of dates denoting the last day of the tax year in the respective country -->
    <xsd:simpleType name="TaxYearList_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            A list of entries to mark the tax years relevant to the document (part). The years are in the form of dates denoting the last day of the tax year in the respective country
            </xsd:documentation>
        </xsd:annotation>
        <xsd:list itemType="xsd:date"/>
    </xsd:simpleType>
    <!--  -->
    <!-- Data type for tax rates -->
    <xsd:simpleType name="TaxRate_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            Data type for tax rates. Tax rates have to be entered as decimal numbers with a total of four digits, two before and two after the decimal point.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:decimal">
            <xsd:totalDigits value="4"/>
            <xsd:fractionDigits value="2"/>
        </xsd:restriction>
    </xsd:simpleType>
    <!--  -->
    <!-- Data type for any kind of numeric data with two decimal fraction digits, especially monetary amounts -->
    <xsd:simpleType name="TwoDigFract_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            Data type for any kind of numeric data with two decimal fraction digits, especially monetary amounts
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:decimal">
            <xsd:fractionDigits value="2"/>
        </xsd:restriction>
    </xsd:simpleType>
    <!--  -->
    <!--  -->
    <!--  Complex Types for the Family of OECD STF documents  -->
    <!--  here: Complex Types of a General Kind  ___  in alphabetical order -->
    <!--  Types that are specific for a particular message type will be defined in the schema of that message document -->
    <!--  -->
    <!-- Description of an Account  -->
    <xsd:complexType name="AcctInfo_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
```

It may be desirable or even necessary to extend the information about a payment by including information about the account that was used for the payment and/or the security to which the payment relates. One or more such entries can be given in an element of this type. The element itself is optional unless stated otherwise for a particular document type, if it is present, however, it must not be empty.

```
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence maxOccurs="unbounded">
            <xsd:choice>
             <xsd:element name="IBAN" type="IBAN_Type"/>
             <xsd:element name="OBAN" type="OBAN_Type"/>
             <xsd:element name="ISIN" type="ISIN_Type"/>
             <xsd:element name="OSIN" type="OSIN_Type"/>
             <xsd:element name="SWIFT" type="SWIFT_Type"/>
            </xsd:choice>
        </xsd:sequence>
    </xsd:complexType>
    <!--  -->
    <!-- Structure of the Address of a Party broken down into its logical Parts -->
    <xsd:complexType name="AddressFix_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
```

Structure of the address for a party broken down into its logical parts, recommended for easy matching. The 'City' element is the only required subelement. All of the subelements are simple text - data type 'string'.

```
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="Street" type="xsd:string" minOccurs="0"/>
            <xsd:element name="BuildingIdentifier" type="xsd:string" minOccurs="0"/>
            <xsd:element name="SuiteIdentifier" type="xsd:string" minOccurs="0"/>
            <xsd:element name="FloorIdentifier" type="xsd:string" minOccurs="0"/>
            <xsd:element name="DistrictName" type="xsd:string" minOccurs="0"/>
            <xsd:element name="POB" type="xsd:string" minOccurs="0"/>
            <xsd:element name="PostCode" type="xsd:string" minOccurs="0"/>
            <xsd:element name="City" type="xsd:string"/>
            <xsd:element name="CountrySubentity" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
    <!--  -->
    <!--  The Address of a Party, given in fixed or free Form, possibly in both Forms -->
    <xsd:complexType name="Address_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
```

The user has the option to enter the data about the address of a party either as one long field or to spread the data over up to eight elements or even to use both formats. If the user chooses the option to enter the data required in separate elements, the container element for this will be 'AddressFix'. If the user chooses the option to enter the data required in a less structured way in 'AddressFree' all available address details shall be presented as one string of bytes, blank or "/" (slash) or carriage return- line feed used as a delimiter between parts of the address. PLEASE NOTE that the address country code is outside both of these elements. The use of the fixed form is recommended as a rule to allow easy matching. However, the use of the free form is recommended if the sending state cannot reliably identify and distinguish the different parts of the address. The user may want to use both formats e.g. if besides separating the logical parts of the address he also wants to indicate a suitable breakdown into print-lines by delimiters in the free text form. in this case 'AddressFix' has to precede 'AddressFree'.

```
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="CountryCode" type="CountryCode_Type"/>
            <xsd:choice>
             <xsd:element name="AddressFree" type="xsd:string"/>
             <xsd:sequence>
                 <xsd:element name="AddressFix" type="AddressFix_Type"/>
                 <xsd:element name="AddressFree" type="xsd:string" minOccurs="0"/>
             </xsd:sequence>
            </xsd:choice>
        </xsd:sequence>
```

```
        <xsd:attribute name="legalAddressType" type="legalAddressType_Type"/>
    </xsd:complexType>
    <!--  -->
    <!--  Document specification: Data identifying and describing the document -->
    <xsd:complexType name="DocSpec_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">Document specification: Data identifying and describing the document, where 'document' here means the part of a message that is to transmit the data about a single payment/transaction or other meaningful self-contained chunk of information. STF messages do not factorize such information in order to transmit repeating data only once (e.g. data about a party that has received multiple payments). 'DocRefId' is an identifier that the sender has to attribute to this document and which has to be unique at least inside the containing message. If the document refers to another one transmitted before, 'CorrMessageRefId' and 'CorrDocRefId' have to contain the  corresponding Id's of the message and document referred to.</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="DocTypeIndic" type="DocTypeIndic_Type"/>
            <xsd:element name="DocRefId" type="xsd:string">
             <xsd:annotation>
                 <xsd:documentation xml:lang="en">Sender's unique identifier of this document </xsd:documentation>
             </xsd:annotation>
            </xsd:element>
            <xsd:element name="CorrMessageRefId" type="xsd:string" minOccurs="0">
             <xsd:annotation>
                 <xsd:documentation  xml:lang="en">Reference id of the message of the document referred to if this is repetition or correction</xsd:documentation>
             </xsd:annotation>
            </xsd:element>
            <xsd:element name="CorrDocRefId" type="xsd:string" minOccurs="0">
             <xsd:annotation>
                 <xsd:documentation  xml:lang="en">Reference  id  of  the  document  referred  to  if  this  is  repetition  or correction</xsd:documentation>
             </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <!--  -->
    <!-- Data (other than Name and Address) to describe and identify an Individual  -->
    <xsd:complexType name="IndivPersData_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            Data (other than Name and Address) to describe and identify an Individual. In general all of the subelements are optional, within certain document types they may be made obligatory. 'Nationality' is of type 'CountryCode_Type' --&amp;gt;, 'BirthDate' is of type date, that is in the  form ccyy-mm-dd, the content of all other subelements is character string.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="Gender" type="Gender_Type" minOccurs="0"/>
            <xsd:element name="Nationality" type="CountryCode_Type" minOccurs="0"/>
            <xsd:element name="BirthDate" type="xsd:date" minOccurs="0"/>
            <xsd:element name="BirthCity" type="xsd:string" minOccurs="0"/>
            <xsd:element name="BirthCitySubentity" type="xsd:string" minOccurs="0"/>
            <xsd:element name="BirthCountryCode" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
    <!--  -->
    <!-- Data (other than Name and Address) to describe and identify a Legal Entity -->
    <xsd:complexType name="LegalPersData_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            Data (other than Name and Address) to describe and identify a legal entity. Currently the foundation date is the only subelement. It is defined as required inside 'LegalPersData_Type', as an empty element should not appear in the document. However, the  element containing LegalPersData is optional. 'FoundDate' is of type date, that is in the  form ccyy-mm-dd.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
```

```
            <xsd:element name="FoundDate" type="xsd:date"/>
        </xsd:sequence>
    </xsd:complexType>
    <!--  -->
    <!--  Message specification: Data identifying and describing the message as a whole -->
    <xsd:complexType name="MessageSpec_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">Message specification: Data identifying and describing the message as a whole. 'SendingCountry' and 'ReceivingCountry' are to identify the relation of the transmission, so that this is visible independently of the transmission context inside and at the very top of the message. The elements are optional as in the SMF record there are no fields exactly  corresponding; it is, however, strongly recommended to use these fields as intended. 'Warning' is for legal constraints: Free text expressing the restrictions for use of the information this message contains and the legal framework under which it is given. 'Contact' should contain all necessary contact information about persons responsible for and involved in the processing of the data transmitted in this message, both legally and technically. This is free text as it is not intended for automatic processing. 'MessageRefId' is a unique identifier that the sender has to attribute to this message and shall be used in any correspondence. 'TaxYearList' is a list of all tax years for which information is transmitted in the documents of the current message. To indicate a tax year, the date of the last day of that year is given. Format for dates is ccyy-mm-dd. List items have to be separated by blanks.</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="SendingCountry" type="CountryCode_Type" minOccurs="0"/>
            <xsd:element name="ReceivingCountry" type="CountryCode_Type" minOccurs="0"/>
            <xsd:element name="Warning" type="xsd:string">
             <xsd:annotation>
                <xsd:documentation xml:lang="en">Free text expressing the restrictions for use of the information this message contains and the  legal framework under which it is given</xsd:documentation>
             </xsd:annotation>
            </xsd:element>
            <xsd:element name="Contact" type="xsd:string">
             <xsd:annotation>
                <xsd:documentation xml:lang="en">All necessary contact information about persons responsible for and involved in the processing of the data transmitted in this message, both legally and technically. Free text as this is not intended for automatic processing. </xsd:documentation>
             </xsd:annotation>
            </xsd:element>
            <xsd:element name="MessageRefId" type="xsd:string">
             <xsd:annotation>
                <xsd:documentation xml:lang="en">Sender's unique identifier for this message</xsd:documentation>
             </xsd:annotation>
            </xsd:element>
            <xsd:element name="TaxYearList" type="TaxYearList_Type">
             <xsd:annotation>
                <xsd:documentation xml:lang="en">A list of all tax years for which information is transmitted in documents of the current message. To indicate a tax year, the date of the last day of that year is given.</xsd:documentation>
             </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <!--  -->
    <!--  General Type for Monetary Amounts -->
    <xsd:complexType name="MonAmnt_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            This data type is to be used whenever monetary amounts are to be communicated. Such amounts shall be given including two fractional digits of the main currency unit. The code for the currency in which the value is expressed has to be taken from the ISO codelist 4217 and added in attribute currCode.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="TwoDigFract_Type">
             <xsd:attribute name="currCode" type="currCode_Type" use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- Structure of the Name of a Party broken down into its logical Parts -->
    <xsd:complexType name="NameFix_Type">
```

```
<xsd:annotation>
    <xsd:documentation xml:lang="en">
    Structure of the name of a party broken down into its logical parts, recommended for easy matching. This type is constructed following the PersonName complex data type of the OASIS CIQ xNL standard. To keep STF as simple as possible it is not formally constructed as a xsd:restriction of that type.

    </xsd:documentation>
</xsd:annotation>
<xsd:sequence>
    <xsd:element name="PrecedingTitle" minOccurs="0" maxOccurs="unbounded">
     <xsd:annotation>
          <xsd:documentation>His Excellency,Estate of the Late ...</xsd:documentation>
     </xsd:annotation>
     <xsd:complexType mixed="true"/>
    </xsd:element>
    <xsd:element name="Title" minOccurs="0" maxOccurs="unbounded">
     <xsd:annotation>
          <xsd:documentation>Greeting title. Example: Mr, Dr, Ms, Herr, etc. Can have multiple titles.</xsd:documentation>
     </xsd:annotation>
     <xsd:complexType mixed="true"/>
    </xsd:element>
    <xsd:element name="FirstName" minOccurs="0" maxOccurs="unbounded">
     <xsd:annotation>
          <xsd:documentation>Represents the position of the name in a name string. Can be Given Name, Forename, Christian Name, Surname, Family Name, etc. Use the attribute "NameType" to define what type this name is.</xsd:documentation>
     </xsd:annotation>
     <xsd:complexType mixed="true">
          <xsd:attribute name="xnlNameType">
               <xsd:annotation>
                <xsd:documentation>Defines the name type of FirstName. Example: Given Name, Forename, Christian Name, Father's Name, etc. In some countries, FirstName could be a Family Name or a SurName. Use this attribute to define the type for this name.</xsd:documentation>
               </xsd:annotation>
          </xsd:attribute>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="MiddleName" minOccurs="0" maxOccurs="unbounded">
     <xsd:annotation>
          <xsd:documentation>Middle name (essential part of the name for many nationalities). Represents the position of the name in the name string. Example: Sakthi in "Nivetha Sakthi Shantha". Can have multiple middle names.</xsd:documentation>
     </xsd:annotation>
     <xsd:complexType mixed="true">
          <xsd:attribute name="xnlNameType">
               <xsd:annotation>
                <xsd:documentation>Defines the name type of Middle Name. Example: First name, middle name, maiden name, father's name, given name, etc.</xsd:documentation>
               </xsd:annotation>
          </xsd:attribute>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="NamePrefix" minOccurs="0">
     <xsd:annotation>
          <xsd:documentation>de, van, van de, von, etc. Example: Derick de Clarke</xsd:documentation>
     </xsd:annotation>
     <xsd:complexType mixed="true">
          <xsd:attribute name="xnlNameType">
               <xsd:annotation>
                <xsd:documentation>Defines the type of name associated with the NamePrefix. For example the type of name is LastName and this prefix is the prefix for this last name.</xsd:documentation>
               </xsd:annotation>
          </xsd:attribute>
     </xsd:complexType>
    </xsd:element>
```

```
            <xsd:element name="LastName" minOccurs="0" maxOccurs="unbounded">
             <xsd:annotation>
                 <xsd:documentation>Represents the position of the name in a name string. Can be Given Name, Forename, Christian Name, Surname, Family Name, etc. Use the attribute "NameType" to define what type this name is.</xsd:documentation>
             </xsd:annotation>
             <xsd:complexType mixed="true">
                 <xsd:attribute name="xnlNameType">
                     <xsd:annotation>
                      <xsd:documentation>Defines the name type of LastName. Example: Father's name, Family name, Sur Name, Mother's Name, etc. In some countries, LastName could be the given name or first name.</xsd:documentation>
                     </xsd:annotation>
                 </xsd:attribute>
             </xsd:complexType>
            </xsd:element>
            <xsd:element name="GenerationIdentifier" minOccurs="0" maxOccurs="unbounded">
             <xsd:annotation>
                 <xsd:documentation>Jnr, Thr Third, III</xsd:documentation>
             </xsd:annotation>
             <xsd:complexType mixed="true"/>
            </xsd:element>
            <xsd:element name="Suffix" minOccurs="0" maxOccurs="unbounded">
             <xsd:annotation>
                 <xsd:documentation>Could be compressed initials - PhD, VC, QC</xsd:documentation>
             </xsd:annotation>
             <xsd:complexType mixed="true"/>
            </xsd:element>
            <xsd:element name="GeneralSuffix" minOccurs="0">
             <xsd:annotation>
                 <xsd:documentation>Deceased, Retired ...</xsd:documentation>
             </xsd:annotation>
             <xsd:complexType mixed="true"/>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <!--       The Name of a Party, given in fixed or free Form, possibly in both Forms -->
    <xsd:complexType name="Name_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            The user has the option to enter the data about the name of a party either as one long field or to spread the data over up to six elements or even to use both formats. If the user chooses the option to enter the data required in separate elements, the container element for this will be 'NameFix'. If the user chooses the option to enter the data required in a less structured way in 'NameFree' all available details on the name of the party shall be presented as one string of bytes, blank or "/" (slash) used as a delimiter between parts of the name.  The use of the fixed form is recommended as a rule to allow easy matching. However, the use of the free form is recommended if the sending state cannot reliably identify and distinguish the different parts of the name. The user may want to use both formats in special circumstances. In this case 'NameFix' has to precede 'NameFree'.
An optional attribute 'nameType' can be used to indicate a special kind of name, as for instance a nickname, a name-at-birth etc.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:choice>
            <xsd:element name="NameFree" type="xsd:string"/>
            <xsd:sequence>
             <xsd:element name="NameFix" type="NameFix_Type"/>
             <xsd:element name="NameFree" type="xsd:string" minOccurs="0"/>
            </xsd:sequence>
        </xsd:choice>
        <xsd:attribute name="nameType" type="nameType_Type" use="optional"/>
    </xsd:complexType>
    <!--  -->
    <!--  Other Bank Account Number: A Bank Account Number other than the standard IBAN, the attribute to indicate the kind of such number  -->
    <xsd:complexType name="OBAN_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
```

```
            Other Bank Account Number: A bank account number other than the standard IBAN, the attribute 'acctNoQlf' has to be used to indicate the kind of such number.
                </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                 <xsd:attribute name="acctNoQlf" type="xsd:string" use="required"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
        <!--  -->
        <!--  Other Security Identification Number: A Security Identification Number other than the standard ISIN, the attribute to indicate the kind of such number  -->
        <xsd:complexType name="OSIN_Type">
            <xsd:annotation>
                <xsd:documentation xml:lang="en">
                Other Security Identification Number: A security identification number other than the standard ISIN, the attribute 'secNoQlf' has to be used to indicate the kind of such number.

