eCH-0165 SIARD Format Specification 2.0 (Draft)

Name	SIARD Format Specification
Standard number	eCH-0165
Category	Standard
Maturity level	Tentative
Version	2.0
Status	Draft – Final E-ARK
Approval date	
Issue date	
Replaces standard	
Languages	English (Original)
Authors	Hedi Bruggisser, Thurgau State Archives, hedi.bruggisser@tg.ch Georg Büchler, KOST, georg.buechler@kost.admin.ch Alain Dubois, Valais State Archives, alain.dubois@admin.vs.ch Martin Kaiser, KOST, martin.kaiser@kost.admin.ch Lambert Kansy, Basel Stadt State Archives, lambert.kansy@bs.ch Markus Lischer, Lucerne State Archives, markus.lischer@lu.ch Claire Röthlisberger-Jourdan, KOST, claire.roethlisberger@kost.admin.ch Hartwig Thomas, Enter AG, hartwig.thomas@enterag.ch Andreas Voss, Swiss Federal Archives, andreas.voss@bar.admin.ch
Publisher / distributor	Verein eCH, Mainaustrasse 30, Postfach, 8034 Zürich T 044 388 74 64, F 044 388 71 80 www.ech.ch / info@ech.ch

Summary

This document contains the specification for the SIARD file format version 2.0. SIARD stands for Software Independent Archival of Relational Databases. It is an extension of the standard eCH-0165 for the SIARD Format version 1.0. The format was developed by the Swiss Federal Archives. It is a normative description of a file format for the long-term preservation of relational databases.

The SIARD format is based on standards including the ISO standards Unicode, XML, and SQL:2008, the URI Internet standard, and the industry standard ZIP. The aim of employing internationally recognised standards is to ensure the long-term preservation of, and access to, the widely used relational database model, as well as easy exchange of database content, independent of proprietary “dump” formats.

Table of Contents

1 Status of the document 5
1.1 Version 2.0 5
2 Introduction 5
2.1 Structure of the document 5
2.1.1 Structure of chapters 5
2.1.2 ID for requirements 6
2.1.3 Distinction between mandatory and optional requirements 6
2.1.4 Notation of folders, files and folder structures 7
2.2 Addressees / target group 8
2.3 Background 8
2.4 Distinctions 8
3 General requirements / principles 10
3.1 Use of standards 10
3.2 Databases as documents 10
3.3 Character sets and characters 10
3.4 File URI Scheme 11
3.5 Identifiers and regular identifiers 11
4 Requirements for the format structure 13
4.1 Construction of the SIARD archive file 13
4.2 Structure of the SIARD archive file 13
4.3 Correspondence between metadata and table data 16
5 Requirements for metadata 21
5.1 Database level metadata 21
5.2 Schema level metadata 23
5.3 Type level metadata 24
5.4 Attribute level metadata 25
5.5 Table level metadata 25
5.6 Column level metadata 27
5.7 Field metadata 28
5.8 Primary key metadata 29
5.9 Foreign key metadata 30

5.10 Reference metadata 30
5.11 Candidate key metadata 31
5.12 Check constraint metadata 31
5.13 Trigger level metadata 32
5.14 View level metadata 32
5.15 Routine level metadata 33
5.16 Parameter metadata 33
5.17 ParameterField metadata 34
5.18 User level metadata 35
5.19 Role level metadata 35
5.20 Privilege level metadata 35
6 Requirements for table data 37
6.1 Table schema definition 37
6.2 Large object data cells 38
6.3 Date and timestamp data cells 39
6.4 Table data 39
7 Version and validity of the specification 42
8 Change management process 42
9 Exclusion of liability / notice regarding third-party rights 42
10 Copyright 42
Appendix A – Participation and Review 45
Appendix B – Abbreviations and Glossary 46
Appendix C – List of Standards Used 48
Appendix D – XML Schema Definitions 49
D.1 `metadata.xsd` 49
D.2 Example of a `metadata.xml` 65
D.3 Example of the XML schema definition of a table: `table0.xsd` 73
D.4 Example of the table data of a table: `table0.xml` 75
D.5 Example of the XML schema definition of a table with advanced and structured data types: table1.xsd 76
D.6 Example of the table data of a table with advanced and structured data types: table1.xml 78

1 Status of the document

This document has been submitted to the Swiss Federal Archives, the Swiss Koordinationsstelle für die dauerhafte Archivierung elektronischer Unterlagen (KOST)[1] and the EU project E-ARK[2]. It is an extension of the standard eCH-0165 for the SIARD Format 1.0. It has the force of a standard for its defined area of use in the specified scope of validity.

1.1 Version 2.0

Version 1.0 of the SIARD Format was developed in 2007 by Enter AG for the Swiss Federal Archives. It was homologized as standard eCH-0165 in 2013. This version 2.0 is based on version 1.0 and defines a format that is backward-compatible with version 1.0.