                </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                 <xsd:attribute name="secNoQlf" type="xsd:string" use="required"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
        <!-- A "Filler" type to accommodate any additional information elements a sender might want to add in order to enhance the value of the "standard" content -->
        <xsd:complexType name="OtherInfo_Type" mixed="true">
            <xsd:annotation>
                <xsd:documentation xml:lang="en">
                OtherInfo is an element to accommodate any additional information that a sender might want to add in order to enhance the value of the "standard" content. The sender has to make sure both by using adequate tag names and adding explanations that the receiver is able to understand sender's intention. As the document is possibly processed automatically there is no guarantee when or even that the content will be recognized by the receiver.
                </xsd:documentation>
            </xsd:annotation>
            <xsd:complexContent mixed="true">
                <xsd:restriction base="xsd:anyType">
                 <xsd:sequence>
                      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                 </xsd:sequence>
                </xsd:restriction>
            </xsd:complexContent>
        </xsd:complexType>
        <!-- Identifier for a Party -->
        <xsd:complexType name="PartyId_Type">
            <xsd:annotation>
                <xsd:documentation xml:lang="en">
                This is the type of an element 'PartyId' which is to contain an identification number/identification code for the party in question. As the identifier may be not strictly numeric, it is just defined as a string of characters. Attributes 'partyIdType' and 'issuedBy' are required to designate the kind (e.g. TIN) and issuer of the identifier. In the case of a TIN the issuer attribute has to be the ISO country code of the issuing country. This has to be guaranteed by the sender without  the type of issuedBy being formally restricted to CountryCode_Type. (In non-TIN cases issuedBy may have to contain some information the kind of which is not known in advance, so as to the formal typing we have to stay here somewhat ambiguous.)
                </xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                 <xsd:attribute name="partyIdType" type="partyIdType_Type" use="required"/>
                 <xsd:attribute name="issuedBy" type="xsd:string" use="required"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
        <!--  Collection of all Data describing a Party -->
```

```
    <xsd:complexType name="Party_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            This container brings together all data about a party. Name and address are required components and each can be present more than once to enable as complete a description as possible. Whenever possible one or more identifiers (TIN etc) should be added as well as a residence country code.  Additional data that describes and identifies the party can be given in the 'PersData' element. The code for the legal type according to the OECD codelist must be added.  The structures of all of the subelements are defined elsewhere in this schema.</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="ResCountryCode" type="CountryCode_Type" minOccurs="0"/>
            <xsd:element name="PartyId" type="PartyId_Type" minOccurs="0" maxOccurs="3"/>
            <xsd:element name="Name" type="Name_Type" maxOccurs="unbounded"/>
            <xsd:element name="Address" type="Address_Type" maxOccurs="unbounded"/>
            <xsd:element name="PersData" type="PersData_Type" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="oecdLegalType" type="oecdLegalType_Type" use="required"/>
    </xsd:complexType>
    <!--  -->
    <!-- Kind of the Payment -->
    <xsd:complexType name="PaymentType_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            A code has to be entered here to describe the nature of the payment. As this code can be taken from a variety of codelists, the attribute 'paymentTypeQlf' --&amp;gt; will indicate the relevant codelist and may itself be qualified by another - optional - attribute 'paymentTypeQlfQlf'. Example: A paymentTypeQlf value of 'cpt' indicates that the value for PaymentType has been taken from a country specific codelist; in this case paymentTypeQlfQlf should give details about that codelist, e.g. the issuing country. As the data type for paymentTypeQlfQlf is just "string", there is no restriction on the format of the information contained in this element. If paymentTypeQlf has a value of 'opt' the content of the element has to be not only of type xsd:string but of oecdPaymentType_Type; this, however, is not reflected - nor enforced - by this schema.

            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
             <xsd:attribute name="paymentTypeQlf" type="paymentTypeQlf_Type" use="required"/>
             <xsd:attribute name="paymentTypeQlfQlf" type="xsd:string" use="optional"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- Data (other than Name and Address) to describe and identify a Party  -->
    <xsd:complexType name="PersData_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            Data (other than Name and Address) to describe and identify a party. Depending on the type of the party (individual or legal person) element 'IndivPersData' or element 'LegalPersData' must be used.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:choice>
            <xsd:element name="IndivPersData" type="IndivPersData_Type"/>
            <xsd:element name="LegalPersData" type="LegalPersData_Type"/>
        </xsd:choice>
    </xsd:complexType>
    <!--  -->
    <!-- Collection of all Data describing a Payment  -->
    <xsd:complexType name="PaymentData_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">Payment Data within international data exchange for tax purposes.  In this container all payment data is brought together. The element 'PaymentType' is required, can occur twice and describes the reason for the payment (e.g. interest). It may use different terms (OECD codelist vs. country specific codelist). This type can involve several  "Payment" elements, which represent part of a group  (gross payment, net payment, tax deducted etc.). To indicate the relevant tax year, the date of the last day of that year is given.  In a PaymentData element as a rule the following equations should hold between the amounts in the Payment child elements that are distinguished by the paymentQlf attribute: NIP = GIP - TWH; TWH = GIP * TR (the paymentQlf values are used in these equations to identify the amounts between which the equations hold) . Amounts that can be calculated from the others by virtue of these equations do not necessarily
```

have to be entered. If amounts are entered for which the above equations do not hold, an explanation should be provided in the OtherInfo element. It is assumed that tax refund is not bound to the other amounts by an equation of the above kind.
</xsd:documentation>
</xsd:annotation>
<xsd:sequence>
<xsd:element name="TaxYearEnd" type="xsd:date">
<xsd:annotation>
<xsd:documentation xml:lang="en">To indicate the relevant tax year, the date of the last day of that year is given.</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="PaymentType" type="PaymentType_Type" maxOccurs="2"/>
<xsd:element name="Payment" type="Payment_Type" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
<!-- -->
<!-- A single Item describing one Aspect of a Payment, e.g. a Net Payment, a Tax Payment -->
<xsd:complexType name="Payment_Type">
<xsd:annotation>
<xsd:documentation xml:lang="en">
A single item describing one aspect of a payment, e.g. a net payment, a tax payment - distinguished by the required attribute 'paymentQlf'. The payment should be described as precisely as possible, even if all of the subelements except the monitary amount itself are optional.
</xsd:documentation>
</xsd:annotation>
<xsd:sequence>
<xsd:element name="PaymentDate" type="xsd:date" minOccurs="0"/>
<xsd:element name="MonAmnt" type="MonAmnt_Type"/>
<xsd:element name="AcctInfo" type="AcctInfo_Type" minOccurs="0"/>
<xsd:element name="TaxRate" type="TaxRate_Type" minOccurs="0"/>
</xsd:sequence>
<xsd:attribute name="paymentQlf" type="paymentQlf_Type" use="required"/>
</xsd:complexType>
</xsd:schema>

Codelist Schemas (for ISO codes only an extract is shown)

isotypes_v1.xsd

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
<!-- ISO 3166 alpha 2 Country Code -->
<xsd:simpleType name="CountryCode_Type">
<xsd:annotation>
<xsd:documentation xml:lang="en">
The appropriate country code from the ISO 3166 two-byte alpha version for the state of which the party concerned is a resident. Omit this only if no data is available.
Valid entries are:
- AF -- AFGHANISTAN
- AL -- ALBANIA
...
- ZM -- ZAMBIA
- ZW -- ZIMBABWE
</xsd:documentation>
</xsd:annotation>
<xsd:restriction base="xsd:string">
<xsd:enumeration value="AF"/>
<xsd:enumeration value="AL"/>
...
<xsd:enumeration value="ZM"/>
<xsd:enumeration value="ZW"/>
</xsd:restriction>
</xsd:simpleType>
<!-- ISO 4217 alpha 3 Currency Code -->
<xsd:simpleType name="currCode_Type">
<xsd:annotation>

```
            <xsd:documentation xml:lang="en">
            The appropriate currency code from the ISO 4217 three-byte alpha version for the currency in which a monetary amount is expressed.
Valid entries are:
AED United Arab Emirates, Dirhams
AFA Afghanistan, Afghanis
...
ZMK Zambia, Kwacha
ZWD Zimbabwe, Zimbabwe Dollars
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="AED"/>
            <xsd:enumeration value="AFA"/>
            ...
            <xsd:enumeration value="ZMK"/>
            <xsd:enumeration value="ZWD"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

oecdtypes_v1.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xsd:simpleType name="oecdLegalType_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            The OECD code describing the legal status of the party:
            01 Individual
            02 Corporation
            03 Partnership
            04 Business organisation other than corporation or partnership
            05 Government or international organisation
            06 Other (specify in the 'OtherInfo' element)
            07 Unknown
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="01"/>
            <xsd:enumeration value="02"/>
            <xsd:enumeration value="03"/>
            <xsd:enumeration value="04"/>
            <xsd:enumeration value="05"/>
            <xsd:enumeration value="06"/>
            <xsd:enumeration value="07"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="oecdPaymentType_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            The OECD code describing the nature of the payments:
            06 Income from immovable property
            07 Business profits
            10 Dividends
            11 Interest
            12 Royalties
            13 Capital Gains
            14 Income from independent personal services
            15 Income from dependent personal services
            15a Gross amount (including fringe benefits)
            15b Money amount only but additional information on fringe benefits to follow in fillers according to bilateral agreements
            15c Money amount only
            16 Directors' fees
```

```
            17 Income derived from activities of an artist or sportsman
            18 Pensions
            19 Income from government services and public pensions
            20 Payments to students for education and training
            21 Other income

            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="06"/>
            <xsd:enumeration value="07"/>
            <xsd:enumeration value="10"/>
            <xsd:enumeration value="11"/>
            <xsd:enumeration value="12"/>
            <xsd:enumeration value="13"/>
            <xsd:enumeration value="14"/>
            <xsd:enumeration value="15"/>
            <xsd:enumeration value="15a"/>
            <xsd:enumeration value="15b"/>
            <xsd:enumeration value="15c"/>
            <xsd:enumeration value="16"/>
            <xsd:enumeration value="17"/>
            <xsd:enumeration value="18"/>
            <xsd:enumeration value="19"/>
            <xsd:enumeration value="20"/>
            <xsd:enumeration value="21"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

USER GUIDE FOR THE OECD STANDARD TRANSMISSION FORMAT

The OECD **S**tandard **T**ransmission **F**ormat for international information exchange in taxation

An introduction

Content

1. Where does STF fit in?

1.1 STF has been defined as the successor of SMF, the OECD **S**tandard **M**agnetic **F**ormat for international information exchange in direct taxation, adopted in 1992 and re-formulated in 1997. To date there is no time limit set for the co-existence of the SMF with the STF.

1.2 STF is part of the SEIT (Standards for Exchange of Information in Taxation) family of OECD recommendations for international information exchange in taxation. In this set of recommendations the responsibility of STF is to define the format of message content, which is achieved by way of Extensible Markup Language (XML) schema. STF is not concerned with the way messages are transmitted, encrypted etc.

1.3 Whilst an important design objective for STF was to stay as closely as possible with SMF (thus making bridging programs possible), it is also a medium term goal to make STF compatible with emerging international XML standards in taxation as aspired to by the Organization for the Advancement of Structured Information Standards (OASIS) TaxXML Technical Committee (TC). It is the intention both of the OASIS TaxXML TC and the OECD TIES group to work for such convergence.

2. What content is STF supposed to support?

2.1 SMF was constructed to support automatic information exchange (in the sense of Article 26 of the OECD Model Convention) for direct tax purposes. Being primarily – even if not only - a modern version of SMF, STF, too, supports this kind of exchange. So the first message format built with STF has been STF_DIRECT for exactly this sort of information.

2.2 It was, however, also a design objective for STF to be flexible and extensible. Therefore STF can easily be extended for any other kind of tax information messages. This includes both the use for other than automatic exchange and for other content than the conventional income information of the SMF type.

3. What is the main structure of an STF message?

3.1 As usual for messages, STF messages are hierarchically structured with a header (MessageSpec) specifying technical information for the message as a whole and an arbitrary number of detail documents. (In a context like this we use the word “document” in a general sense, not in the strict meaning of XML, where a document is always the most comprehensive unit that contains one and only one root element. Documents in the strict XML sense are what we call messages here.) In the present state of STF development there is only one kind of such documents defined (STF_DIRECT), but as soon as other document formats will be developed, they can be included in such messages as well.

Figure 1 depicts this overall structure.

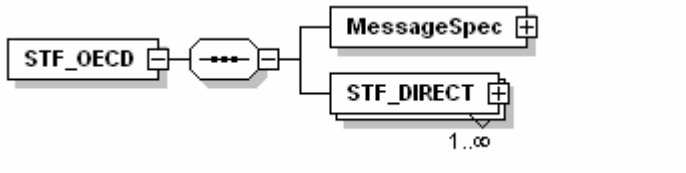

Figure 1: Overall Structure of STF Messages

In XML schema terms this is expressed as

```
<xsd:element name="STF_OECD">
        <xsd:complexType>
            <xsd:sequence>
             <xsd:element name="MessageSpec" type="MessageSpec_Type"/>
             <xsd:element name="STF_DIRECT" type="STF_Direct_Type" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="version" fixed="1.0"/>
        </xsd:complexType>
</xsd:element>
```

SchemaFragment 1

An attribute of name “version” and value “1.0” designates the current status of development.

3.2 The structure of the message specification (header) element is shown in figure 2.

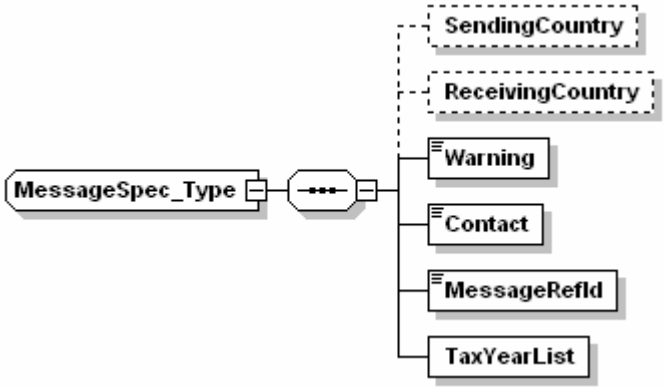

Figure 2: Structure of the Message Specification Header

The element contains data identifying and describing the message as a whole. 'SendingCountry' and 'ReceivingCountry' are to identify the relation of the transmission, so this is visible at the very top of the message and independentl of the transmission content further downstream. The elements are optional because

- they are not indispensable for successful transmission, ,
- there are no exactly corresponding fields in the SMF record
- STF shall closely resemble the SMF.

It is, however, strongly recommended to use these identifying fields as in tended.
‘Warning' is for legal constraints: free text expressing the restrictions for use of the information this message contains and the legal framework under which it is exchanged. 'Contact' should contain all necessary contact information about persons responsible for and involved in the processing of the data transmitted in this message, both legally (competent authority) and technically. This is free text as it is not intended for automatic processing. 'MessageRefId' is a unique identifier that the sender has to attribute to this message and shall be used in any correspondence. 'TaxYearList' is a list of all tax years for which information is transmitted in the documents of the current message. To indicate a tax year,

the date of the last day of that year is given. Format for dates is ccyy-mm-dd. List items have to be separated by blanks.

4. What is the modular structure of the schemas for STF definition?

4.1 STF documents are XML documents. The TIES Technology Task Team has defined:

(1) an XML schema document (stftypes-1.0.xsd) containing a set of simple and complex data types for the use in any STF schema defining a particular document type

(2) an XML schema document (stfdirect-1.0.xsd) for the definition of the XML documents that will replace SMF records, together with the definition of the message container STF_OECD for these documents

(3) two additional XML schema documents for OECD and ISO code lists to be used in STF documents, these schemas contain enumerations of the admissible code-values.

4.2 The core of STF is the definition of the data types to be used in STF documents. It is expected that this set of types will be extended as soon as new documents will be defined. With the advent of new document definitions there will certainly arise additional needs that were not yet addressed from a purely stfdirect perspective. Such new types are expected to fall into three categories:

(1) types that – even if not necessary for stfdirect – are of a certain generality and shall therefore be added to the stftypes collection

(2) types that are specially needed for just a certain document definition without a more general usefulness; they shall be added in a separate XML schema for the use of that one document definition only

(3) types that though close to others already defined in stftypes differ somewhat for the modelling of some aspect in the new document; they shall be derived as extensions or restrictions of their general stftypes relatives.

4.3 As long as stfdirect is the only document type in the STF family, it is considered adequate not to complicate the schema structure more than needed for this situation. Therefore:

(1) the general message structure and the stfdirect document structure are defined inside the same schema;

(2) all the above mentioned schemas are put into the same namespace (urn:oecd:ties:stf:v1);

This results in the structure shown in figure 3.

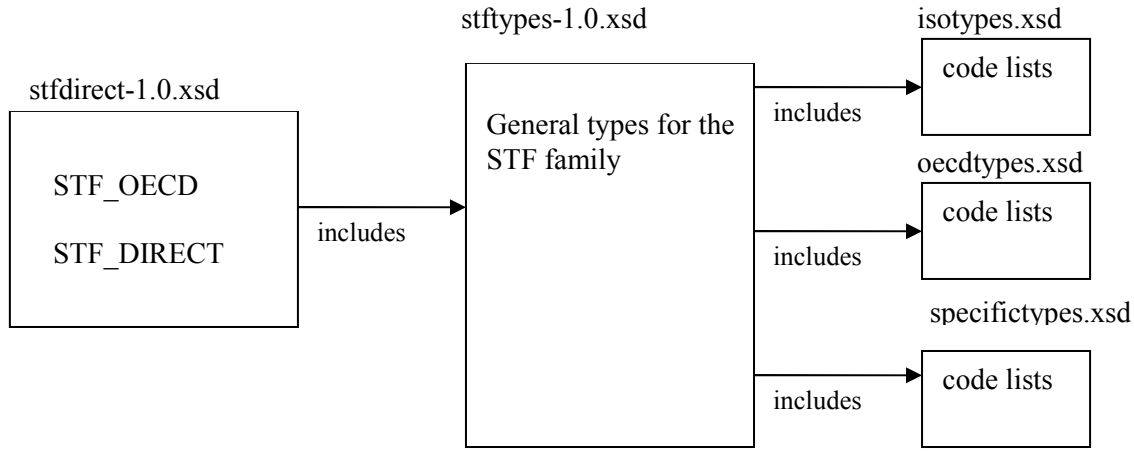

Figure 3: Inclusion Structure of STF schemas (to date)

4.4 When new document types are added in the future, the structure of figure 3 will probably not be adequate any more. Every document type will then be defined in a separate namespace together with its special data types and the general STF (core) types including the type for the message will be imported.

5. What is the structure of an STF_DIRECT document?

5.1 The high-level structure of an STF_DIRECT document is shown in figure 4.

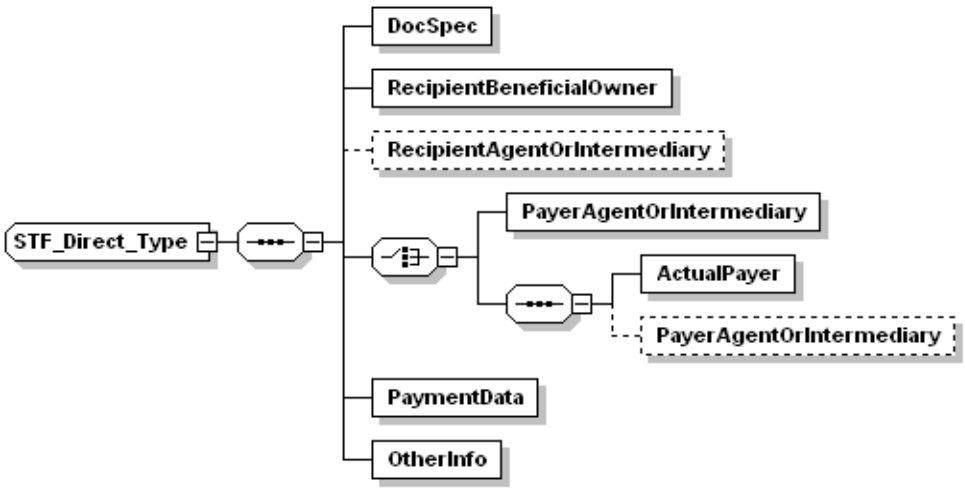

Figure 4: STF_DIRECT, highest level

It will be noted that the components of this structure fall into four categories:

- DocSpec, PaymentData and OtherInfo each represent a particular type of information occuring once in the document.