The main new features are:

- Upgrade of SQL:1999 support to SQL:2008 support.
- Support for all SQL:2008 types, in particular user-defined data types (UDTs),
- More explicit validation rules for data type definitions using regular expressions,
- Small modification of the definition, showing when to store large objects inline as part of the table XML,
- Support for storing large objects outside of the SIARD file using “file:” URIs[3],
- Support for “deflate” as a compression mechanism[4].

2 Introduction

2.1 Structure of the document

2.1.1 Structure of chapters

Each chapter in this Specification is constructed according to the same pattern. After a brief introduction, the requirements are listed in a table.

ID	Description of requirement	M/O
contains the ID of the requirement	contains the text of the requirement	stipulates whether mandatory or optional

[1] http://kost-ceco.ch/

[2] http://www.eark-project.com/

[3] http://en.wikipedia.org/wiki/File_URI_scheme , http://tools.ietf.org/html/rfc1738

[4] Requested by KOST. See http://www.ech.ch/alfresco/guestDownload/attach/workspace/SpacesStore/7822a908-cf37-4146-9dab-90a5763380f2/Addendum_2015-01-23_eCH-0165_V1.0_SIARD-Formatspezifikation.pdf

A requirement is frequently further explained by means of recommendations, notes and examples, each of which is specifically indicated as such.

ID	Description of requirement	M/O
A_3.1-1	Text of requirement **Example** Text of example **Note** Text of note ***Recommendation*** *The text of recommendations is in italics.*	M

2.1.2 ID for requirements

The requirements are unambiguously identifiable by means of an ID.

ID
A_3.1-1

This ID is constructed according to the following pattern:

G_	Letter + _		identifies main chapters
	G_	=	General requirements / principles
	T_	=	Requirements for table data
	M_	=	Requirements for metadata
	P_	=	Requirements for package structure
3.1-1	The number begins with the number of the chapter (which groups together requirements on the same topic), and the number after the dash is consecutive, thus designating all the requirements in the chapter.		

2.1.3 Distinction between mandatory and optional requirements

Each requirement is either mandatory or optional. This is indicated by a letter:

Abbreviation	Meaning
M	Mandatory requirement This requirement must be met in order to obtain a valid SIARD file.
O	Optional requirement This requirement should be met. It simplifies handling and constitutes best practice.

2.1.4 Notation of folders, files and folder structures

The following symbols and parameters are used for the notation of folders, files, etc.

Symbol	Meaning
/	Folder
header/	A folder with the name “header”
xy.txt	File (with file extension “txt”)
dir1/	Example folders (in red)
abc.pdf	Example files (in red)
…	Placeholder for files or folders that are not relevant to the explanation
[]	Placeholder for an expression or basic type such as “string”, “integer”, etc.
<xx>	Placeholder for any desired string of characters

2.2 Addressees / target group

This is a technical document for IT specialists involved in the long-term archiving of relational databases.

2.3 Background

The term “SIARD” stands for Software Independent Archival of Relational Databases. It is an open file format for the long-term archiving of relational databases in the form of text data based on XML that are packaged in a container file (SIARD archive)[5].

Long-term archiving is the preservation, normally without a time limit, of the information stored in the SIARD files, while retaining the bit stream and the ability to interpret and display the data in a way that is human-readable and comprehensible.

If the structure and content of a relational database are translated into the SIARD format, it will subsequently be possible to access and exchange the data in the database at any time, even when the original database software is no longer available or can no longer be run. This has been achieved by the use of suitable standards for the SIARD format that are widely supported internationally. This long-term interpretability of the database content is essentially based on two standards: XML and SQL:2008.

2.4 Distinctions

It should be noted that the SIARD format is only the long-term storage format for a specific type of digital documents (relational databases), and is therefore designed entirely independently of package structures such as the SIP (Submission Information Package), AIP (Archival Information Package) and DIP (Dissemination Information Package) in the OAIS model[6].

It is assumed that a database in SIARD format is archived as part of such an information package together with other documents (externalized large object files, translation maps for external file names, database documentation, business documents relevant to the understanding of the database, etc.).

Just as an XML-based Word or e-mail file contains an internal file structure consisting of metadata, primary data and various auxiliary data, an archived relational database in SIARD format contains its own metadata describing the document more precisely in addition to the actual table data – regardless of the metadata catalogue that an archive records in its OAIS packages.

---

[5] The SIARD database archiving format is distinct from the SIARD Suite application. This was developed by the Swiss Federal Archives SFA in order to generate and edit SIARD files and import them back into database environments

[6] http://public.ccsds.org/publications/archive/650x0m2.pdf

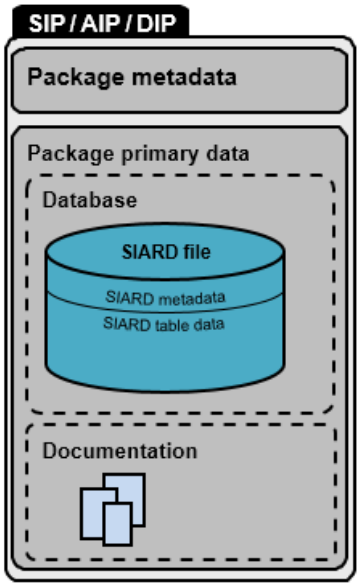


Fig. 1: Diagram of an information package containing a SIARD file

3 General requirements / principles

3.1 Use of standards

To ensure that the contents of a database remain interpretable over a long period, the SIARD format is essentially based on two ISO standards: XML and SQL:2008.

ID	Description of requirement	M/O
G_3.1-1	All database content is stored in a collection of XML files in accordance with ISO/IEC 19503:2005. The schema definitions and SQL code must in each case conform to SQL:2008 in accordance with ISO/IEC 9075. The only exceptions are BLOB and CLOB data (Binary Large OBjects and Character Large OBjects), which are stored in separate binary and text files but are referenced in the XML files.	M

3.2 Databases as documents

A relational database is treated as a single document to be archived, so that the references between the data in individual tables are preserved.

ID	Description of requirement	M/O
G_3.2-1	A relational database is archived in a single SIARD file.	M
G_3.2-2	The primary data in a relational database are completely archived in a SIARD file. This means that each SELECT query directed to the original and the archived data yields the same results[7].	M

3.3 Character sets and characters

ID	Description of requirement	M/O
G_3.3-1	In general, all the data are stored in the Unicode character set in accordance with ISO 10646.	M
G_3.3-2	When extracting from databases that support other character sets, mapping to the corresponding Unicode character sets is carried out. For this reason, the national character string types (NCHAR, NVARCHAR, NCLOB) from the database product must generally be translated into non-national types (CHAR, VARCHAR or CLOB). This convention is supported by XML, irrespective of whether an XML file is stored in UTF-8 format or in UTF-16 format.	M
G_3.3-3	In the XML files in the SIARD format, all characters that have a special meaning in the XML syntax are replaced by unit references, of the type xs:string in all fields. Additionally, the Unicode control characters 0-31 and 127-159 are coded using the solidus ("\") to ensure that the validity of the XML file is guaranteed.	M

[7] For more details see http://www.enterag.ch/hartwig/SIARD_Criterion.pdf.

<table>
<tr><th>ID</th><th>Description of requirement</th><th>M/O</th></tr>
<tr><td>G_3.3-4</td><td>Characters that cannot be represented in UNICODE (codes 0-8, 14-31, 127-159), as well as the escape character '\' and multiple space characters are escaped as \u00<xx> in XML. Quote, less than and ampersand are represented in XML as entity references.

<table>
<tr><th>Original characters</th><th>Characters in the SIARD format</th></tr>
<tr><td>0 to 8</td><td>\u0000 to \u0008</td></tr>
<tr><td>14 to 31</td><td>\u000E to \u001F</td></tr>
<tr><td>32</td><td>\u0020, for multiple spaces</td></tr>
<tr><td>"</td><td>"</td></tr>
<tr><td>&</td><td>&</td></tr>
<tr><td>'</td><td>'</td></tr>
<tr><td><</td><td><</td></tr>
<tr><td>></td><td>></td></tr>
<tr><td>\</td><td>\u005c</td></tr>
<tr><td>127 to 159</td><td>\u007F to \u009F</td></tr>
</table></td><td>M</td></tr>
</table>

3.4 File URI Scheme

The file URI scheme used for referencing externally stored large objects is defined in RFC 1738[8].

ID	Description of requirement	M/O
G_3.4-1	In general, all externally referenced files are specified using a file URI as specified in RFC 1738.	M
G_3.4-2	File URIs are stored as URL-encoded ASCII strings in a SIARD archive.	M
G_3.4-3	It is recommended to use a file system for file URIs, where individual file entries within a ZIP file can be addressed. Thus. *file:///d:/sips/sip1234.zip* would refer to the ZIP file, whereas *file:///d:/sips/sip1234.zip/* refers to the root folder *inside* the ZIP file.	O

3.5 Identifiers and regular identifiers

In SQL:2008 there are regular identifiers[9] excluding spaces and special characters which are not case-sensitive but are stored in upper case in the SIARD archive, and delimited identifiers which are case-sensitive and may also contain special characters or equal an SQL

[8] http://en.wikipedia.org/wiki/File_URI_scheme , http://tools.ietf.org/html/rfc1738

[9] An SQL:2008 identifier must begin with a letter (A-Z) or an underscore (_) followed by letters (A-Z), numbers (0-9) or an underscore (_), with a maximum of 128 characters.

keyword. These must be quoted in double quotation marks in expressions. In the SIARD archive they are stored without the quotes.

The definition of what constitutes a special character is set out in the SQL standard. The upper-case version of a letter is defined by the Unicode standard.

In the metadata, a regular identifier is stored in upper case, while a delimited identifier is stored without quotation marks. The SQL:2008 standard stipulates that as soon as an identifier contains a character that is not permitted in a regular identifier or equals an SQL keyword, it is deemed to be a delimited identifier.