- All other components are of the same category: they denote parties in the transaction reported.

The construction may at first seem somewhat complicated, but should be rather clear after some inspection. A dotted line box indicates "optional", data for parties so denotated can either be present or not, boxes with solid lines indicate obligatory entries. In a document of stfdirect type data for the beneficial owner must be present, whereas data for an agent or intermediary acting on behalf of the beneficial owner need not be present. The modelling of the data for the payer side is a bit more sophisticated. We have to make sure that data for at least one of "actual payer" and "payer agent or intermediary" are provided, but there is no stringent rule that a particular one of those is obligatory. The "choice" symbol  stands for "one of these". You may want to verify that the construction given in Figure 4 allows for all of these situations:

- actual payer data only
- payer agent or intermediary data only
- both actual payer and payer agent or intermediary data

and at the same time requires one of these situations .

5.2 The DocSpec element serves as a descriptor of the particular stfdirect document to which it belongs, just as the MessageSpec element does for the whole collection of documents in the message. DocSpec has this structure:

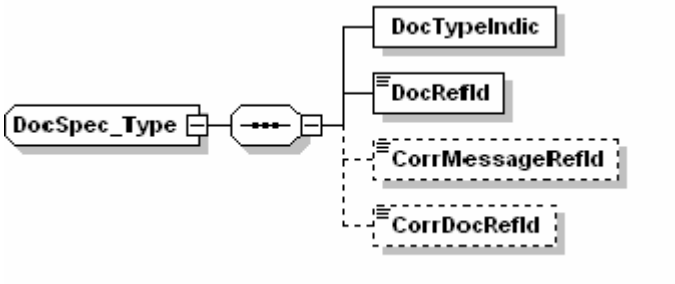

Figure 5: Document Specification Element

The document type indicator (DocTypeIndic) contains administrative data about the status of the document (is it "new" data sent for the first time – the normal case, hopefully near to 100% of documents -, or is it a correction of a document sent before, or is it a repetition of a document which was sent before but possibly not received in an orderly way).

The document reference identification (DocRefId) is the unique identifier of this document. For later reference to be possible it has to be unique at least within the message in which it is contained.

The following two elements are optional and only needed in case of a correction or a repetitive sending. (As they are actually obligatory and not optional in these cases, the schema here is a somewhat weak model of the overall situation.) The elements refer to documents sent before by giving the DocRefId and MessageRefId of the document referred to and the message it was in.

5.3 Payment data are the reason why the document is sent. Here the sending administration enters the information that has become known about income of the beneficial owner in the source country. Here is the structure of the element:

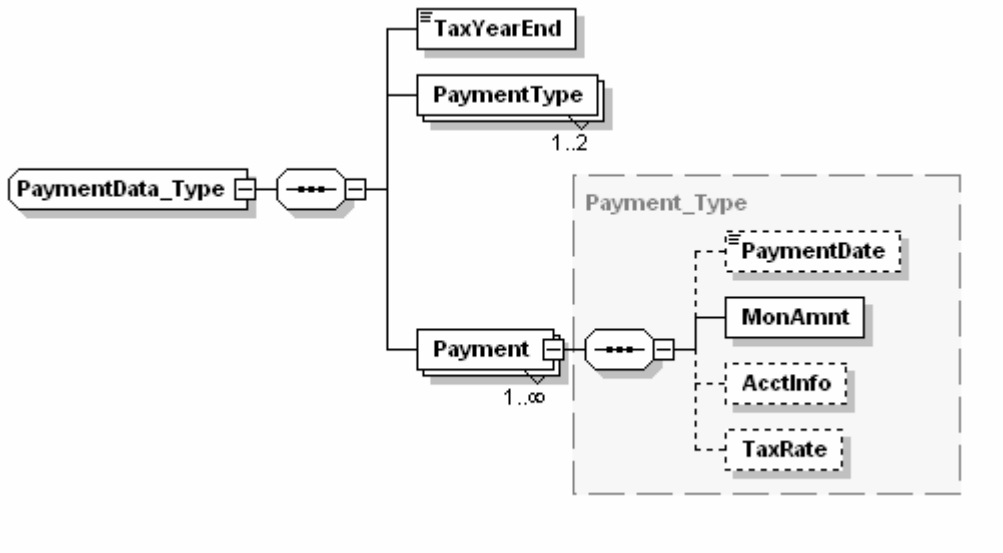

Figure 6: Data about the Income

Each single document serves for information about one and only one income item of the beneficial owner. It follows that several documents have to be transmitted (preferably in the same message) if there is the need to inform about income from several sources, at several points of time etc.

The tax year to which the payment belongs is entered in the element TaxYearEnd, which is a date field (format ccyy-mm-dd in coherence with general XML rules). This is not just a four digit element for the year in order to cope with cases where the tax year does not coincide with the calendar year.

The type of the payment received by the beneficiary is coded in the elements OECDPaymentType and SpecificPaymentType. Their structure is governed by these schema definitions:

```
<xsd:simpleType name="OECDPaymentType_Type">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
        The OECD code describing the nature of the payments:
        06 Income from immovable property
        ……
        21 Other income
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="06"/>
        ……
        <xsd:enumeration value="21"/>
    </xsd:restriction>
</xsd:simpleType>
```

SchemaFragment *2a*

```
<xsd:complexType name="SpecificPaymentType_Type">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">Type for explanation of a payment by a code that is specific for a certain legislation, e.g. for the sending country. In the OECD file for this schema part is a dummy code. The enumeration element and the annotation-documentation in the OECD prepared file serve as an example for real legislation specific codes and their documentation.</xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="SpecifPaymentType_Type">
             <xsd:attribute name="specificPaymentTypeQlf" type="xsd:string" fixed="Dummy"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:simpleType name="SpecifPaymentType_Type">
```

```
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="00"/>
        </xsd:restriction>
    </xsd:simpleType>
```

SchemaFragment 2b

In order to provide sufficient freedom for describing the nature of the income a country/legislation specific payment code may be included in addition to the standard OECD payment code. A sending country may want to transmit a special income code used in this country to best describe what income the beneficiary has received.

If no such specific code is transmitted, the element SpecificPaymentType should not be used. If it is used nevertheless, it has to look exactly like this in order to keep the document from becoming invalid:

```
<SpecificPaymentType  specificPaymentTypeQlf ="Dummy">00</SpecificPaymentType>
```

A sending country that wants to use specific payment codes has to edit the file specifictypes_v1.xsd. This file keeps the (country) specific codes (by now just for payment types) separate from the general OECD types. The attribute specificPaymentTypeQlf, which has to be fixed for all documents sent by this country and relying on this particular set of payment codes, has to be set to a value identifying this code list (e.g. country code + year). Then the enumeration of the codes in specifictypes_v1.xsd has to be adjusted according to need and should be accompanied by proper explanation of the meaning of the codes.

For the payment itself there is a multiplicity of elements. It has to be born in mind that all of these elements belong to one and only one income item; they represent different aspects of this income item, as the gross payment, the net payment, the tax, and the refund. Here is what the element has to look like in XML schema format:

```
<xsd:complexType name="Payment_Type">
    <xsd:sequence>
        <xsd:element name="PaymentDate" type="xsd:date" minOccurs="0"/>
        <xsd:element name="MonAmnt" type="MonAmnt_Type"/>
        <xsd:element name="AcctInfo" type="AcctInfo_Type" minOccurs="0"/>
        <xsd:element name="TaxRate" type="TaxRate_Type" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="paymentQlf" type="paymentQlf_Type" use="required"/>
</xsd:complexType>
```

SchemaFragment 3

The payment qualifier (paymentQlf) is the attribute which distinguishes gross, net etc. and has to be one of gip (gross income paid), nip (net income paid), twh (tax withheld), and trf (tax refunded). The tax rate (TaxRate) can optionally be given for any payment item. Tax rates have to be entered as decimal numbers with a total of four digits, two before and two after the decimal point. The date of the payment can be added to any of the items and should designate the day specific to the particular payment, e.g. the refund. Monetary amounts in STF are always qualified by an attribute currCode which is to give the ISO 4217 currency code relevant for the number in the MonAmnt element. For cases where the account into which the payment went matters (and is known) there is a field AcctInfo available, which looks like this:

```
<xsd:complexType name="AcctInfo_Type">
    <xsd:sequence maxOccurs="unbounded">
        <xsd:choice>
         <xsd:element name="IBAN" type="IBAN_Type"/>
         <xsd:element name="OBAN" type="OBAN_Type"/>
         <xsd:element name="ISIN" type="ISIN_Type"/>
```

```
        <xsd:element name="OSIN" type="OSIN_Type"/>
        <xsd:element name="SWIFT" type="SWIFT_Type"/>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>
```

SchemaFragment 4

The IBAN, ISIN, and SWIFT elements shall contain account identifiers as their names say and shall have the standard format of the respective identifiers. OBAN and OSIN stand for “other bank account number” and “other securities identification number” and are to be used for non-standard cases; attributes 'acctNoQlf' and ‘secNoQlf’ respectively have to be used to indicate the kind of such numbers.

5.4 Other information that may be needed to adequately describe the case at hand isn’t part of the element PaymentData but goes into the element OtherInfo. There are no restrictions to the format of this element, which may also have child elements. The sender has to make sure both by using adequate tag names and adding explanations that the receiver is able to understand the sender's intention. As the document is possibly processed automatically there is no guarantee when or even that the content will be recognized by the receiver.

5.5 Identification of the parties involved in the payment is vital for the document to be of any value at all. Therefore a large part of the document content is given to data describing the parties. This is done in a uniform way for all parties, in XML language: all party elements like RecipientBeneficialOwner, ActualPayer etc. are of the same type, Party_Type. So we will have to get acquainted with Party_Type. Here is the broad picture:

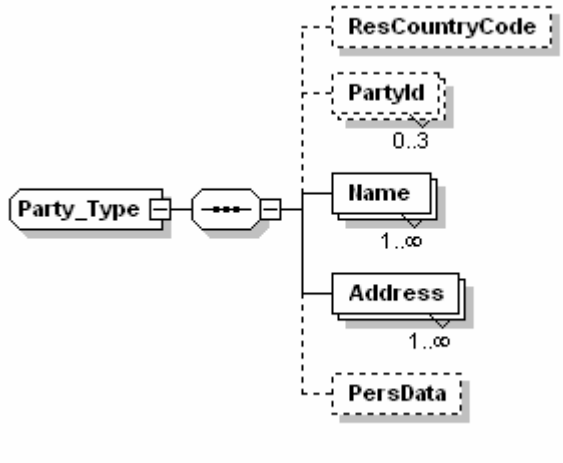

Figure 7: Common structure of all Party elements

There has to be at least one name and one address element inside a party element, but to offer a wider range of descriptive information the number of such elements is not limited. That means that you can add nicknames, names at birth etc. as well as business and other addresses. The nature of names and addresses can be indicated by optional attributes, the admissible values are

for names:

```
<xsd:enumeration value="SMFAliasOrOther"/>
<xsd:enumeration value="indiv"/>
```

```
<xsd:enumeration value="alias"/>
<xsd:enumeration value="nick"/>
<xsd:enumeration value="aka"/>
<xsd:enumeration value="dba"/>
<xsd:enumeration value="legal"/>
<xsd:enumeration value="atbirth"/>
```

and for addresses:

```
<xsd:enumeration value="residentialOrBusiness"/>
<xsd:enumeration value="residential"/>
<xsd:enumeration value="business"/>
<xsd:enumeration value="registeredOffice"/>
```

Most of this will be self-explanatory, let us just mention that aka stands for "also known as", dba for "doing business as". SMFAliasOrOther is an attribute value that should only be used if the document is generated by a bridging program from a SMF record. If there is an entry in the field group "alias or other" in the SMF record (a group within the beneficial owner group which holds all of "aka", "dba" etc.), the bridging program will not be able to decide which kind of name that is and therefore will translate just into "SMFAliasOrOther".

We will go into the detailed structure of names and addresses later.

The residence country (in the relevant time period), to be represented in element ResCountryCode by its ISO 3166 two-byte alpha code, is considered to be a property of the party, not an address, although it is most likely that it will coincide with at least one of the address country codes for this party. To be sufficiently general, the element had to be left optional, even if information about someone with unknown residence country will probably less than helpful.

Another important item of the party description is formal identifiers (to be entered in elements PartyId), for which 3 optional entries are provided. The idea is to give whatever official identification "numbers" are known by the sending country. PartyId elements are declared as shown in Schema Fragment 5:

```
<xsd:complexType name="PartyId_Type">
   <xsd:simpleContent>
    <xsd:extension base="xsd:string">
        <xsd:attribute name="partyIdType" type="partyIdType_Type" use="required"/>
        <xsd:attribute name="issuedBy" type="xsd:string" use="required"/>
    </xsd:extension>
   </xsd:simpleContent>
 </xsd:complexType>
```

SchemaFragment 5

The attribute partyIdType is to distinguish between the kinds of identifiers like Tax Identification Number (TIN), Tax File Number (TFN) and others. To-date TIN, TFN and IdNo are defined as valid entries. It is required to add another attribute (issuedBy) to describe the body that has issued the identifier to the party. In the case of a TIN this should be just the country code of the issuing country, in other cases a non-formalised entry will be adequate.

To even better describe and hopefully identify the party, an optional element PersData can take more information, depending on the type of the party (individual or legal). The content will become clear from Figure 8:

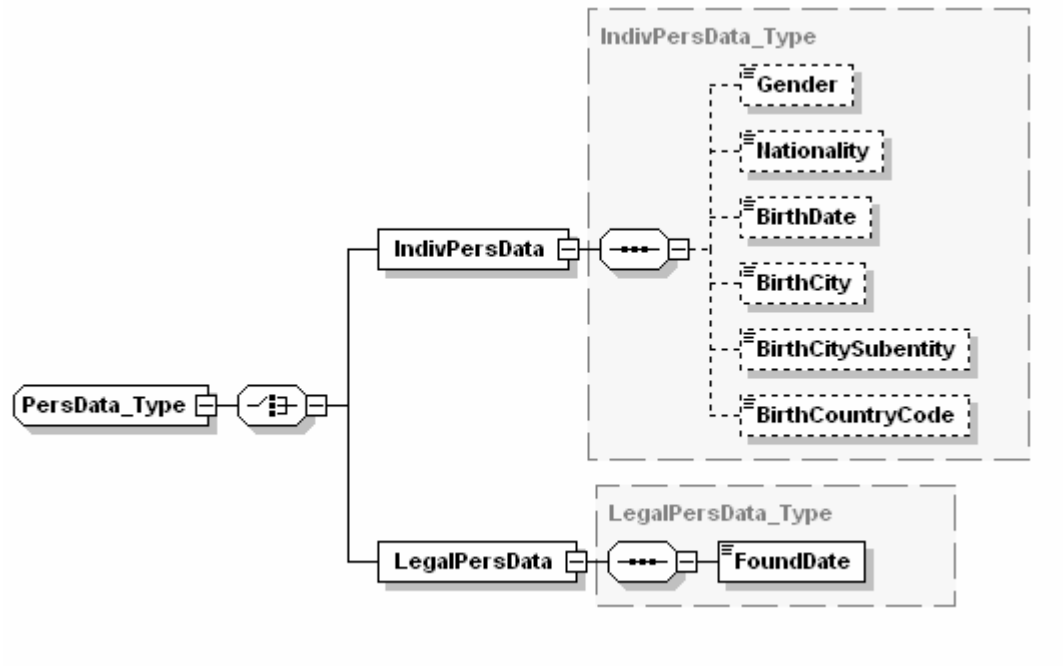

Figure 8: Additional identifying data about a party

In the following paragraphs we will now see how names and addresses are dealt with inside the party structure of STF.

5.6 Names

Here is the broad picture:

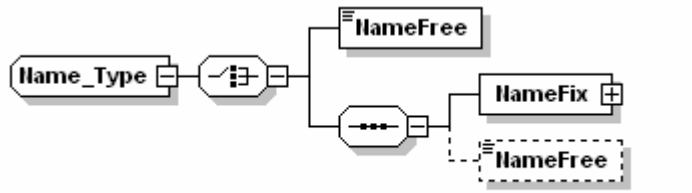

Figure 9: Name structure

It will be noticed that a name can be either a NameFree element, or a NameFix element, or a sequence of both. NameFree will be used to deal with the common situation that it is not really clear for the sending country what are the roles of different particles in a sequence of words that constitute the name of a party. In such cases it may be better to leave it to the receiving country to sort that out, as it may be better acquainted with the name structure of the party. Ideally of course the name of the party is well structured into parts that are identified by the sending country. To serve cases where a structured name (NameFix) can be given, but only with some doubt as to the validity of the break-down into its parts, the sending side may choose to provide a NameFree in addition.
Widely accepted international standards for name structure are only just emerging and for individuals STF has chosen to adhere – although not too closely – to the CIQ standard for names (CIQ: Customer Information Quality, an OASIS family of standards), resulting in the following structure:

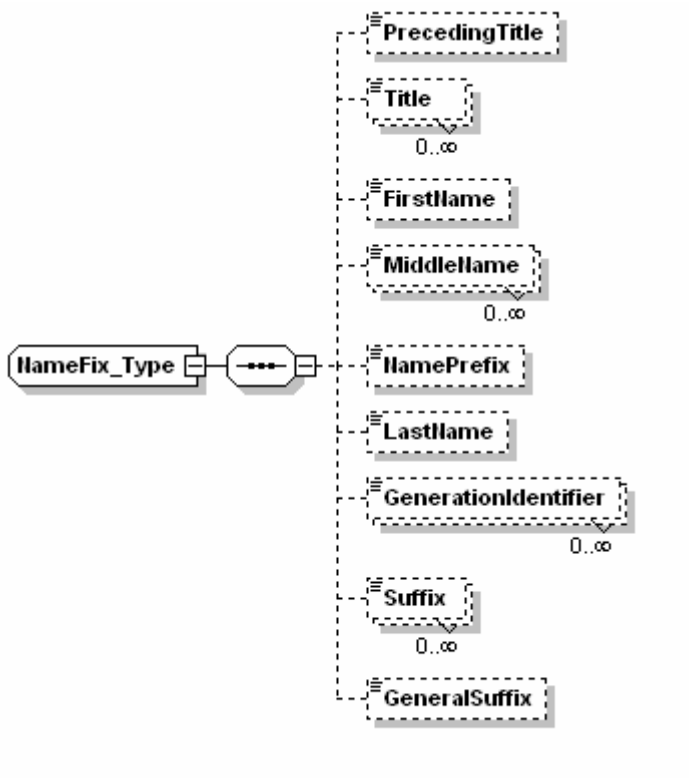

Figure 9: Structured Name in STF

All elements in this structure are optional, as there is no guarantee that a particular one will definitely be present in all cases. Of course there will have to be at least one entry for the name to be useful. Following CIQ, FirstName, MiddleName, NamePrefix, and LastName designate exactly what their names say, that is: the sequence in "normal" usage. The meaning e.g. of the first part of a name may, however, vary from one cultural environment to another. Therefore all of the elements mentioned have to be qualified by an attribute, which is called xnlNameType (xNL is the standard for names in the CIQ family of standards). For the time being there is no predefined set of values for xnlNameType, as also CIQ leaves this to the user. "given Name", "family name" etc. may be values you may want to use.
For legal entities always the free form shall be used for the name; there does not seem to be any useful standardised way of breaking down such names into well-defined parts.