ID	Description of requirement	M/O
G_3.5-1	In general, all identifiers are stored in the Unicode character set.	M
G_3.5-2	Regular identifiers are in upper case without quotation marks.	M
G_3.5-3	Delimited identifiers are stored without quotation marks.	M

4 Requirements for the format structure

4.1 Construction of the SIARD archive file

The SIARD archive file is realised as a ZIP archive.

ID	Description of requirement	M/O
G_4.1-1	The SIARD file is stored as a single, ZIP archive in accordance with the specification published by the company PkWare, version 6.3.2[10].	M
G_4.1-2	SIARD files must either be uncompressed or else compressed using the “deflate” algorithm as described in RFC 1951[11].	M
G_4.1-3	The SIARD file is not password-protected or encrypted.	M
G_4.1-4	Both the ZIP32 and ZIP64 variants are permitted for the ZIP archive.	M
G_4.1-5	The ZIP archive has the file extension “.siard”.	M

4.2 Structure of the SIARD archive file

A relational database archived in the SIARD format consists of two components: the metadata, which describe the structure of the archived database, and the table data, which represent the table content. The metadata also indicate where the various table data are to be found in the archive.

ID	Description of requirement	M/O

P_4.2-1	The table data are located in the `content/` folder and the metadata in the `header/` folder. No further folders or files are permitted.
	
	Example
	Structure of the SIARD file (schematic)
	`Northwind.siard`
	`content/`
	`header/`
		M

[10] ZIP files were originally defined by Phil Katz and are today extensively used as a *de facto* standard. The current version 6.3.2 of the specification published by PkWare can be found at http://www.pkware.com/documents/casestudies/APPNOTE.TXT.

[11] https://www.ietf.org/rfc/rfc1951.txt

ID	Description of requirement	M/O

P_4.2-2	The `content/` folder contains one or more schema folders in which the individual table folders are located. No other folders or files are permitted.
	
	Example
	Structure of the SIARD file (schematic)
	`Northwind.siard`
	`content/`
	`schema0/`
	`table0/`
	`table1/`
	`table2/`
	`...`
	`schema1/`
	`table0/`
	`...`
	
	Recommendation
	It is advisable to normalise the schema and table folder names and to use, for example, `schema0/` *and* `table0/` *instead of the actual name (see restrictions under P_4.2-5).*
		M

P_4.2-3	The individual table folders contain an XML file and an XSD file, the names of which (folder designation and both file names) must be identical. With the exception of BLOB and CLOB folders together with their content (BIN or TXT files), no other folders or files are permitted.
	
	Example
	Structure of the SIARD file (schematic)
	`Northwind.siard`
	`content/`
	`schema0/`
	`table0/`
	`table0.xml`
	`table0.xsd`
	`lob4`[12]`/`
	`record0.bin`
	`record1.bin`
	`table1/`
	`table1.xml`
	`table1.xsd`
	`...`
	
	Recommendation
	It is advisable to normalise the lob folders and lob files and to use, for example, `lob4/` *and* `record0.bin` *or* `record0.txt` *instead of the actual name (see restrictions under P_4.2-6).*
		M

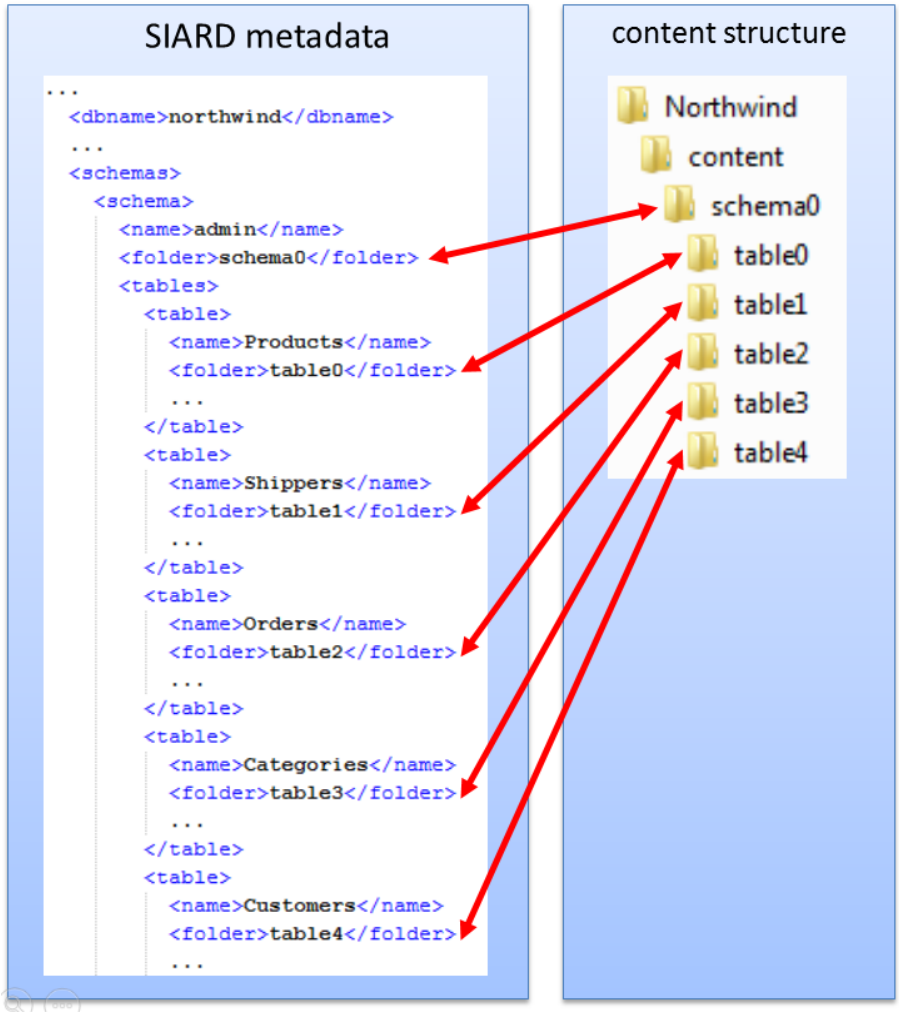
[12] In this example, column 4 contains additional lob files that are accordingly stored in `lob4/`.