5.7 Addresses

The top level view on addresses is this:

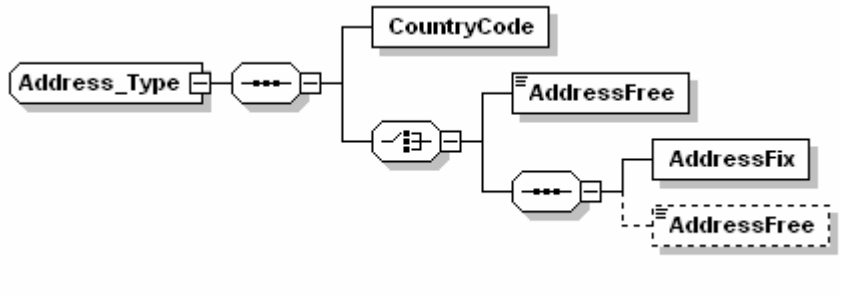

Figure 10: Address structure

Like for names the address can be either an AddressFree element, or an AddressFix element, or a sequence of both. The country code is left outside these structures, as it is be a well-discernable field that never should be imbedded (and hidden) in an unformatted character sequence like in AddressFree.
For addresses more or less the same remarks apply as for names: AddressFree will be used to deal with the common situation that it is not really clear for the sending country what are the roles of different particles in a sequence of words that constitute the address. Also in such cases it may be better to leave it to the receiving country to sort that out, as it may be better acquainted with the address structure of the party. Ideally of course the address of the party is well structured into parts that are identified by the sending country. To serve cases where a structured address (AddressFix) can be given, but only with some doubt as to the validity of the break-down into its parts, the sending side may choose to provide a AddressFree in addition.
Widely accepted international standards for address structure are only just emerging. For STF it has been considered to mimic the CIQ standard for addresses as it did with names. It was found, however, that xAL, the address standard inside CIQ, has gained its extreme flexibility and wide applicability by a degree of complexity that did not seem adequate for STF. This design decision was flagged as "to be monitored", as possible widespread use of xAL in OECD member countries may well suggest reconsidering the design. For the time being, addresses in their fixed format are structured like this in STF:

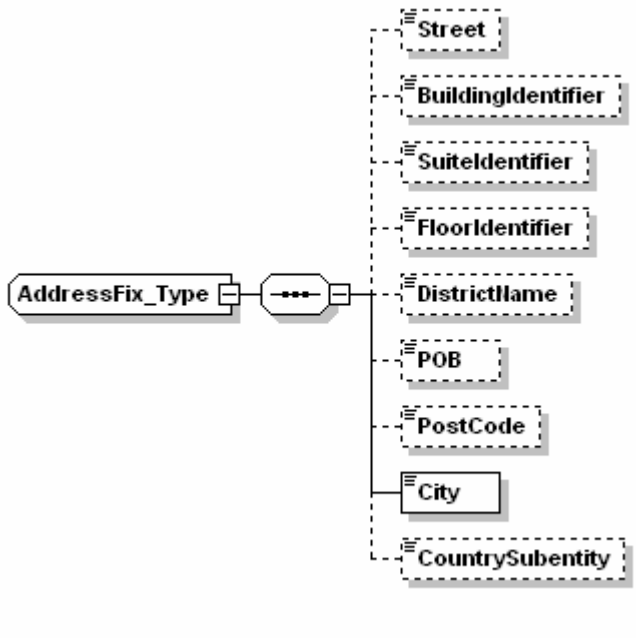

Figure 11: Structured Address in STF

The only mandatory element in this structure is the name of the city. Other address parts shall be given as available.

6. Where is detailed advice for all of the elements and their content to be found?

The central source for all guidance concerning STF is the set of STF XML schemas. Annotations are to be found in the schemas for more or less all of the relevant elements and data types. They are in many cases just replica of the comments to SMF fields in the SMF Manual. As an XML schema is not readily readable by non-IT people and as even for those it may be cumbersome to find a specific piece of documentation in a lengthy schema, comments have been extracted by an automated procedure from the schemas and an "Electronic Manual" has been generated in HTML-format. Users are thus able to find guidance simply by directing their browser to the relevant URL. (---- or a mirror in a country's own web site.)
The Electronic Manual presents the user two columns of information. The left side column is an indented list of the elements and attributes that constitute the most comprehensive STF_OECD

document in accordance with the schema. The right hand side contains the annotations to all data types defined. There are two kinds of interactivity provided: On “mouse-over” at an element in the left hand side hierarchy comments concerning the element are displayed. On “click” over left hand side elements the right hand side is positioned to display comments concerning the data type of the element. (Not all elements possess mouse-over comments nor do all data types possess clickable comments.)
The (left column) structural image of an STF document in the Electronic Manual cannot totally repeat all of the structure information of the schemas. For instance it does not contain an indication whether an element is optional or mandatory. If the complete picture is needed, the reference should be either the schema itself (Appendix 1) or the comprehensive diagram in Appendix 2.
For users with an IT background who want a really comprehensive documentation there is also an automatically generated HTML based documentation which takes in account everything from the schemas but is better suited for reading than the schema code itself. A Word version is attached as Appendix 3.
To avoid even more pages to be generated the country code and currency code lists have been shortened in the printed versions to contain just a few examples.

7. How is coexistence between SMF and STF guaranteed?

Even if XML is the worldwide acknowledged standard for transmission of data between systems and many if not most of OECD member states have an e-Government policy including XML as a preferred standard, countries with a working SMF environment may be reluctant to spend resources for a migration to STF. On the other hand, countries that want to introduce automatic means for international information exchange in taxation just now will probably not want to introduce methods that reflect the IT situation of the 1990s. To adequately deal with that transient situation, bridging programs have been written that transform SMF records into STF_DIRECT documents and vice-versa.

These programs are XSL transformations and have the self-explanatory names

smf2stf
stf2smf

It will certainly be understood that neither OECD nor the authors of the bridging programs can be held liable for any errors of these programs or consequences of such errors. The programs are offered as a support to smooth migration, responsibility for the use of the programs and the data being exchanged stays with the users. Therefore this DISCLAIMER is included in the transformation code:

THIS TRANSFORMATION HAS BEEN WRITTEN AND TESTED WITH CARE. THERE WILL BE, HOWEVER, NO GUARANTEE WHATSOEVER REGARDING ITS CORRECTNESS. ANYONE USING THIS TRANSFORMATION WILL DO THIS UNDER HIS OR HER OWN RESPONSIBILITY AND BEFORE USING IT WILL HAVE TO TEST IT AS CONSIDERED NECESSARY. NO LIABILITY WILL BE ACCEPTED BY OECD, THE OECD TIES GROUP, OR THE AUTHORS OF THIS TRANSFORMATION FOR ANY DIRECT OR INDIRECT DAMAGE THAT MAY RESULT FROM USING THIS TRANSFORMATION. THIS TRANSFORMATION MAY BE USED AND CHANGED FREELY IF AND ONLY IF THESE CONDITIONS ARE ACCEPTED.

It has to be remarked that - mostly due to the slightly enhanced generality of STF compared to SMF - bridging can be done nearly 100%, but not exactly 100%. In the following paragraphs we will explain the issues to be noticed.

I. Bridge from SMF to STF

7.0 Bridging can be done either at the sending or at the receiving side of the SMF file. There is some merit and demerit to either choice. Bridging at the source will enable the sending country to validate the resulting STF file against the schemas and thus filter out any irregularities early in the process. The sender will be better prepared to deal with the file that they

have written themselves and some problems concerning readability may be avoided when an STF file in UTF-8 encoding is transmitted But it also means that the sending country has to keep record of countries that use the STF. As long as the STF has not become the prevailing format it seems best to leave it up to the parties involved in the exchange to decide who will do the bridging

7.1 Bridging is done via an XSL Transformation. As such transformations operate on XML files, a preparatory task is to format the SMF file into an XML file. The transformation program expects the input wrapped by root element tags "SMFFile","/SMFFile" and the records made into "Record" elements, i.e. every record has to be surrounded by "Record", "/Record" tags. Also depending on the XSL transformation procedure a processing instruction

```
<?xml-stylesheet type="text/xsl" href="pppp\smf2stf01.2.xsl"?>
```

may have to be added at the beginning, with pppp to be replaced by the path to the transformation file. It may also be the case that code transformation from mainframe encoding – preferably to UTF-8 – has to be done prior to the XSL. These preparations will be rather straightforward but have to be done according to the individual situation at the member state's site and the SMF file at hand and are therefore not supported by the OECD bridging system.

7.2 SMF files do not have an equivalent to the STF message header "MessageSpec". Therefore there is no source in such files from which the MessageSpec element could be automatically generated from. It is therefore the duty of the person preparing the bridging program's execution to enter the relevant data into the XSL code before the transformation is done. Here is the part of the XSL that has to be adapted:

```
<!-- ********************************************************************* -->
<!-- *************   To be edited before Transformation ************** -->
        <MessageSpec>
         <SendingCountry>Country Code</SendingCountry>
         <ReceivingCountry>Country Code</ReceivingCountry>
         <Warning>Legal Information</Warning>
         <Contact>Who to contact for this message</Contact>
         <MessageRefId></MessageRefId>   <!-- recommendation: leave void  -->
         <TaxYearList>list of tax year ends in form: 2002-12-31</TaxYearList>
        </MessageSpec>
<!-- ********************************************************************* -->
<!-- ********************************************************************* -->
```

7.3 In SMF, referencing records that were sent before (for correction or repetition cases) is done on the record level only, there is no message (file) identifier in SMF. Therefore for STF files generated from SMF, matching between records cannot be based on the combination of message identifier and document identifier. It is therefore recommended not to attribute message identifiers to STF messages generated from SMF files. (The empty element MessadeRefId has to stay there in order to make the document valid.) All matching should thus remain unaffected, though it will not be possible to refer to the message itself by an unambiguous identifier.

7.4 If for the beneficial owner in the SMF record something is entered in the field group "Alias Or Other", a "Name" child element is generated for the STF "RecipientBeneficialOwner" element with the value "SMFAliasOrOther" for the "nameType" attribute.

7.5 Any entries in the SMF field group "In Care Of Person" will be lost. (It seems that nobody has ever made use of this feature in SMF.)

7.6 If in the SMF record there are erroneously for name or address type other entries than '0' (for fixed format) and '1' (for free format) the bridging program will assume the content of the following field(s) to be in free format and transform accordingly.

7.7 If in SMF a gender of "M" (male) or "F" (female) is given for the beneficial owner, there will be a child element "PersData" for the beneficial owner element in STF with that information and – if

present in SMF – information about the city of birth. The entry "N" (non individual) will result in *no* PersData element, as there had to be mandatory content, which is, however, not available in SMF.

II. Bridge from STF to SMF

7.8 An STF document can supply multiple party identifiers for all parties, they can be TINs, but can also other kinds of identifiers. SMF is only supposed to have TINs as identifiers for parties, so any other identifiers in an STF document will be lost by the bridging transformation. For all parties SMF has two TIN fields along with the respective fields for country codes designating the issuing state of the TINs. There is a special situation for the beneficial owner party, as here SMF explicitly asks for the first TIN (and country code) to belong to the residence country and the second to the source country. The bridging program does the following:
- if the element RecipientBeneficialOwner contains a ResCountryCode element, the first TIN field of the bridging result will only contain an entry if there is a TIN PartyId element for the beneficial owner with this country code as the issuedBy attribute. The second TIN field of the result record will contain the data from another TIN PartyId element for the beneficial owner (if any), but there is no test executed whether this will belong to the source country;
- if on the other hand the element RecipientBeneficialOwner does not contain a ResCountryCode element (which is optional in STF), the TIN fields of the result will just contain the data from any two TIN PartyId elements (as far as existent in the STF document).

7.9 An STF document can contain multiple names for all parties. SMF can have two names (including "alias or other") for the recipient beneficial owner party and only one name for the other parties. In bridging, the main name field for the beneficial owner party is filled with the first STF name found which has no nameType attribute or where the nameType attribute is "legal" or "individual"; it is left blank if no such element exists (that is, all Name elements present are nameType-d as "aka", "dba" etc.). Name elements exceeding the number of two (for the beneficial owner) or one (for all other parties) are lost by bridging, with the one exception of a Name element with nameType attribute "at birth": atbirth-names are appended to the name inside the main name field with the addition of "at birth". Also in the case that both a free form and a fixed form name are given one (the fixed form) is lost.

7.10 An STF document can contain PaymentDate elements within every Payment element. There is only one field for a payment date in SMF. This field will be filled from the Payment element with 'gip' (gross income paid) as value for the paymentQlf attribute, if such element exists and has a PaymentDate (which is optional for all payments). If this does not result in filling the field, the next source to look for the payment date will be the 'nip' (net income paid) Payment element. If neither gip nor nip has dates, the date field in the SMF record will be left blank.

8. Examples of Elements and Messages

8.1 Examples of the simplest and the most complete MessageSpec element

```
<MessageSpec>
        <Warning>Only to be used in conformance with our Agreement</Warning>
        <Contact>Rosalie Sender mailto: Rosalie@sender.gov.de</Contact>
        <MessageRefId>123123</MessageRefId>
        <TaxYearList>2004-12-31</TaxYearList>
</MessageSpec>
```

```
<MessageSpec>
        <SendingCountry>GB</SendingCountry>
        <ReceivingCountry>US</ReceivingCountry>
        <Warning>Please do not use this</Warning>
```

```
        <Contact>Mark Anthony phone 007 135 791</Contact>
        <MessageRefId>1234567</MessageRefId>
        <TaxYearList>1999-04-05 2002-04-05</TaxYearList>
</MessageSpec>
```

8.2 Examples of a DocSpec element heading a "new" document and of one correcting another that was sent before (document 1000001 in the message belonging to the first MessageSpec above)

```
<DocSpec>
        <DocTypeIndic>1</DocTypeIndic>
        <DocRefId>987654</DocRefId>
</DocSpec>


<DocSpec>
        <DocTypeIndic>2</DocTypeIndic>
        <DocRefId>5656565</DocRefId>
        <CorrMessageRefId>123123</CorrMessageRefId>
        <CorrDocRefId>1000001</CorrDocRefId>
</DocSpec>
```

8.3 Examples of Name elements

Name element in free format belonging to an individual person party

```
<Name nameType="indiv">
        <NameFree>Arndt Liesen</NameFree>
</Name>
```

Name element in free format belonging to a legal entity

```
<Name nameType="legal">
        <NameFree>Arndt Liesen IT consultancy and training Incorporated</NameFree>
</Name>
```

Name element in fixed format belonging to an individual

```
<Name nameType="indiv">
    <NameFix>
        <PrecedingTitle>Her Excellency</PrecedingTitle>
        <Title>Ms</Title>
        <FirstName xnlNameType="Given Name">Mary</FirstName>
        <MiddleName xnlNameType="Middle Initial">R</MiddleName>
        <NamePrefix xnlNameType="Prefix">de</NamePrefix>
        <LastName xnlNameType="Family Name">Smith</LastName>
        <GenerationIdentifier>II</GenerationIdentifier>
        <Suffix>PhD</Suffix>
        <GeneralSuffix>Retired</GeneralSuffix>
    </NameFix>
</Name>
```

8.4 Examples of Address elements

Address element of a business site in Germany in free format

```
<Address legalAddressType="business">
    <CountryCode>DE</CountryCode>
    <AddressFree>Friedhofstrasse 1 53225 Bonn</AddressFree>
</Address>
```

Same address in fixed format

```
<Address legalAddressType="business">
    <CountryCode>DE</CountryCode>
    <AddressFix>
        <Street>Friedhofstrasse 1</Street>
        <PostCode>53619</PostCode>
        <City>Bonn</City>
    </AddressFix>
</Address>
```

Complex residential address in fixed format (example adapted from an OASIS CIQ standard example, describing the Australian address

block 2, RIPPON BUILDING Level 12, Suite 1A
47 Kingston Avenue North, North Ryde, NSW 2113, Australia)

```
<Address legalAddressType="residential”>
    <CountryCode>AU</CountryCode>
    <AddressFix>
        <Street> 47 Kingston Avenue North</Street>
        <BuildingIdentifier>Block 2, Rippon Building</BuildingIdentifier>
        <SuiteIdentifier>Suite 1A</SuiteIdentifier>
        <FloorIdentifier>Level 12</FloorIdentifier>
        <City>North Ryde</City>
        <CountrySubentity>NSW</CountrySubentity>
    </AddressFix>
</Address>
```

8.5 Examples of elements of Party type

A beneficial owner Party element representing an individual person

```
<RecipientBeneficialOwner oecdLegalType="01">
    <ResCountryCode>DE</ResCountryCode>
    <PartyId partyIdType="TIN" issuedBy="DE">777666555</PartyId>
    <Name nameType="indiv">
        <NameFree>Arndt Liesen</NameFree>
    </Name>
    <Address legalAddressType="residential">
        <CountryCode>DE</CountryCode>
        <AddressFix>
            <Street>Mystreet 77</Street>
            <PostCode>77777</PostCode>
            <City>Mycity</City>
        </AddressFix>
    </Address>
    <PersData>
        <IndivPersData>
            <Gender>M</Gender>
            <Nationality>DE</Nationality>
            <BirthDate>1939-04-16</BirthDate>
            <BirthCity>Duisburg - Germany</BirthCity>
        </IndivPersData>
    </PersData>
</RecipientBeneficialOwner>
```

A payer Party element (address example adapted from an OASIS CIQ standard example)

```
<ActualPayer oecdLegalType="02">
    <ResCountryCode>JP</ResCountryCode>
    <PartyId partyIdType="TFN" issuedBy="JP Tax Authority">1234567</PartyId>
    <Name nameType="legal">
        <NameFree>The very big Payer of Income from Interest Inc.</NameFree>
    </Name>
    <Address legalAddressType="residentialOrBusiness">
        <CountryCode>JP</CountryCode>
        <AddressFree> Japan 530-0001 Osaka Prefecture Oasaka City North Ku Plum Rice Field
```

```
    1-2-2 the 2nd Building before the Oasaka Station </AddressFree>
    </Address>
</ActualPayer>
```

8.6 Examples of PaymentData elements

A gross interest payment (OECD payment type 11) of EUR 2000 was effected in tax year 2003

```
<PaymentData>
    <TaxYearEnd>2003-12-31</TaxYearEnd>
    <OECDPaymentType >11</OECDPaymentType>
    <Payment paymentQlf="gip">
        <MonAmnt currCode="EUR">2000</MonAmnt>
    </Payment>
</PaymentData>
```

In the tax year ending on April 5 2001 these payments were effected, qualified as "Director's Fees" (OECD payment type 16), but more precisely qualified by a country specific payment type of "P47a" according to some classification scheme called "UK2001":
- gross payment of 4000 pounds
- reduced to net payment of 2000 pounds
- followed by a tax refund of 1000 pounds.

```
<PaymentData>
    <TaxYearEnd>2001-04-05</TaxYearEnd>
    <OECDPaymentType>16</OECDPaymentType>
    <SpecificPaymentType specificPaymentTypeQlf="UK2001">P47a</SpecificPaymentType>
    <Payment paymentQlf="gip">
        <PaymentDate>2000-06-31</PaymentDate>
        <MonAmnt currCode="GBP">4000</MonAmnt>
    </Payment>
    <Payment paymentQlf="nip">
        <PaymentDate>2000-06-31</PaymentDate>
        <MonAmnt currCode="GBP">2000</MonAmnt>
    </Payment>
    <Payment paymentQlf="trf">
        <PaymentDate>2000-09-10</PaymentDate>
        <MonAmnt currCode="GBP">1000</MonAmnt>
    </Payment>
</PaymentData>
```

For this the schema file for country specific payment codes, countryspecifictypes_v1.xsd, would have had to be edited like this:

```
…
<xsd:complexType name="SpecificPaymentType_Type">
        <xsd:annotation>
            <xsd:documentation>Type for explanation of a payment by a code that is specific for the UK (version
of 2001)
            special codes used are:
    - S20    interest from extremely large deposits
    - …
    - P47a  fees for directors of chains of nightclubs
    - ….

</xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="SpecifPaymentType_Type ">

             <xsd:attribute name="SpecifPaymentTypeQlf" type="xsd:string" fixed="UK2001"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:simpleType name=" SpecifPaymentType_Type ">
```

```
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="S20"/>
            ....
            <xsd:enumeration value="P47a"/>
            ....
        </xsd:restriction>
    </xsd:simpleType>
```

8.7 A complete message containing two documents for different tax years, one a correction to another document assumed to be sent before

```
<STF_OECD xmlns="urn:oecd:ties:stf:v1"  version="1.0">
    <MessageSpec>
        <SendingCountry>US</SendingCountry>
        <ReceivingCountry>DE</ReceivingCountry>
        <Warning>Only to be used in conformance with our Agreement</Warning>
        <Contact>Rosalie Sender mailto: Rosalie@sender.gov.us</Contact>
        <MessageRefId>US20023-4</MessageRefId>
        <TaxYearList>2003-12-31 2002-12-31</TaxYearList>
    </MessageSpec>
    <STF_DIRECT version="1.0">
        <DocSpec>
            <DocTypeIndic>1</DocTypeIndic>
            <DocRefId>987654</DocRefId>
        </DocSpec>
        <RecipientBeneficialOwner oecdLegalType="01">
            <ResCountryCode>DE</ResCountryCode>
            <PartyId partyIdType="TFN" issuedBy="DE">32/001/47133</PartyId>
            <PartyId partyIdType="TIN" issuedBy="US">123456433</PartyId>
            <Name nameType="indiv">
             <NameFix>
                <PrecedingTitle>Her Excellency</PrecedingTitle>
                <Title>Ms</Title>
                <FirstName xnlNameType="Given Name">Mary</FirstName>
                <MiddleName xnlNameType="Middle Initial">R</MiddleName>
                <NamePrefix xnlNameType="Prefix">de</NamePrefix>
                <LastName xnlNameType="Family Name">Smith</LastName>
                <GenerationIdentifier>II</GenerationIdentifier>
                <Suffix>PhD</Suffix>
                <GeneralSuffix>Retired</GeneralSuffix>
             </NameFix>
            </Name>
            <Name nameType="aka">
             <NameFree>Mary the Belle</NameFree>
            </Name>
            <Name nameType="atbirth">
             <NameFree>Marie Dupont</NameFree>
            </Name>
            <Address legalAddressType="residential">
             <CountryCode>DE</CountryCode>
             <AddressFix>
                <Street>Friedhofstrasse 1</Street>
                <PostCode>53225</PostCode>
                <City>Bonn</City>
             </AddressFix>
            </Address>
            <PersData>
             <IndivPersData>
                <Gender>F</Gender>
                <Nationality>FR</Nationality>
                <BirthDate>1937-08-13</BirthDate>
                <BirthCity>Paris</BirthCity>
                <BirthCitySubentity>Montmartre</BirthCitySubentity>
                <BirthCountryCode>FR</BirthCountryCode>
             </IndivPersData>
            </PersData>
        </RecipientBeneficialOwner>
        <RecipientAgentOrIntermediary oecdLegalType="01">
            <ResCountryCode>DE</ResCountryCode>
```

```
            <Name nameType="legal">
             <NameFree>The Mary the Belle Trust</NameFree>
            </Name>
            <Address legalAddressType="business">
             <CountryCode>DE</CountryCode>
             <AddressFree>53221 Bonn</AddressFree>
            </Address>
        </RecipientAgentOrIntermediary>
        <ActualPayer oecdLegalType="02">
            <ResCountryCode>US</ResCountryCode>
            <PartyId partyIdType="TIN" issuedBy="US">99999999</PartyId>
            <Name nameType="legal">
             <NameFree>Grey Dancers Great Performances</NameFree>
            </Name>
            <Address legalAddressType="business">
             <CountryCode>US</CountryCode>
             <AddressFix>
                 <Street>100 Broadway</Street>
                 <City>NewYork</City>
                 <CountrySubentity>NY</CountrySubentity>
             </AddressFix>
            </Address>
        </ActualPayer>
        <PaymentData>
            <TaxYearEnd>2003-12-31</TaxYearEnd>
            <OECDPaymentType >17</OECDPaymentType>
            <Payment paymentQlf="gip">
             <PaymentDate>2003-07-06</PaymentDate>
             <MonAmnt currCode="USD">7100</MonAmnt>
            </Payment>
        </PaymentData>
        <OtherInfo>Please report back on matching with a real person</OtherInfo>
    </STF_DIRECT>
    <STF_DIRECT version="1.0">
        <DocSpec>
            <DocTypeIndic>2</DocTypeIndic>
            <DocRefId>564534</DocRefId>
            <CorrMessageRefId>US10001-7</CorrMessageRefId>
            <CorrDocRefId>561212</CorrDocRefId>
        </DocSpec>
        <RecipientBeneficialOwner oecdLegalType="03">
            <ResCountryCode>DE</ResCountryCode>
            <Name nameType="legal">
             <NameFree>The Big Earners Partnership</NameFree>
            </Name>
            <Address legalAddressType="residentialOrBusiness">
             <CountryCode>DE</CountryCode>
             <AddressFree>Somewhere in Frankkfurt, Germany</AddressFree>
            </Address>
        </RecipientBeneficialOwner>
        <PayerAgentOrIntermediary oecdLegalType="04">
            <ResCountryCode>US</ResCountryCode>
            <PartyId partyIdType="TIN" issuedBy="US">124534</PartyId>
            <Name nameType="legal">
             <NameFree>First Banking for Nothing</NameFree>
            </Name>
            <Address legalAddressType="unspecified">
             <CountryCode>US</CountryCode>
             <AddressFree>77 Gold Avenue, Las Vegas, Nevada</AddressFree>
            </Address>
            <PersData>
             <LegalPersData>
                 <FoundDate>2002-01-01</FoundDate>
             </LegalPersData>
            </PersData>
        </PayerAgentOrIntermediary>
        <PaymentData>
            <TaxYearEnd>2002-12-31</TaxYearEnd>
            <OECDPaymentType>11</OECDPaymentType>
            <SpecificPaymentType specificPaymentTypeQlf="US special">11-11</SpecificPaymentType>
            <Payment paymentQlf="gip">
             <PaymentDate>2002-01-02</PaymentDate>
             <MonAmnt currCode="EUR">900000001</MonAmnt>
```

```
            <TaxRate>30.5</TaxRate>
           </Payment>
           <Payment paymentQlf="trf">
            <PaymentDate>2003-03-15</PaymentDate>
            <MonAmnt currCode="USD">100000000</MonAmnt>
           </Payment>
         </PaymentData>
         <OtherInfo>US-special income type 11-11 is interest from doubtful source</OtherInfo>
    </STF_DIRECT>
</STF_OECD>
```

9. What artefacts are available for STF?

All of the following is available in electronic form at (URL to be inserted).

9.1 This introductory manual and its appendices

STFexplained.doc
STFexpl-app1-schemas.doc
STFexpl-app2-diagram.png
STFexpl-app3-tecdocu.doc

9.2 The STF schema files

stfdirect-1.0.xsd
stftypes-1.0.xsd
isotypes_v1.xsd
oecdtypes_v1.xsd
specifictypes_v1.xsd

9.3 The bridging programs (XSL stylesheets)

stf2smf-1.0.xsl
smf2stf-1.0.xsl

9.4 The electronic manual in html (hierarchic structure with links to annotations)

STFmanual.html (the main document presenting three frames)
STFhead.html
STFstru.html
STFdoc.html

9.5 The complete generated documentation for the schema-system in html

stf-1.0-generatedDocu.html
stf-1.0-generatedDocu_pxx.png (93 png-files for use by stf-1.0-generatedDocu.html)