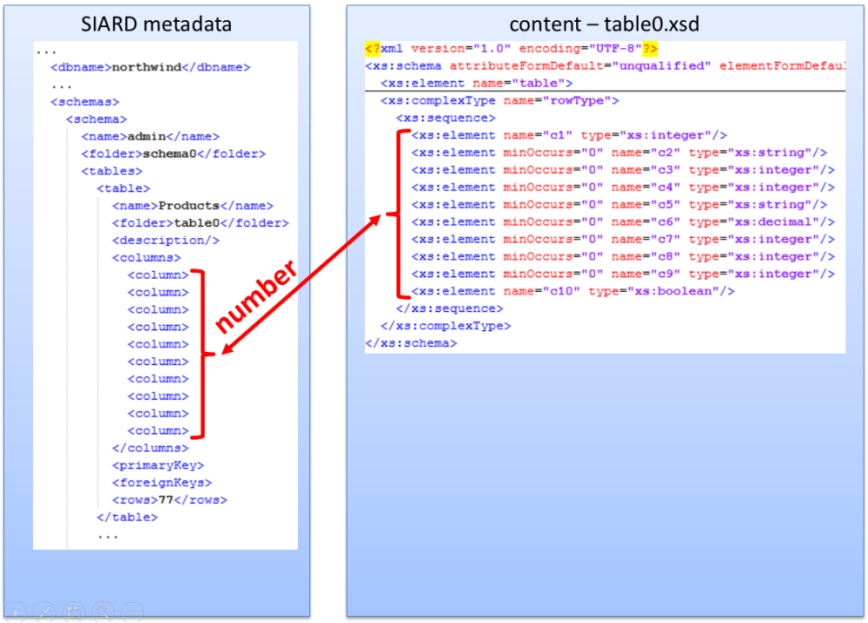
ID	Description of requirement	M/O
P_4.2-4	The header/ folder must contain an empty subfolder /header/version/2.0 identifying the version of the SIARD Format in order to facilitate the recognition of the SIARD Format (e.g. by PRONOM).	M

P_4.2-5	The `metadata.xml` and `metadata.xsd` files must be present in the `header/` folder. Additional files, such as style sheets, are permitted.
	
	Example
	Structure of the SIARD file (schematic)
	`Northwind.siard`
	`content/`
	`...`
	`header/`
	`metadata.xml`
	`metadata.xsd`
	`...`
		M

P_4.2-6	All file and folder names must be structured as follows: The name must begin with a letter [a-z or A-Z] and must then contain only the following characters: • a-z • A-Z • 0-9 • _ • . (may only be used to separate the name from the extension) ***Recommendation*** *Where possible, the length of the file and folder names should not exceed 20 characters to avoid problems with excessively long path lengths in Windows.*	M

4.3 Correspondence between metadata and table data

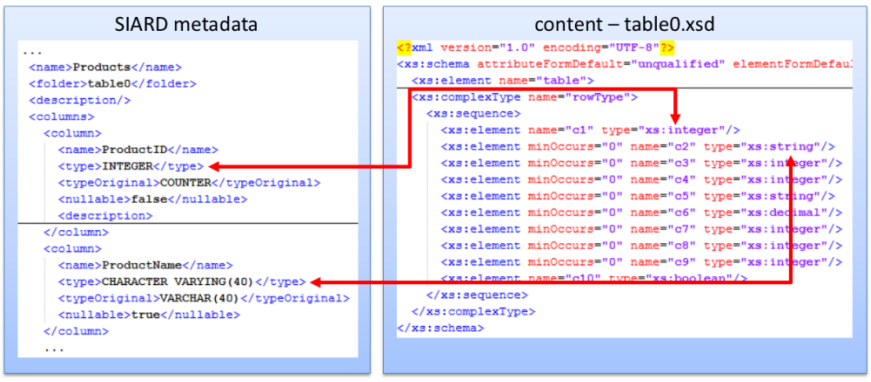
P_4.3-1	The structure prescribed in the `metadata.xml` must be identical to that in the `content/` folder. **Example**  	M