BRIDGING PROGRAMME FROM SMF TO STF

```
<?xml version="1.0" encoding="UTF-8"?>
<!--  DISCLAIMER:
THIS TRANSFORMATION HAS BEEN WRITTEN AND TESTED WITH CARE. THERE WILL BE,
HOWEVER, NO GUARANTEE WHATSOEVER REGARDING ITS CORRECTNESS. ANYONE USING THIS
TRANSFORMATION WILL DO THIS UNDER HIS OR HER OWN RESPONSIBILITY AND BEFORE
USING IT WILL HAVE TO TEST IT  AS CONSIDERED NECESSARY. NO LIABILITY WILL BE
ACCEPTED BY OECD, THE OECD TIES GROUP, OR THE AUTHORS OF THIS TRANSFORMATION
FOR  ANY  DIRECT  OR  INDIRECT  DAMAGE  THAT  MAY  RESULT  FROM  USING  THIS
TRANSFORMATION. THIS TRANSFORMATION MAY BE USED AND CHANGED FREELY IF AND ONLY
IF THESE CONDITIONS ARE ACCEPTED. -->
<xsl:stylesheet                                                    version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
       <xsl:output method="xml"/>
       <!--
******************************************************************** -->
       <!--
******************************************************************** -->
       <!--                                                           **********
********** -->
       <!--   **********                                        Main   Template
********** -->
       <!--                                                           **********
********** -->
       <!--
******************************************************************** -->
       <!--
******************************************************************** -->
       <xsl:template match="/">
          <STF_OECD                                  xmlns="urn:oecd:ties:stf:v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
             <!--
******************************************************************** -->
             <!-- *************   To be edited before Transformation **********
-->
             <MessageSpec>
               <SendingCountry>US</SendingCountry>
               <ReceivingCountry>DE</ReceivingCountry>
               <Warning>Legal Information</Warning>
               <Contact>Who to contact for this message</Contact>
               <MessageRefId/>
               <!-- recommendation: leave void -->
               <TaxYearList> 2002-12-31</TaxYearList>
               <!-- list of tax year ends in this format -->
             </MessageSpec>
```

```
            <!--
********************************************************************** -->
            <!--
********************************************************************** -->
            <xsl:for-each select="SMFFile/Record">
              <xsl:call-template name="record"/>
            </xsl:for-each>
          </STF_OECD>
       </xsl:template>
       <!--
********************************************************************** -->
       <!--
********************************************************************** -->
       <!--                                                          **********
********** -->
       <!--   **********                                        Record   Template
********** -->
       <!--                                                          **********
********** -->
       <!--
********************************************************************** -->
       <!--
********************************************************************** -->
       <xsl:template name="record">
          <STF_DIRECT xmlns="urn:oecd:ties:stf:v1">
            <!--    ****************                  Document    Specification
******************** -->
            <DocSpec>
              <DocTypeIndic>
                 <xsl:value-of select="substring(.,1,1)"/>
              </DocTypeIndic>
              <DocRefId>
                 <xsl:value-of        select="normalize-space(substring(.,2411,
70))"/>
              </DocRefId>
              <xsl:if test="substring(.,1,1)!='1' ">
                 <CorrDocRefId>
                    <xsl:value-of     select="normalize-space(substring(.,2481,
70))"/>
                 </CorrDocRefId>
              </xsl:if>
            </DocSpec>
            <!--    ***************                  RecipientBeneficialOwner
******************** -->
            <RecipientBeneficialOwner oecdLegalType="{substring(.,46,2)}">
              <xsl:if test="substring(.,2,2)!='  '">
                 <ResCountryCode>
                    <xsl:value-of select="substring(.,2,2)"/>
                 </ResCountryCode>
              </xsl:if>
              <xsl:if test="substring(.,4,20)!='                    '">
                 <PartyId partyIdType="TIN" issuedBy="{substring(.,2,2)}">
                    <xsl:value-of        select="normalize-space(substring(.,4,
20))"/>
                 </PartyId>
              </xsl:if>
              <xsl:if test="substring(.,26,20)!='                    '">
```

```
                  <PartyId partyIdType="TIN" issuedBy="{substring(.,24,2)}">
                     <xsl:value-of       select="normalize-space(substring(.,26,
20))"/>
                  </PartyId>
               </xsl:if>
               <Name>
                  <xsl:variable name="RecType">
                     <xsl:value-of select="substring(.,46,2)"/>
                  </xsl:variable>
                  <xsl:if test="$RecType=01">
                     <xsl:attribute name="nameType">indiv</xsl:attribute>
                  </xsl:if>
                  <xsl:if test="$RecType='02' or $RecType='03' or $RecType='04'
or $RecType='05'">
                     <xsl:attribute name="nameType">legal</xsl:attribute>
                  </xsl:if>
                  <xsl:call-template name="stfname">
                     <xsl:with-param name="begin">56</xsl:with-param>
                  </xsl:call-template>
               </Name>
               <xsl:if test="substring(.,340,1)!=' '">
                  <Name nameType="SMFAliasOrOther">
                     <xsl:call-template name="stfname">
                        <xsl:with-param name="begin">340</xsl:with-param>
                     </xsl:call-template>
                  </Name>
               </xsl:if>
               <Address>
                  <xsl:attribute                 name="legalAddressType"><xsl:if
test="substring(.,762,1)=0">residentialOrBusiness</xsl:if><xsl:if
test="substring(.,762,1)=1">registeredOffice</xsl:if><xsl:if
test="substring(.,762,1)=2">unspecified</xsl:if></xsl:attribute>
                  <xsl:call-template name="stfaddress">
                     <xsl:with-param name="begin">763</xsl:with-param>
                  </xsl:call-template>
               </Address>
               <xsl:if test="substring(.,915,1)!=' '">
                  <Address>
                     <xsl:attribute              name="legalAddressType"><xsl:if
test="substring(.,915,1)=0">residentialOrBusiness</xsl:if><xsl:if
test="substring(.,915,1)=1">registeredOffice</xsl:if><xsl:if
test="substring(.,915,1)=2">unspecified</xsl:if></xsl:attribute>
                     <xsl:call-template name="stfaddress">
                        <xsl:with-param name="begin">916</xsl:with-param>
                     </xsl:call-template>
                  </Address>
               </xsl:if>
               <xsl:if             test="substring(.,267,1)='F'             or
substring(.,267,1)='M'">
                  <PersData>
                     <IndivPersData>
                        <Gender>
                           <xsl:value-of select="substring(.,267,1)"/>
                        </Gender>
                        <xsl:if test="substring(.,48,8)!='        '">
                           <BirthDate>
                              <xsl:call-template name="stfdate">
```

```
                              <xsl:with-param      name="begin">48</xsl:with-
param>
                           </xsl:call-template>
                        </BirthDate>
                     </xsl:if>
                     <xsl:if                      test="substring(.,268,35)!='
'">
                        <BirthCity>
                           <xsl:value-of                    select="normalize-
space(substring(.,268, 35))"/>
                        </BirthCity>
                     </xsl:if>
                     <xsl:if                      test="substring(.,303,35)!='
'">
                        <BirthCitySubentity>
                           <xsl:value-of                    select="normalize-
space(substring(.,303, 35))"/>
                        </BirthCitySubentity>
                     </xsl:if>
                     <xsl:if test="substring(.,338,2)!='  '">
                        <BirthCountryCode>
                           <xsl:value-of select="substring(.,338,2)"/>
                        </BirthCountryCode>
                     </xsl:if>
                  </IndivPersData>
               </PersData>
            </xsl:if>
         </RecipientBeneficialOwner>
         <!--    ****************              RecipientAgentOrIntermediary
******************** -->
         <xsl:if test="substring(.,1112,1)!=' '">
           <RecipientAgentOrIntermediary oecdLegalType="07">
              <xsl:if                           test="substring(.,1068,22)!='
'">
                 <PartyId                                    partyIdType="TIN"
issuedBy="{substring(.,1068,2)}">
                    <xsl:value-of select="substring(.,1070,20)"/>
                 </PartyId>
              </xsl:if>
              <xsl:if                           test="substring(.,1090,22)!='
'">
                 <PartyId                                    partyIdType="TIN"
issuedBy="{substring(.,1090,2)}">
                    <xsl:value-of                          select="normalize-
space(substring(.,1092,20))"/>
                 </PartyId>
              </xsl:if>
              <Name>
                 <xsl:call-template name="stfname">
                    <xsl:with-param name="begin">1112</xsl:with-param>
                 </xsl:call-template>
              </Name>
              <Address>
                 <xsl:call-template name="stfaddress">
                    <xsl:with-param name="begin">1323</xsl:with-param>
                 </xsl:call-template>
              </Address>
```

```
              </RecipientAgentOrIntermediary>
             </xsl:if>
             <!--     *************************                    ActualPayer
********************************** -->
             <ActualPayer oecdLegalType="{substring(.,1519,2)}">
              <xsl:if test="substring(.,1475,22)!='                      '">
                 <PartyId partyIdType="TIN" issuedBy="{substring(.,1475,2)}">
                    <xsl:value-of                              select="normalize-
space(substring(.,1477,20))"/>
                 </PartyId>
              </xsl:if>
              <xsl:if test="substring(.,1497,22)!='                      '">
                 <PartyId partyIdType="TIN" issuedBy="{substring(.,1497,2)}">
                    <xsl:value-of                              select="normalize-
space(substring(.,1499,20))"/>
                 </PartyId>
              </xsl:if>
              <Name>
                 <xsl:variable name="RecType">
                    <xsl:value-of select="substring(.,1519,2)"/>
                 </xsl:variable>
                 <xsl:if test="$RecType=01">
                    <xsl:attribute name="nameType">indiv</xsl:attribute>
                 </xsl:if>
                 <xsl:if test="$RecType='02' or $RecType='03' or $RecType='04'
or $RecType='05'">
                    <xsl:attribute name="nameType">legal</xsl:attribute>
                 </xsl:if>
                 <xsl:call-template name="stfname">
                    <xsl:with-param name="begin">1521</xsl:with-param>
                 </xsl:call-template>
              </Name>
              <Address>
                 <xsl:call-template name="stfaddress">
                    <xsl:with-param name="begin">1732</xsl:with-param>
                 </xsl:call-template>
              </Address>
             </ActualPayer>
             <!--  *************************         PayerAgentOrIntermediary
********************************** -->
             <xsl:if test="substring(.,1928,1)!=' '">
              <PayerAgentOrIntermediary oecdLegalType="07">
                 <xsl:if                           test="substring(.,1884,22)!='
'">
                    <PartyId                                    partyIdType="TIN"
issuedBy="{substring(.,1884,2)}">
                       <xsl:value-of                           select="normalize-
space(substring(.,1886,20))"/>
                    </PartyId>
                 </xsl:if>
                 <xsl:if                           test="substring(.,1497,22)!='
'">
                    <PartyId                                    partyIdType="TIN"
issuedBy="{substring(.,1906,2)}">
                       <xsl:value-of                           select="normalize-
space(substring(.,1908,20))"/>
                    </PartyId>
```

```
                </xsl:if>
                <Name>
                   <xsl:call-template name="stfname">
                      <xsl:with-param name="begin">1928</xsl:with-param>
                   </xsl:call-template>
                </Name>
                <Address>
                   <xsl:call-template name="stfaddress">
                      <xsl:with-param name="begin">2139</xsl:with-param>
                   </xsl:call-template>
                </Address>
             </PayerAgentOrIntermediary>
          </xsl:if>
          <!--     **************************                    PaymentData
************************************ -->
          <PaymentData>
             <TaxYearEnd>
                <xsl:call-template name="stfdate">
                   <xsl:with-param name="begin">2291</xsl:with-param>
                </xsl:call-template>
             </TaxYearEnd>
             <OECDPaymentType>
                <xsl:value-of select="normalize-space(substring(.,2307,4))"/>
             </OECDPaymentType>
             <xsl:if test="substring(.,2311,4)!='    '">
                <SpecificPaymentType
specificPaymentTypeQlf="{substring(.,24,2)}">
                   <xsl:value-of                          select="normalize-
space(substring(.,2311,4))"/>
                </SpecificPaymentType>
             </xsl:if>
             <xsl:if test="substring(.,2315,3)!='   '">
                <Payment paymentQlf="gip">
                   <xsl:if test="substring(.,2299,8)!='        '">
                      <PaymentDate>
                         <xsl:call-template name="stfdate">
                            <xsl:with-param name="begin">2299</xsl:with-param>
                         </xsl:call-template>
                      </PaymentDate>
                   </xsl:if>
                   <MonAmnt currCode="{substring(.,2315,3)}">
                      <xsl:value-of select="substring(.,2318,18)"/>
                   </MonAmnt>
                </Payment>
             </xsl:if>
             <xsl:if test="substring(.,2336,3)!='   '">
                <Payment paymentQlf="nip">
                   <xsl:if test="substring(.,2299,8)!='        '">
                      <PaymentDate>
                         <xsl:call-template name="stfdate">
                            <xsl:with-param name="begin">2299</xsl:with-param>
                         </xsl:call-template>
                      </PaymentDate>
                   </xsl:if>
                   <MonAmnt currCode="{substring(.,2336,3)}">
                      <xsl:value-of select="substring(.,2339,18)"/>
                   </MonAmnt>
```

```
               </Payment>
            </xsl:if>
            <xsl:if test="substring(.,2357,3)!='   '">
               <Payment paymentQlf="twh">
                  <xsl:if test="substring(.,2299,8)!='        '">
                     <PaymentDate>
                        <xsl:call-template name="stfdate">
                           <xsl:with-param name="begin">2299</xsl:with-param>
                        </xsl:call-template>
                     </PaymentDate>
                  </xsl:if>
                  <MonAmnt currCode="{substring(.,2357,3)}">
                     <xsl:value-of select="substring(.,2360,18)"/>
                  </MonAmnt>
                  <xsl:if test="substring(.,2378,4)!='    '">
                     <TaxRate>
                        <xsl:value-of
select="substring(.,2378,2)"/>.<xsl:value-of select="substring(.,2380,2)"/>
                     </TaxRate>
                  </xsl:if>
               </Payment>
            </xsl:if>
            <xsl:if test="substring(.,2382,3)!='   '">
               <Payment paymentQlf="trf">
                  <xsl:if test="substring(.,2403,8)!='        '">
                     <PaymentDate>
                        <xsl:call-template name="stfdate">
                           <xsl:with-param name="begin">2403</xsl:with-param>
                        </xsl:call-template>
                     </PaymentDate>
                  </xsl:if>
                  <MonAmnt currCode="{substring(.,2382,3)}">
                     <xsl:value-of select="substring(.,2385,18)"/>
                  </MonAmnt>
               </Payment>
            </xsl:if>
         </PaymentData>
         <!--     **************************                    OtherInfo
********************************** -->
         <OtherInfo>
            <xsl:if test="substring(.,2551,10)!='          '">
               <ContentOfSMFFillerGeneralUse>
                  <xsl:value-of select="substring(.,2551,105)"/>
               </ContentOfSMFFillerGeneralUse>
            </xsl:if>
            <xsl:if test="substring(.,2656,10)!='          '">
               <ContentOfSMFFillerSpecificArrangements>
                  <xsl:value-of select="substring(.,2656,105)"/>
               </ContentOfSMFFillerSpecificArrangements>
            </xsl:if>
         </OtherInfo>
      </STF_DIRECT>
   </xsl:template>
   <!--
************************************************************************ -->
   <!--
************************************************************************ -->
```

```
       <!--                                                          **********
********** -->
       <!--   **********                                          Name   Template
********** -->
       <!--                                                          **********
********** -->
       <!--
************************************************************************** -->
       <!--
************************************************************************** -->
       <xsl:template name="stfname">
          <xsl:param name="begin">1</xsl:param>
          <xsl:if test="substring(.,$begin,1)!='0'">
             <NameFree xmlns="urn:oecd:ties:stf:v1">
              <xsl:value-of                                  select="normalize-
space(substring(.,$begin+1,210))"/>
             </NameFree>
          </xsl:if>
          <xsl:if test="substring(.,$begin,1)='0'">
             <NameFix xmlns="urn:oecd:ties:stf:v1">
              <xsl:if                       test="substring(.,$begin+141,35)!='
'">
                 <Title>
                    <xsl:value-of                            select="normalize-
space(substring(.,$begin+141,35))"/>
                 </Title>
              </xsl:if>
              <FirstName>
                 <xsl:attribute name="xnlNameType">GivenName</xsl:attribute>
                 <xsl:value-of                               select="normalize-
space(substring(.,$begin+71,70))"/>
              </FirstName>
              <LastName>
                 <xsl:attribute name="xnlNameType">SurName</xsl:attribute>
                 <xsl:value-of                               select="normalize-
space(substring(.,$begin+1,70))"/>
              </LastName>
              <xsl:if                       test="substring(.,$begin+176,35)!='
'">
                 <Suffix>
                    <xsl:value-of                            select="normalize-
space(substring(.,$begin+176,35))"/>
                 </Suffix>
              </xsl:if>
             </NameFix>
          </xsl:if>
       </xsl:template>
       <!--
************************************************************************** -->
       <!--
************************************************************************** -->
       <!--                                                          **********
********** -->
       <!--   **********                                          Date   Template
********** -->
       <!--                                                          **********
********** -->
```

```
       <!--
************************************************************************ -->
       <!--
************************************************************************ -->
       <xsl:template name="stfdate">
          <xsl:param name="begin">1</xsl:param>
          <xsl:if test="substring(.,$begin+6,2)='  '">
             <xsl:choose>
              <xsl:when test="substring(.,$begin+4,2)='  '">
                 <xsl:attribute name="xsi:type">xsd:gYear</xsl:attribute>
              </xsl:when>
              <xsl:otherwise>
                 <xsl:attribute name="xsi:type">xsd:gYearMonth</xsl:attribute>
              </xsl:otherwise>
             </xsl:choose>
          </xsl:if>
          <xsl:value-of select="substring(.,$begin,4)"/>
          <xsl:if    test="substring(.,$begin+4,2)!='         '">-<xsl:value-of
select="substring(.,$begin+4,2)"/>
          </xsl:if>
          <xsl:if    test="substring(.,$begin+6,2)!='         '">-<xsl:value-of
select="substring(.,$begin+6,2)"/>
          </xsl:if>
       </xsl:template>
       <!--
************************************************************************ -->
       <!--
************************************************************************ -->
       <!--                                                          **********
********** -->
       <!--   **********                                       Address   Template
********** -->
       <!--                                                          **********
********** -->
       <!--
************************************************************************ -->
       <!--
************************************************************************ -->
       <xsl:template name="stfaddress">
          <xsl:param name="begin">1</xsl:param>
          <CountryCode xmlns="urn:oecd:ties:stf:v1">
             <xsl:if test="substring(.,$begin+150,2)='  '">
              <xsl:attribute name="xsi:type">xsd:string</xsl:attribute>
             </xsl:if>
             <xsl:value-of select="substring(.,$begin+150,2)"/>
          </CountryCode>
          <xsl:if test="substring(.,$begin,1)!='0'">
             <AddressFree xmlns="urn:oecd:ties:stf:v1">
              <xsl:value-of                                    select="normalize-
space(substring(.,$begin+1,149))"/>
             </AddressFree>
          </xsl:if>
          <xsl:if test="substring(.,$begin,1)='0'">
             <AddressFix xmlns="urn:oecd:ties:stf:v1">
              <Street>
                 <xsl:value-of                                 select="normalize-
space(substring(.,$begin+1,70))"/>
```

```
               </Street>
               <xsl:if test="substring(.,$begin+141,9)!='         '">
                  <PostCode>
                     <xsl:value-of select="substring(.,$begin+141,9)"/>
                  </PostCode>
               </xsl:if>
               <City>
                  <xsl:value-of                                  select="normalize-
space(substring(.,$begin+71,35))"/>
               </City>
               <xsl:if                         test="substring(.,$begin+106,35)!='
'">
                  <CountrySubentity>
                     <xsl:value-of                               select="normalize-
space(substring(.,$begin+106,35))"/>
                  </CountrySubentity>
               </xsl:if>
            </AddressFix>
         </xsl:if>
      </xsl:template>
</xsl:stylesheet>
```

BRIDGING PROGRAMME FROM STF TO SMF

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
This xslt stylesheet transforms an "SFT_OECD" document into a text file
conforming to SMF record layout.
The SFT_OECD layout here referred is as the XML Schema "stfdirect-0.3.4.xsd"
with types schema files pasted in (standard XSLT cannot process included
files).

Author: Sergio Baldelli - Ministry of Economy and Finance - Italy -
sergio.baldelli@finanze.it
-->
<!--  AL 1104: supported version of stfdirect is stfdirect-1.0.xsd now. -->
<!--   Parts that are surrounded by comments marked "AL 1104" have been
changed, removed or added by Arndt Liesen in November 2004 -->
<!-- AL 1104 DISCLAIMER:
THIS TRANSFORMATION HAS BEEN WRITTEN AND TESTED WITH CARE. THERE WILL BE,
HOWEVER, NO GUARANTEE WHATSOEVER REGARDING ITS CORRECTNESS. ANYONE USING THIS
TRANSFORMATION WILL DO THIS UNDER HIS OR HER OWN RESPONSIBILITY AND BEFORE
USING IT WILL HAVE TO TEST IT  AS CONSIDERED NECESSARY. NO LIABILITY WILL BE
ACCEPTED BY OECD, THE OECD TIES GROUP, OR THE AUTHORS OF THIS TRANSFORMATION
FOR ANY DIRECT OR INDIRECT DAMAGE THAT MAY RESULT FROM USING THIS
TRANSFORMATION. THIS TRANSFORMATION MAY BE USED AND CHANGED FREELY IF AND ONLY
IF THESE CONDITIONS ARE ACCEPTED. -->
<!-- ATTENTION: changes must be done in order to match the latest version of
STF:
          - OtherInfo
          - RBO TINs
          - ...
-->
<!-- AL 1104: The above has been taken care of as far as it seemed necessary
-->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:stf="urn:oecd:ties:stf:v1">
       <xsl:output    method="text"    indent="no"    omit-xml-declaration="yes"
encoding="UTF-8"/>
       <!--  ************* Global variable definition with SMF field length
***************** -->
       <!--          *************          OECD          codes          fields
************************************** -->
       <xsl:variable name="OECDrecipientPayerTypeLen" select="2"/>
       <xsl:variable name="OECDpaymentTypeLen" select="4"/>
       <!--            *************            Identification            fields
************************************** -->
       <xsl:variable name="TINLen" select="20"/>
       <xsl:variable name="genderLen" select="1"/>
       <!--               *************               Name               fields
******************************************** -->
       <xsl:variable name="nameFormatTypeLen" select="1"/>
       <xsl:variable name="keyNameLen" select="70"/>
```

```
       <xsl:variable name="otherNameLen" select="70"/>
       <xsl:variable name="titleLen" select="35"/>
       <xsl:variable name="suffixLen" select="35"/>
       <xsl:variable name="nameFreeLen" select="210"/>
       <!--              **************              Address              fields
****************************************** -->
       <xsl:variable name="addressTypeLen" select="1"/>
       <xsl:variable name="addressFormatTypeLen" select="1"/>
       <xsl:variable name="streetLen" select="70"/>
       <xsl:variable name="cityLen" select="35"/>
       <xsl:variable name="citySubEntityLen" select="35"/>
       <xsl:variable name="countrySubEntityLen" select="35"/>
       <xsl:variable name="postalCodeLen" select="9"/>
       <xsl:variable name="addressFreeLen" select="149"/>
       <xsl:variable name="countryCodeLen" select="2"/>
       <!-- ISO 3166 -->
       <!--        **************        Record        Information        fields
********************************** -->
       <xsl:variable name="dataTypeIndicatorLen" select="1"/>
       <!-- SMF field #1 -->
       <xsl:variable name="referenceLen" select="70"/>
       <!-- SMF field #101 & #102 -->
       <xsl:variable name="fillerLen" select="105"/>
       <!-- SMF field #103 & #104 -->
       <!--               **************               Tax               fields
********************************************** -->
       <xsl:variable name="taxYearEndLen" select="8"/>
       <xsl:variable name="paymentTypeLen" select="4"/>
       <xsl:variable name="currencyCodeLen" select="3"/>
       <!-- ISO 4217 -->
       <xsl:variable name="amountLen" select="18"/>
       <!-- payment and tax withheld or refund -->
       <xsl:variable name="taxRateLen" select="4"/>
       <!--            **************           Miscellaneous           fields
************************************ -->
       <xsl:variable name="dateLen" select="8"/>
       <!-- ISO 8601 -->
       <xsl:variable name="partyLen" select="407"/>
       <!-- Whole party length excepted RBO -->
       <!--          **************          SMF          record          length
************************************** -->
       <xsl:variable name="recLen" select="2760"/>
       <xsl:variable       name="alias">nick       aka       dba       alias
SMFAliasOrOther</xsl:variable>
       <xsl:variable name="mainName">legal individ</xsl:variable>
       <xsl:variable                                    name="blanks"><![CDATA[*
]]></xsl:variable>
       <!-- Used to get white spaces to fill fields up to their length. NB The
star character (*) is necessary to avoid XSLT blank stripping (any other non
whitespace character would be OK). The star is skipped setting 2 the start
position of substring function. -->
       <!--  As above but for debugging    <xsl:variable
name="blanks"><![CDATA[*....'....|....'....|....'....|....'....|....'....|....
'....|....'....|....'....|....'....|....'....|....'....|....'....|....'....|..
..'....|....'....|....'....|....'....|....'....|....'....|....'....|....'....|
....'....|....'....|....'....|....'....|....'....|....'....|....'....|....'...
.|....'....|....'....|....'....|....'....|....'....|....'....|....'....|....'.
```