P_4.3-2	The number of columns in a table specified in the `metadata.xml` must be identical to that in the corresponding `table[number].xsd` file. **Example**  	M
P_4.3-3	The data type information on the column definitions in the `metadata.xml` must be identical to that in the corresponding `table[number].xsd` file. The SQL:2008 built-in data types are converted into XML data types in the `table[number].xsd` schema files in accordance with the following table.	M

SQL:2008	XML
BIGINT	xs:integer
BINARY LARGE OBJECT	blobType
BINARY VARYING(…)	xs:hexBinary
BINARY(…)	xs:hexBinary
BIT VARYING(…)	xs:hexBinary
BIT(…)	xs:hexBinary
BOOLEAN	xs:boolean
CHARACTER LARGE OBJECT	clobType
CHARACTER VARYING(…)	xs:string
CHARACTER(…)	xs:string
DATE	dateType
DECIMAL(…)	xs:decimal
DOUBLE PRECISION	xs:float

FLOAT(...)	xs:float
INTEGER	xs:integer
INTERVAL	xs:duration
NATIONAL CHARACTER LARGE OBJECT	clobType
NATIONAL CHARACTER VARYING(...)	xs:string
NATIONAL CHARACTER(...)	xs:string
NUMERIC(...)	xs:decimal
REAL	xs:float
SMALLINT	xs:integer
TIME	timeType
TIME WITH TIME ZONE	timeType
TIMESTAMP	dateTimeType
TIMESTAMP WITH TIME ZONE	dateTimeType
XML	clobType

Example



All date and time types are specified in the UTC “time zone”. It is recommended, that they are terminated with a “Z”.

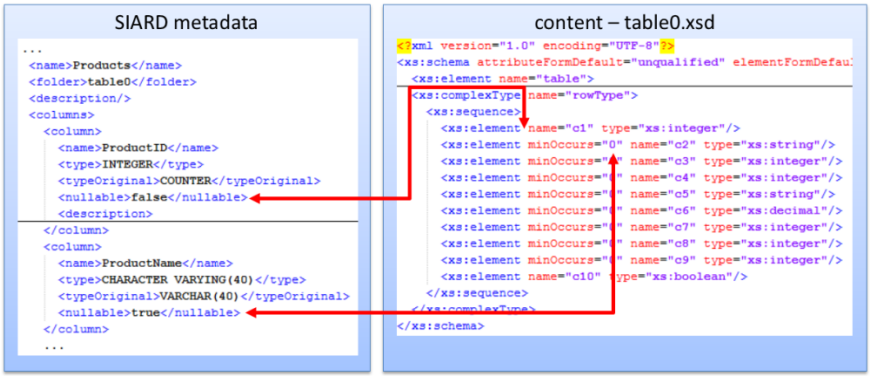
The dateType is a restriction of xs:date in UTC to years between 0001 and 9999 (SQL:2008 restriction).

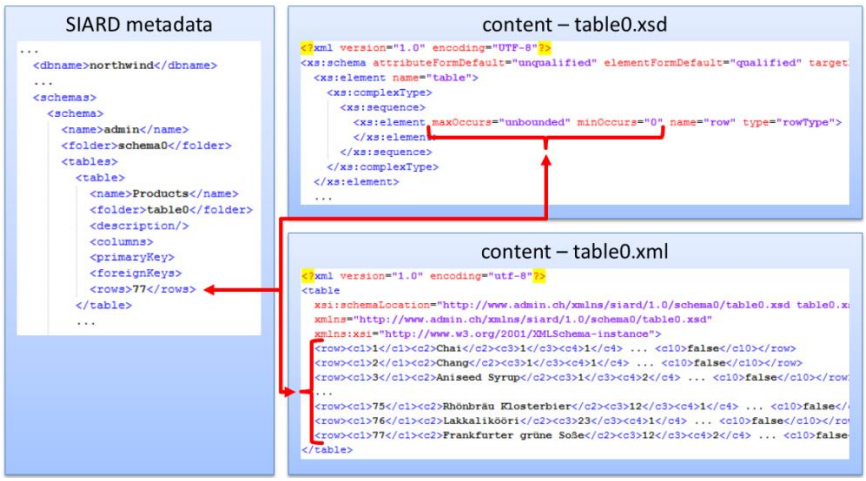
The timeType is a restriction of xs:time to the UTC “time zone”.

The dateTimeType is a restriction of xs:dateTime in the UTC “time zone” to years between 0001 and 9999 (SQL:2008 restriction).

The clobType is an extension of xs:string (for inlined values) with optional attributes file, length and messageDigest, which are needed, if the CLOB value is not simply inlined as a string. In this case the value of the file attribute is the (URL-encoded) file URI (possibly relative to the nearest lobFolder), where the CLOB is stored, the value of the length attribute is the *length in (UTF-8)*, and the optional messageDigest is a string following the same pattern (algorithm+value) as the global messageDigest (see 5.1-1).

The blobType is an extension of xs:hexBinary (for inlined values) with optional attributes file, length and messageDigest, which needed if the BLOB value is not

	simply inlined as a hex string. In this case the value of the file attribute is the (URL-encoded) file URI (possibly relative to the nearest lobFolder), where the BLOB is stored, the value of the length attribute is the *length in bytes*, and the optional messageDigest is a string following the same pattern (algorithm+value) as the global messageDigest (see 5.1-1)	
P_4.3-4	The named DISTINCT data types are converted to the XML data type in the `table[number].xsd` schema files which would be used for representing their base types.	M
P_4.3-5	The named ROW container type is converted in the `table[number].xsd` schema files into a sequence of structured XML elements `<r1>, <r2>, …` containing an entry for each field. The data type of each field is converted to the XML data type just like a column data type. See example `table0.xsd` in appendix D3.	M
P_4.3-6	The ARRAY container type is converted in the `table[number].xsd` schema files into a sequence of structured XML elements `<a1>, <a2>, …` which are converted to the XML data type corresponding to the base type of the array. See example `table0.xsd` in appendix D3.	M
P_4.3-7	The named user-defined data type (UDT) is converted in the `table[number].xsd` schema files into a sequence of structured XML elements `<u1>, <u2>, …` which are converted to the XML data type corresponding to the type of each attribute. See example `table0.xsd` in appendix D3.	M
P_4.3-8	The nullable information on the column definitions in the `metadata.xml` must be identical to that in the corresponding `table[number].xsd` file. **Example**  **Note** The SQL:2008 notation "<nullable>true</nullable>" becomes “minOccurs="0"" in XML. "<nullable>false</nullable>" corresponds to "minOccurs="1"" in XML; however, as this is the default value, it is often omitted.	M
P_4.3-9	The column sequence in the `metadata.xml` must be identical to that in the corresponding `table[number].xsd`.	M
P_4.3-10	The field sequence in the type definition of a column of the `metadata.xml` must be identical to the corresponding attribute sequence in the type definition of the `metadata.xml`.	M

P_4.3-11	The field sequence in the table definition of the `metadata.xml` must be identical to the field sequence in the corresponding `table[number].xsd`.	M
P_4.3-12	The number of lines in a table in `metadata.xml` must fit into the area specified in the corresponding `table[number].xsd`. The number of lines in a table in `metadata.xml` must be identical to the number of lines in the corresponding `table[number].xml`. **Example**  ***Recommendation*** *It is advisable to use the range 0 to infinity (maxOccurs="unbounded" minOccurs="0") in the* `table[number].xsd`*. This avoids problems when validating* `table[number].xml` *against* `table[number].xsd`*.*	M

5 Requirements for metadata

The metadata in the SIARD archive store the structure of the archived database and indicate where different table data are to be found in the archive.

All the metadata are collated in a single `metadata.xml` file in the `header/` folder. Unlike a relational database, the file has a hierarchical structure.

There is a schema definition `metadata.xsd` for the `metadata.xml` file. This is also stored in the `header/` folder.

ID	Description of requirement	M/O
M_5.0-1	The schema definition `metadata.xsd` must be complied with for the `metadata.xml` file. This means that `metadata.xml` must be capable of being positively validated against `metadata.xsd`.	M

The contents of the individual levels are defined below.

5.1 Database level metadata

The `metadata.xml` file contains the following global information at database level:

ID	Description of requirement	M/O
M_5.1-1	All metadata that are designated as mandatory in `metadata.xsd` at database level must be completed accordingly.	M

The following database metadata are stored in the `metadata.xml` file:

Identifier	Meaning	M/O
version	SIARD format version	M
dbname	Short database identifier	M
description	Description of the meaning and content of the database as a whole	O
archiver	Name of the person who carried out the archiving of the table data from the database	O
archiverContact	Contact details (telephone, e-mail) of the person who carried out the archiving of the table data from the database	O
dataOwner	Owner of the data in the database; the institution or person that, at the time of archiving, has the right to grant usage rights for the data and is responsible for compliance with legal obligations such as data protection guidelines	M
dataOriginTimespan	Origination period of the data in the database; approximate indication in text form	M