```
...|....'....|....'....|....'....|....'....|....'....|....'....|....'....|....'....|....
'....|....'....|....'....|....'....|....'....|....'....|....'....|....'....|....'....|..
..'....|....'....|....'....|....'....|....'....|....'....|....'....|....'....|....'....|
....'....|....'....|....'....|....'....|....'....|....'....|....'....|....'....|....'...
.|....'....|....'....|....'....|....'....|....'....|....'....|....'....|....'....|....'.
...|....'....|....'....|....'....|....'....|....'....|....'....|....'....|]]></xsl:varia
ble>-->
       <!-- AL 1104 Begin -->
       <xsl:variable name="zeros">00000000000000000</xsl:variable>
       <!-- AL 1104 End -->
       <xsl:template match="stf:STF_OECD">
          <xsl:for-each select="stf:STF_DIRECT">
             <!-- Processes STF equivalents of SMF records -->
             <xsl:value-of select="stf:DocSpec/stf:DocTypeIndic"/>
             <!-- SMF #1 -->
             <xsl:apply-templates select="stf:RecipientBeneficialOwner"/>
             <xsl:choose>
               <xsl:when test="stf:RecipientAgentOrIntermediary">
                  <xsl:apply-templates
select="stf:RecipientAgentOrIntermediary"/>
                  <!-- SMF #41-55 -->
               </xsl:when>
               <xsl:otherwise>
                  <!-- Fills with blanks -->
                  <xsl:value-of select="substring($blanks, 2, $partyLen)"/>
               </xsl:otherwise>
             </xsl:choose>
             <xsl:choose>
               <xsl:when test="stf:ActualPayer">
                  <xsl:apply-templates select="stf:ActualPayer"/>
                  <!-- SMF APr #56-71 -->
               </xsl:when>
               <xsl:otherwise>
                  <!-- Fills with blanks -->
                  <xsl:value-of          select="substring($blanks,          2,
$OECDrecipientPayerTypeLen + $partyLen)"/>
                  <!-- APr has OECD payer code -->
               </xsl:otherwise>
             </xsl:choose>
             <xsl:choose>
               <xsl:when test="stf:PayerAgentOrIntermediary">
                  <xsl:apply-templates select="stf:PayerAgentOrIntermediary"/>
                  <!-- SMF APr #72-86 -->
               </xsl:when>
               <xsl:otherwise>
                  <!-- Fills with blanks -->
                  <xsl:value-of select="substring($blanks, 2, $partyLen)"/>
               </xsl:otherwise>
             </xsl:choose>
             <xsl:apply-templates select="stf:PaymentData"/>
             <!-- SMF #87-100 -->
             <xsl:value-of             select="concat(stf:DocSpec/stf:DocRefId,
substring($blanks,            2,            $referenceLen            -string-
length(stf:DocSpec/stf:DocRefId)))"/>
             <!-- SMF #101 -->
             <xsl:value-of         select="concat(stf:DocSpec/stf:CorrDocRefId,
substring($blanks,            2,            $referenceLen            -string-
```

```
length(stf:DocSpec/stf:CorrDocRefId)))"/>
            <!-- SMF #102 -->
            <xsl:value-of select="concat(stf:OtherInfo, substring($blanks, 2,
$fillerLen -string-length(stf:OtherInfo)))"/>
            <!-- SMF #103 -->
            <xsl:value-of select="substring($blanks, 2, $fillerLen)"/>
            <!-- SMF #104 -->
         </xsl:for-each>
      </xsl:template>
      <xsl:template match="stf:MessageSpec"/>
      <!-- Avoid default rules -->
      <!--                    **************************                    RBO
********************************** -->
      <xsl:template match="stf:RecipientBeneficialOwner">
         <!-- SMF #2 - #5 -->
         <!-- AL 1104 begin -->
         <xsl:choose>
            <xsl:when test="stf:ResCountryCode">
              <xsl:value-of   select="concat(substring(stf:ResCountryCode,   1,
$countryCodeLen),    substring($blanks,    2,    $countryCodeLen    -    string-
length(substring(stf:ResCountryCode, 1, $countryCodeLen))))"/>
              <xsl:call-template name="TINrbo1">
                 <xsl:with-param name="ResCC">
                    <xsl:value-of select="stf:ResCountryCode"/>
                 </xsl:with-param>
                 <xsl:with-param name="partyIdNo">1</xsl:with-param>
              </xsl:call-template>
              <xsl:call-template name="TINrbo2">
                 <xsl:with-param name="ResCC">
                    <xsl:value-of select="stf:ResCountryCode"/>
                 </xsl:with-param>
                 <xsl:with-param name="partyIdNo">1</xsl:with-param>
              </xsl:call-template>
            </xsl:when>
            <xsl:otherwise>
              <xsl:call-template name="TIN">
                 <xsl:with-param name="partyIdNo">1</xsl:with-param>
                 <xsl:with-param name="TINno">1</xsl:with-param>
              </xsl:call-template>
            </xsl:otherwise>
         </xsl:choose>
         <!-- AL 1104  end -->
         <xsl:value-of  select="concat(@oecdLegalType,  substring($blanks,  2,
$OECDrecipientPayerTypeLen - string-length(@oecdLegalType)))"/>
         <!-- SMF #6 -->
         <!-- STF date format: CCYY-MM-DD, SMF format CCYYMMDD: must strip
dashes -->
         <xsl:value-of
select="concat(translate(stf:PersData/stf:IndivPersData/stf:BirthDate,     '-',
''),        substring($blanks,        2,        $dateLen        -        string-
length(translate(stf:PersData/stf:IndivPersData/stf:BirthDate, '-', ''))))"/>
         <!-- SMF #7-->
         <xsl:call-template name="RBONames">
            <!-- Main RBO name SMF #8-12 -->
            <xsl:with-param name="nameNo" select="1"/>
         </xsl:call-template>
         <!-- AL 23.12.04 -->
```

```
         <xsl:choose>
            <xsl:when test="stf:PersData/stf:IndivPersData/stf:Gender">
              <xsl:value-of
select="concat(stf:PersData/stf:IndivPersData/stf:Gender,    substring($blanks,
2, $genderLen - string-length(stf:PersData/stf:IndivPersData/stf:Gender)))"/>
            </xsl:when>
            <xsl:otherwise>

            <xsl:text>U</xsl:text>
            </xsl:otherwise>
         </xsl:choose>
         <!-- SMF #13-->
         <xsl:value-of
select="concat(stf:PersData/stf:IndivPersData/stf:BirthCity,
substring($blanks,          2,          $cityLen          -          string-
length(stf:PersData/stf:IndivPersData/stf:BirthCity)))"/>
         <!-- SMF #14-->
         <xsl:value-of
select="concat(stf:PersData/stf:IndivPersData/stf:BirthCitySubentity,
substring($blanks,        2,        $citySubEntityLen        -        string-
length(stf:PersData/stf:IndivPersData/stf:BirthCitySubentity)))"/>
         <!-- SMF #15-->
         <xsl:value-of
select="concat(stf:PersData/stf:IndivPersData/stf:BirthCountryCode,
substring($blanks,         2,         $countryCodeLen         -         string-
length(stf:PersData/stf:IndivPersData/stf:BirthCountryCode)))"/>
         <!-- SMF #16-->
         <xsl:call-template name="RBOAliasOrOtherNames">
            <!-- RBO Alias name #17-21 -->
            <xsl:with-param name="nameNo" select="1"/>
         </xsl:call-template>
         <!-- 'In care of' name is not possible in STF: fills with blanks
#22-26  -->
         <xsl:value-of  select="substring($blanks,  2,  $nameFormatTypeLen  +
$nameFreeLen )"/>
         <xsl:apply-templates select="stf:Address[1]"/>
         <!-- SMF #27-33 -->
         <xsl:choose>
            <xsl:when test="stf:Address[2]">
              <!-- RBO may have a second address. #34-40 -->
              <xsl:apply-templates select="stf:Address[2]"/>
            </xsl:when>
            <xsl:otherwise>
              <!-- Fills with blanks -->
              <xsl:value-of  select="substring($blanks,  2,  $addressTypeLen  +
$addressFreeLen + $addressFormatTypeLen + $countryCodeLen)"/>
            </xsl:otherwise>
         </xsl:choose>
      </xsl:template>
      <!--       **************************       RAI,       PAI,       APr
********************************** -->
      <xsl:template match="stf:RecipientAgentOrIntermediary | stf:ActualPayer
| stf:PayerAgentOrIntermediary">
         <xsl:call-template name="TIN">
            <xsl:with-param name="partyIdNo">1</xsl:with-param>
            <xsl:with-param name="TINno">1</xsl:with-param>
         </xsl:call-template>
```

```
          <xsl:if test="name() = 'ActualPayer'">
             <!-- APr has OECD payer code -->
             <xsl:value-of select="concat(@oecdLegalType, substring($blanks, 2,
$OECDrecipientPayerTypeLen - string-length(@oecdLegalType)))"/>
          </xsl:if>
          <xsl:call-template name="NamesDefault">
             <xsl:with-param name="nameNo" select="1"/>
          </xsl:call-template>
          <xsl:apply-templates select="stf:Address[1]"/>
       </xsl:template>
       <!-- ************************** PD ********************************
-->
       <xsl:template match="stf:PaymentData">
          <xsl:value-of  select="concat(translate(stf:TaxYearEnd,  '-',  ''),
substring($blanks, 2, $dateLen - string-length(translate(stf:TaxYearEnd, '-',
'')))))"/>
          <!-- SMF #87-->
          <!-- SMF #88 - In STF, as in SMF, more than payment is possible
there is no sequence
               order as in SMF;  only attribute specifyes what kind of payment
it refers. In TT4
               discussion has been decided that when there is a payment of type
GIP than its
               date is used for mapping into SMF, otherwise any one (1st) is OK
               AL-1104: otherwise the NIP date is choosen, if this is not
present either, the date field will be left blank.
               -->
          <xsl:choose>
             <xsl:when  test="stf:Payment[@paymentQlf  =  'gip']/stf:PaymentDate
">
               <xsl:value-of select="concat(translate(stf:Payment[@paymentQlf =
'gip']/stf:PaymentDate, '-', ''), substring($blanks, 2, $dateLen - string-
length(translate(stf:Payment[@paymentQlf    =    'gip']/stf:PaymentDate,    '-',
'')))))"/>
             </xsl:when>
             <xsl:otherwise>
               <xsl:value-of select="concat(translate(stf:Payment[@paymentQlf =
'nip']/stf:PaymentDate, '-', ''), substring($blanks, 2, $dateLen - string-
length(translate(stf:Payment[@paymentQlf    =    'nip']/stf:PaymentDate,    '-',
'')))))"/>
             </xsl:otherwise>
          </xsl:choose>
          <!-- AL23.12.04 changes   for SMF#89 and SMF#90 due to different
payment data structure -->
          <!-- SMF #89 -->
          <!--  OECD payment type -->
          <xsl:value-of
select="concat(substring(stf:OECDPaymentType,1,$paymentTypeLen),
substring($blanks,         2,         $paymentTypeLen         -         string-
length(substring(stf:OECDPaymentType,1,$paymentTypeLen))))"/>
          <!-- SMF #90 -->
          <xsl:choose>
             <xsl:when test="stf:SpecificPaymentType">
               <!-- country specific payment type -->
               <xsl:value-of
select="concat(substring(stf:SpecificPaymentType,1,$paymentTypeLen),
substring($blanks,         2,         $paymentTypeLen         -         string-
```

```
length(substring(stf:SpecificPaymentType,1,$paymentTypeLen)))"/>
            </xsl:when>
            <xsl:otherwise>
              <!-- No country specific payment type: fills with blank -->
              <xsl:value-of select="substring($blanks, 2, $paymentTypeLen)"/>
            </xsl:otherwise>
          </xsl:choose>
          <xsl:choose>
            <!-- SMF #91-92 -->
            <xsl:when test="stf:Payment[@paymentQlf = 'gip']">
              <xsl:apply-templates select="stf:Payment[@paymentQlf = 'gip']"/>
            </xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="substring($blanks, 2,  $currencyCodeLen +
$amountLen)"/>
            </xsl:otherwise>
          </xsl:choose>
          <!-- SMF #93-94 -->
          <xsl:choose>
            <xsl:when test="stf:Payment[@paymentQlf = 'nip']">
              <xsl:apply-templates select="stf:Payment[@paymentQlf = 'nip']"/>
            </xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="substring($blanks, 2, $currencyCodeLen +
$amountLen)"/>
            </xsl:otherwise>
          </xsl:choose>
          <xsl:choose>
            <!-- SMF #95-96 -->
            <xsl:when test="stf:Payment[@paymentQlf = 'twh']">
              <xsl:apply-templates select="stf:Payment[@paymentQlf = 'twh']"/>
            </xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="substring($blanks, 2, $currencyCodeLen +
$amountLen)"/>
            </xsl:otherwise>
          </xsl:choose>
          <!-- SMF #97 - Must manage format conversion from STF -->
          <xsl:variable name="intDigit" select="concat(substring('0000 ', 1, 2
- string-length(substring-before(stf:Payment/stf:TaxRate, '.'))), substring-
before(stf:Payment/stf:TaxRate, '.'))"/>
          <xsl:variable       name="fracDigit"       select="concat(substring-
after(stf:Payment/stf:TaxRate, '.'), substring('0000 ', 1, 2 - string-
length(substring-after(stf:Payment/stf:TaxRate, '.'))))"/>
          <xsl:value-of select="concat($intDigit, $fracDigit)"/>
          <xsl:choose>
            <!-- SMF #98-99 -->
            <xsl:when test="stf:Payment[@paymentQlf = 'trf']">
              <xsl:apply-templates select="stf:Payment[@paymentQlf = 'trf']"/>
            </xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="substring($blanks, 2, $currencyCodeLen +
$amountLen)"/>
            </xsl:otherwise>
          </xsl:choose>
          <!-- SMF #100 -->
          <xsl:value-of   select="concat(translate(stf:Payment[@paymentQlf   =
```

```
'trf']/stf:PaymentDate, '-', ''), substring($blanks, 2, $dateLen - string-
length(translate(stf:Payment[@paymentQlf    =    'trf']/stf:PaymentDate,    '-',
'')))"/>
       </xsl:template>
       <!--                 **************************                  Payment
********************************** -->
       <xsl:template match="stf:Payment">
          <xsl:value-of select="stf:MonAmnt/@currCode"/>
          <!-- Must manage possible amount with trailing decimal -->
          <xsl:choose>
             <xsl:when test="contains(stf:MonAmnt, '.')">
              <!--    AL    1104    <xsl:value-of    select="concat(substring-
before(stf:MonAmnt,   '.'),   substring($blanks,   2,   $amountLen   -   string-
length(substring-before(stf:MonAmnt, '.'))))"/> -->
              <!-- AL 1104 Begin -->
              <xsl:value-of select="concat( substring($zeros, 2, $amountLen -
string-length(substring-before(stf:MonAmnt,                    '.'))),substring-
before(stf:MonAmnt, '.'))"/>
              <!-- AL 1104 End -->
             </xsl:when>
             <xsl:otherwise>
              <!--    AL    1104    <xsl:value-of    select="concat(stf:MonAmnt,
substring($blanks, 2, $amountLen - string-length(stf:MonAmnt)))"/>
              -->
              <!-- AL 1104 Begin -->
              <xsl:value-of select="concat( substring($zeros, 2, $amountLen -
string-length(stf:MonAmnt)),stf:MonAmnt)"/>
              <!-- AL 1104 End -->
             </xsl:otherwise>
          </xsl:choose>
       </xsl:template>
       <!--                    **************************                    TIN
********************************** -->
       <!--  In SMF only 2 TINs are possible while in STF more than 2 PartyId
may occur.
          Thus for each SMF TIN field it is necessary to scan all the STF
PartyId (starting
          from the related SMF field position) in order to find PartyId of TIN
type. If no such
          type field is filled with blanks
       -->
       <!--           **************************           RBO           TIN
********************************** -->
       <!--  In RBO TINs must be managed in a special way. While in the other
parties the two first
          PartyIds of TIN type are mapped respectively into the 1st and 2nd
TIN fild, in RBO the
          fisical order is not relevant. The PartyId (of type TIN) of which
the issuedBy attribute
          matches the residence country code must mapped in the first SMF TIN
field.
          Refer to TIN template for all common features.
       -->
       <!-- AL 1104 begin special handling for recipient beneficial owner
TINs-->
       <xsl:template name="TINrbo1">
          <xsl:param name="partyIdNo"/>
```

```
         <xsl:param name="ResCC"/>
         <xsl:choose>
            <xsl:when test="stf:PartyId[number($partyIdNo)]">
              <!-- Still PartyId remaining -->
              <xsl:choose>
                 <xsl:when
test="stf:PartyId[number($partyIdNo)]/@partyIdType='TIN'                    and
stf:PartyId[number($partyIdNo)]/@issuedBy=$ResCC">
                    <xsl:value-of
select="concat(stf:PartyId[number($partyIdNo)], substring($blanks, 2, $TINLen
- string-length(stf:PartyId[number($partyIdNo)])))"/>
                 </xsl:when>
                 <xsl:otherwise>
                    <!-- Not a TIN identifier from the residence country -->
                    <xsl:call-template name="TINrbo1">
                       <xsl:with-param  name="partyIdNo"  select="$partyIdNo  +
1"/>
                       <xsl:with-param name="ResCC" select="$ResCC"/>
                    </xsl:call-template>
                 </xsl:otherwise>
              </xsl:choose>
            </xsl:when>
            <xsl:otherwise>
              <!-- no more PartyId elements remaining -->
              <xsl:value-of select="substring($blanks, 2, $TINLen )"/>
            </xsl:otherwise>
         </xsl:choose>
       </xsl:template>
       <xsl:template name="TINrbo2">
         <xsl:param name="partyIdNo"/>
         <xsl:param name="ResCC"/>
         <xsl:choose>
            <xsl:when test="stf:PartyId[number($partyIdNo)]">
              <!-- Still PartyId remaining -->
              <xsl:choose>
                 <xsl:when
test="stf:PartyId[number($partyIdNo)]/@partyIdType='TIN'                    and
stf:PartyId[number($partyIdNo)]/@issuedBy!=$ResCC">
                    <xsl:value-of
select="concat(substring(stf:PartyId[number($partyIdNo)]/@issuedBy,          1,
$countryCodeLen),    substring($blanks,    2,    $countryCodeLen    -    string-
length(substring(stf:PartyId[number($partyIdNo)]/@issuedBy,                  1,
$countryCodeLen))))"/>
                    <xsl:value-of
select="concat(stf:PartyId[number($partyIdNo)], substring($blanks, 2, $TINLen
- string-length(stf:PartyId[number($partyIdNo)])))"/>
                 </xsl:when>
                 <xsl:otherwise>
                    <!-- Not a TIN identifier from other than the residence
country -->
                    <xsl:call-template name="TINrbo2">
                       <xsl:with-param  name="partyIdNo"  select="$partyIdNo  +
1"/>
                       <xsl:with-param name="ResCC" select="$ResCC"/>
                    </xsl:call-template>
                 </xsl:otherwise>
              </xsl:choose>
```

```
            </xsl:when>
            <xsl:otherwise>
              <!-- no more PartyId elements remaining -->
              <xsl:value-of  select="substring($blanks,  2,  $countryCodeLen   +
$TINLen )"/>
            </xsl:otherwise>
          </xsl:choose>
       </xsl:template>
       <!-- AL 1104  end -->
       <!--  ********************* TIN general case ***************** -->
       <xsl:template name="TIN">
          <xsl:param name="partyIdNo"/>
          <xsl:param name="TINno"/>
          <xsl:if test="$TINno &lt;  3">
             <!-- Only 2 TIN field groups in SMF -->
             <xsl:choose>
              <xsl:when test="stf:PartyId[number($partyIdNo)]">
                 <!-- Still PartyId remaining -->
                 <xsl:choose>
                    <xsl:when
test="stf:PartyId[number($partyIdNo)]/@partyIdType='TIN'">
                       <xsl:value-of
select="concat(substring(stf:PartyId[number($partyIdNo)]/@issuedBy,          1,
$countryCodeLen),    substring($blanks,    2,    $countryCodeLen    -    string-
length(substring(stf:PartyId[number($partyIdNo)]/@issuedBy,                  1,
$countryCodeLen))))"/>
                       <xsl:value-of
select="concat(stf:PartyId[number($partyIdNo)],  substring($blanks,  2,  $TINLen
- string-length(stf:PartyId[number($partyIdNo)])))"/>
                       <xsl:call-template name="TIN">
                          <xsl:with-param name="partyIdNo" select="$partyIdNo +
1"/>
                          <xsl:with-param name="TINno" select="$TINno + 1"/>
                       </xsl:call-template>
                    </xsl:when>
                    <xsl:otherwise>
                       <!-- Not a TIN identifier -->
                       <xsl:call-template name="TIN">
                          <xsl:with-param name="partyIdNo" select="$partyIdNo +
1"/>
                          <xsl:with-param name="TINno" select="$TINno"/>
                          <!-- No TIN inc-->
                       </xsl:call-template>
                    </xsl:otherwise>
                 </xsl:choose>
              </xsl:when>
              <xsl:otherwise>
                 <!-- no more PartyId elements remaining -->
                 <xsl:value-of  select="substring($blanks,  2,  $countryCodeLen
+ $TINLen )"/>
                 <xsl:call-template name="TIN">
                    <xsl:with-param name="partyIdNo" select="$partyIdNo + 1"/>
                    <xsl:with-param name="TINno" select="$TINno + 1"/>
                 </xsl:call-template>
              </xsl:otherwise>
            </xsl:choose>
          </xsl:if>
```

```
      </xsl:template>
      <!--              ****************************             NamesDefault
******************************** -->
      <!-- Tests all the Name elements and maps the 1st one with no attribute
"nameType".
         Otherwise call template to test for nameType = "legal" or "indiv".
      -->
      <xsl:template name="NamesDefault">
         <xsl:param name="nameNo"/>
         <xsl:choose>
            <xsl:when test="stf:Name[number($nameNo)]">
              <!-- Still names to scan -->
              <xsl:choose>
                 <xsl:when test="stf:Name[number($nameNo)][not(@nameType)]">
                    <xsl:apply-templates select="stf:Name[number($nameNo)]"/>
                 </xsl:when>
                 <xsl:otherwise>
                    <!-- Call itself on next element -->
                    <xsl:call-template name="NamesDefault">
                       <xsl:with-param name="nameNo" select="$nameNo + 1"/>
                    </xsl:call-template>
                 </xsl:otherwise>
              </xsl:choose>
            </xsl:when>
            <xsl:otherwise>
              <xsl:call-template name="NamesLegalOrIndiv">
                 <xsl:with-param name="nameNo" select="1"/>
              </xsl:call-template>
            </xsl:otherwise>
         </xsl:choose>
      </xsl:template>
      <!--            *************************            NamesLegalOrIndiv
************************* -->
      <!--  Tests all the Name elements and maps the 1st one with
            attribute = "legal" or "indiv".
            Otherwise fills with blanks.
      -->
      <xsl:template name="NamesLegalOrIndiv">
         <xsl:param name="nameNo"/>
         <xsl:choose>
            <xsl:when test="stf:Name[number($nameNo)]">
              <!-- Still names to scan -->
              <xsl:choose>
                 <xsl:when     test="stf:Name[number($nameNo)]/@nameType     =
'legal'">
                    <xsl:apply-templates select="stf:Name[number($nameNo)]"/>
                 </xsl:when>
                 <xsl:otherwise>
                    <!-- Call itself on next element -->
                    <xsl:call-template name="NamesLegalOrIndiv">
                       <xsl:with-param name="nameNo" select="$nameNo + 1"/>
                    </xsl:call-template>
                 </xsl:otherwise>
              </xsl:choose>
            </xsl:when>
            <xsl:otherwise>
              <!-- Fills with blanks -->
```

```
              <xsl:value-of select="substring($blanks, 2, $nameFormatTypeLen +
$nameFreeLen )"/>
            </xsl:otherwise>
         </xsl:choose>
      </xsl:template>
      <!--          ************************          RBO          Names
******************************** -->
      <!-- Tests all the Name elements and maps the 1st one for which holds:
         no attribute NameType OR NameType = "legal" OR NameType = "indiv"
      -->
      <xsl:template name="RBONames">
         <xsl:param name="nameNo"/>
         <xsl:choose>
            <xsl:when test="stf:Name[number($nameNo)]">
              <xsl:choose>
                 <xsl:when  test="not(stf:Name[number($nameNo)]/@nameType)  or
(contains($mainName,          stf:Name[number($nameNo)]/@nameType)          and
stf:Name[number($nameNo)]/@nameType != '')">
                    <!-- contain() returns true with void string; in case of
empty attribute -->
                    <xsl:apply-templates select="stf:Name[number($nameNo)]"/>
                 </xsl:when>
                 <xsl:otherwise>
                    <xsl:call-template name="RBONames">
                       <!-- Call itself on next element -->
                       <xsl:with-param name="nameNo" select="$nameNo + 1"/>
                    </xsl:call-template>
                 </xsl:otherwise>
              </xsl:choose>
            </xsl:when>
            <xsl:otherwise>
              <!-- no one Name element (if any) satisfies the test: fills with
blanks -->
              <xsl:value-of select="substring($blanks, 2, $nameFormatTypeLen +
$nameFreeLen )"/>
            </xsl:otherwise>
         </xsl:choose>
      </xsl:template>
      <!--        **********************        RBOAliasOrOther        Names
************************** -->
      <!-- Tests all the Name elements and maps the 1st one for which holds:
         There is a nameType attribute AND its value is not a void string AND
         it is one of the following values: "aka", "dba", "nick", "alias",
"SMFAliasOrOther".
         Otherwise fills with blanks.
      -->
      <xsl:template name="RBOAliasOrOtherNames">
         <xsl:param name="nameNo"/>
         <xsl:choose>
            <xsl:when test="stf:Name[number($nameNo)]">
              <xsl:choose>
                 <xsl:when    test="stf:Name[number($nameNo)]/@nameType    and
stf:Name[number($nameNo)]/@nameType     !=     ''     and     contains($alias,
stf:Name[number($nameNo)]/@nameType)">
                    <xsl:apply-templates select="stf:Name[number($nameNo)]"/>
                 </xsl:when>
                 <xsl:otherwise>
```

```
                  <!-- Call itself on next element -->
                  <xsl:call-template name="RBOAliasOrOtherNames">
                     <xsl:with-param name="nameNo" select="$nameNo + 1"/>
                  </xsl:call-template>
               </xsl:otherwise>
            </xsl:choose>
         </xsl:when>
         <xsl:otherwise>
            <!-- no one Name element (if any) satisfies the test: fills with
blanks -->
            <xsl:value-of select="substring($blanks, 2, $nameFormatTypeLen +
$nameFreeLen )"/>
         </xsl:otherwise>
      </xsl:choose>
   </xsl:template>
   <!--                  ****************************                  Name
******************************** -->
   <xsl:template match="stf:Name">
      <xsl:choose>
         <xsl:when test="stf:NameFree">
            <xsl:text>1</xsl:text>
            <!-- SMF Name format type flag -->
            <xsl:choose>
               <!-- At birth name must not be appended to Alias name -->
               <xsl:when test="contains($alias, @nameType)">
                  <xsl:value-of                select="concat(./stf:NameFree,
substring($blanks, 2, $nameFreeLen - string-length(./stf:NameFree)))"/>
               </xsl:when>
               <xsl:otherwise>
                  <!-- Not an alias name: must manage at birth -->
                  <xsl:call-template name="atbirthFree"/>
               </xsl:otherwise>
            </xsl:choose>
         </xsl:when>
         <xsl:otherwise>
            <!-- Not a free format name -->
            <xsl:call-template name="namefix"/>
         </xsl:otherwise>
      </xsl:choose>
   </xsl:template>
   <!--       *************************       Fix       format       name
************************** -->
   <xsl:template name="namefix">
      <xsl:text>0</xsl:text>
      <!-- SMF Name format type flag -->
      <xsl:choose>
         <!-- At birth name must not be appended to Alias name -->
         <xsl:when test="contains($alias, @nameType)">
            <xsl:value-of          select="concat(.//stf:LastName          ,
substring($blanks, 2, $keyNameLen - string-length(.//stf:LastName)))"/>
         </xsl:when>
         <xsl:otherwise>
            <!-- Not an alias name: must manage at birth -->
            <xsl:call-template name="atbirthFix"/>
         </xsl:otherwise>
      </xsl:choose>
      <!-- Appends STF FirstName and MiddleName into SMF Other Name field
```

```
-->
          <xsl:variable         name="otherNameBuff"         select="normalize-
space(concat(.//stf:FirstName, ' ', .//stf:MiddleName))"/>
          <xsl:value-of  select="concat($otherNameBuff,  substring($blanks,  2,
$otherNameLen - string-length($otherNameBuff)))"/>
          <!-- Appends into SMF Suffix field STF GenerationIdentifier, Suffix
and GeneralSuffix -->
          <xsl:value-of    select="concat(.//stf:Title,    substring($blanks,2,
$titleLen - string-length(.//stf:Title)))"/>
          <xsl:variable          name="suffixBuff"          select="normalize-
space(concat(.//stf:GenerationIdentifier,    '    ',    .//stf:Suffix,    '    ',
.//stf:GeneralSuffix))"/>
          <xsl:value-of    select="concat($suffixBuff,    substring($blanks,2,
$suffixLen - string-length($suffixBuff)))"/>
       </xsl:template>
       <!--     **********************     "at     birth"     (free     form)
************************ -->
       <xsl:template name="atbirthFree">
          <xsl:choose>
             <xsl:when                                              test="following-
sibling::stf:Name[@nameType='atbirth']/stf:NameFree">
               <xsl:variable           name="buff"           select="normalize-
space(concat(./stf:NameFree,      '      at      birth:      ',      following-
sibling::stf:Name[@nameType='atbirth']/stf:NameFree))"/>
               <xsl:value-of    select="concat($buff,    substring($blanks,    2,
$nameFreeLen - string-length($buff)))"/>
             </xsl:when>
             <xsl:when                                              test="preceding-
sibling::stf:Name[@nameType='atbirth']/stf:NameFree">
               <xsl:variable           name="buff"           select="normalize-
space(concat(./stf:NameFree,      '      at      birth:      ',      preceding-
sibling::stf:Name[@nameType='atbirth']/stf:NameFree))"/>
               <xsl:value-of    select="concat($buff,    substring($blanks,    2,
$nameFreeLen - string-length($buff)))"/>
             </xsl:when>
             <xsl:when                                              test="following-
sibling::stf:Name[@nameType='atbirth']/stf:NameFix">
               <xsl:variable           name="buff"           select="normalize-
space(concat(./stf:NameFree,      '      at      birth:      ',      following-
sibling::stf:Name[@nameType='atbirth']/stf:NameFix/stf:LastName))"/>
               <xsl:value-of    select="concat($buff,    substring($blanks,    2,
$nameFreeLen - string-length($buff)))"/>
             </xsl:when>
             <xsl:when                                              test="preceding-
sibling::stf:Name[@nameType='atbirth']/stf:NameFix">
               <xsl:variable           name="buff"           select="normalize-
space(concat(./stf:NameFree,      '      at      birth:      ',      preceding-
sibling::stf:Name[@nameType='atbirth']/stf:NameFix/stf:LastName))"/>
               <xsl:value-of    select="concat($buff,    substring($blanks,    2,
$nameFreeLen - string-length($buff)))"/>
             </xsl:when>
             <xsl:otherwise>
               <!-- No sibling name with at birth attribute: copies the bare
name -->
               <xsl:value-of  select="concat(./stf:NameFree,  substring($blanks,
2, $nameFreeLen - string-length(./stf:NameFree)))"/>
             </xsl:otherwise>
```

```
         </xsl:choose>
      </xsl:template>
      <!--    ***************************    "at    birth"    (fix    form)
************************** -->
      <xsl:template name="atbirthFix">
         <xsl:choose>
            <xsl:when                                              test="following-
sibling::stf:Name[@nameType='atbirth']/stf:NameFree">
              <xsl:variable           name="buff"           select="normalize-
space(concat(./stf:NameFix/stf:LastName,    '    at    birth:    ',    following-
sibling::stf:Name[@nameType='atbirth']/stf:NameFree))"/>
              <xsl:value-of    select="concat($buff,    substring($blanks,    2,
$keyNameLen - string-length($buff)))"/>
            </xsl:when>
            <xsl:when                                              test="preceding-
sibling::stf:Name[@nameType='atbirth']/stf:NameFree">
              <xsl:variable           name="buff"           select="normalize-
space(concat(./stf:NameFix/stf:LastName,    '    at    birth:    ',    preceding-
sibling::stf:Name[@nameType='atbirth']/stf:NameFree))"/>
              <xsl:value-of    select="concat($buff,    substring($blanks,    2,
$keyNameLen - string-length($buff)))"/>
            </xsl:when>
            <xsl:when                                              test="following-
sibling::stf:Name[@nameType='atbirth']/stf:NameFix">
              <xsl:variable           name="buff"           select="normalize-
space(concat(./stf:NameFix/stf:LastName,    '    at    birth:    ',    following-
sibling::stf:Name[@nameType='atbirth']/stf:NameFix/stf:LastName))"/>
              <xsl:value-of    select="concat($buff,    substring($blanks,    2,
$keyNameLen - string-length($buff)))"/>
            </xsl:when>
            <xsl:when                                              test="preceding-
sibling::stf:Name[@nameType='atbirth']/stf:NameFix">
              <xsl:variable           name="buff"           select="normalize-
space(concat(./stf:NameFix/stf:LastName,    '    at    birth:    ',    preceding-
sibling::stf:Name[@nameType='atbirth']/stf:NameFix/stf:LastName))"/>
              <xsl:value-of    select="concat($buff,    substring($blanks,    2,
$keyNameLen - string-length($buff)))"/>
            </xsl:when>
            <xsl:otherwise>
              <!-- No sibling name with at birth attribute: copies the bare
name -->
              <xsl:value-of          select="concat(./stf:NameFix/stf:LastName,
substring($blanks,          2,          $keyNameLen          -          string-
length(./stf:NameFix/stf:LastName)))"/>
            </xsl:otherwise>
         </xsl:choose>
      </xsl:template>
      <!--                  ***************************                Address
********************************** -->
      <!--  In STF there is a choice between free/fix address but, when fix is
chosen it can be            followed by an AddressFree element containing the
same data in one string; thus
         the second child of Address (1st CountryCode) has to be tested in
order to determine
         which format to choose for the SMF sake. Note that if a free format
follow a fix one
         the free is dropped
```

```
       -->
       <xsl:template match="stf:Address">
          <xsl:if test="parent::stf:RecipientBeneficialOwner">
             <!-- Only RBO has "Address type" field -->
             <xsl:choose>
               <xsl:when            test="contains('stf:residentialOrBusiness',
@legalAddressType)">
                  <!-- AL 23.12.04 -->
                  <xsl:text>0</xsl:text>
               </xsl:when>
               <xsl:when                  test="contains('stf:registeredOffice',
@legalAddressType)">
                  <!-- AL 23.12.04 -->
                  <xsl:text>1</xsl:text>
               </xsl:when>
               <xsl:otherwise>
                  <xsl:text>2</xsl:text>
               </xsl:otherwise>
             </xsl:choose>
          </xsl:if>
          <xsl:choose>
             <xsl:when test="*[2] = stf:AddressFree">
               <!-- Second child = AddressFree ? -->
               <xsl:text>1</xsl:text>
               <xsl:value-of                    select="concat(./stf:AddressFree,
substring($blanks, 2, $addressFreeLen - string-length(./stf:AddressFree)))"/>
             </xsl:when>
             <xsl:otherwise>
               <xsl:text>0</xsl:text>
               <xsl:apply-templates select="stf:AddressFix"/>
             </xsl:otherwise>
          </xsl:choose>
          <xsl:value-of   select="concat(stf:CountryCode,   substring($blanks,2,
$countryCodeLen - string-length(stf:CountryCode)))"/>
       </xsl:template>
       <!--      *************************      Fix      format      address
************************* -->
       <xsl:template match="stf:AddressFix">
          <!-- Appends into SMF Street field STF elements: street, building,
suite and floor  -->
          <xsl:variable          name="streetBuff"          select="normalize-
space(concat(.//stf:Street,    '    ',    .//stf:BuildingIdentifier,    '    ',
.//stf:SuiteIdentifier, ' ',  .//stf:FloorIdentifier))"/>
          <xsl:value-of    select="concat($streetBuff,    substring($blanks,2,
$streetLen - string-length($streetBuff)))"/>
          <xsl:value-of select="concat(stf:City, substring($blanks,2, $cityLen
- string-length(stf:City)))"/>
          <xsl:value-of                    select="concat(stf:CountrySubentity,
substring($blanks,2,          $countrySubEntityLen          -          string-
length(stf:CountrySubentity)))"/>
          <xsl:value-of    select="concat(stf:PostCode,    substring($blanks,2,
$postalCodeLen - string-length(stf:PostCode)))"/>
       </xsl:template>
</xsl:stylesheet>
```