Identifier	Meaning	M/O
lobFolder	A “file:” URI representing the base URI for relative URIs indicating the possibly external storage location of large objects. If it is missing, default value of the root folder inside the ZIP file is assumed. Relative *lobFolder* URIs in the column metadata are relative to this value. **Note** If the “file:” URI refers to an extended file system, where ZIP files are treated as folders, the relative URI “..” refers to the external folder, where the SIARD archive resides. If such a file system extension is not supported, absolute “file:” URIs must be used for specifying an external storage location for LOB files. It is strongly recommended to choose all column *lobFolder* entries and all LOB *file* attributes as relative URIs. Thus, on relocation of the SIARD file or its information package, only this absolute URI must be changed to point to the new location.	O
producerApplication	Name and version of the application that downloaded the SIARD file.	O
archivalDate	Date on which the table data were archived	M
messageDigest	Hexadecimal message digest code over the `content/` folder with a prefix indicating the type of digest algorithm (like MD5, SHA-1 or SHA-256) followed by the hexadecimal message digest code – for example “MD55234Fd874EFADFCA86E5CDDC97398991”. More than one message digest (based on different algorithms) can be stored. If no message digest is stored, then the integrity must be assured by storing something like a message digest outside the SIARD file. ***Recommendation*** *If the message digest option is used, the following must be implemented:* *The content and header directories are stored in the ZIP file as separate (empty)* `content/` *and* `header/` *entries. To ensure that the integrity of the primary data can be checked, the entry for the header directories must be inserted after all the primary data in the* `content/` *entry and before all the other metadata entries. The message digest mentioned below is computed from offset 0 to the offset of the* `header/` *entry of the SIARD archive.*	O
clientMachine	DNS name of the (client) computer on which the archiving was carried out	O
databaseProduct	Database product and version from which the archiving of the table data was carried out	O
connection	Connection string used to archive the table data	O
databaseUser	Database user ID of the user of the SIARD tool for archiving the table data from the database	O
schemas	List of schemas in the database	M
users	List of database users	M
roles	List of database roles	O

Identifier	Meaning	M/O
privileges	List of user and role privileges	O

5.2 Schema level metadata

The schema metadata are archived in the `metadata.xml` file as with the global information on the database.

ID	Description of requirement	M/O
M_5.2-1	All metadata that are designated as mandatory in `metadata.xsd`at schema level must be completed accordingly.	M

The following schema metadata are stored in the `metadata.xml`file:

Identifier	Meaning	M/O
name	Schema name in the database	M
folder	Name of the schema folder under `content/` in the SIARD archive	M
types	List of (named) advanced or structured types in the schema	O
description	Description of the meaning and content of the schema	O
tables	List of the tables in the database	M
views	List of the queries stored in the database	O
routines	List of the routines (formerly “stored procedures”) in the schema	O

5.3 Type level metadata

ID	Description of requirement	M/O
M_5.3-1	The type metadata of a schema can be archived in the `metadata.xml`file	O

The following type metadata are stored in the `metadata.xml`file if an advanced or structured data type is archived:

Identifier	Meaning	M/O
name	Type name in the schema	M
category	Category of advanced or structured type (“distinct”, “row”, “array” or “udt”).	M
underSchema	If the type is based on a super type, schema name of super type	O
underName	If the type is based on a super type, name of super type	O
instantiable	True, if type can be instantiated, false otherwise	M
final	True, if no sub types can be created to this type, false otherwise	M
base	Name of (primitive) base type if category is “distinct”	O
cardinality	(Maximum) number of elements if category is “array”	O
attributes	List of attributes, if category is “row” or “udt”	O
description	Description of the meaning and content of the data type	O

5.4 Attribute level metadata

ID	Description of requirement	M/O
M_5.4-1	The attribute metadata that are used in the type can be archived in the `metadata.xml` file	O

Identifier	Meaning	M/O
name	Name of the attributer	M
type	SQL:2008 built-in data type of the attribute	O
typeOriginal	Original column type for the built-in type **Note** As the various database programs that describe themselves as SQL-compliant permit very different data types, the *original* type is listed here as well as the SQL:2008 type. A translation of the proprietary types to SQL:2008 types is to be defined and documented in the corresponding application for each database program that supports the SIARD format.	O
typeSchema	Schema of advanced or structured data type	O
typeName	Name of advanced or structured data type	O
attributes	List of attributes, if category of type of attribute is “row” or “udt”	O
defaultValue	Default value of attribute	O
description	Description of the meaning and function of the routine	O

5.5 Table level metadata

Like the global information on the database and the schema metadata, table level metadata are archived in the `metadata.xml` file.

ID	Description of requirement	M/O
M_5.5-1	All metadata that are designated as mandatory in `metadata.xsd`at table level must be completed accordingly.	M

The following table metadata are stored in the `metadata.xml`file:

Identifier	Meaning	M/O
name	Table name in the schema	M
folder	Name of the table folder in the schema folder	M

Identifier	Meaning	M/O
description	Description of the meaning and content of the table	O
columns	List of the columns in the table	M
primaryKey	Primary key of the table	O
foreignKeys	List of the foreign keys in the table	O
candidateKeys	List of the candidate keys in the table	O
checkConstraints	List of the constraints in the table	O
triggers	List of the triggers in the table	O
rows	Number of datasets	M

5.6 Column level metadata

Like the global information on the database, the schema metadata and the table level metadata, the column level metadata are archived in the `metadata.xml`file. Column metadata describe a column in a table or a view.

ID	Description of requirement	M/O
M_5.6-1	All metadata that are designated as mandatory in the `metadata.xsd`at column level must be completed appropriately.	M

The following column metadata are stored in the `metadata.xml`file:

Identifier	Meaning	M/O
name	Column name in the table, ROW or UDT. The column name must be unambiguous within a given table.	M
folder	Entry name of the LOB folder in the table folder, if the column is stored internally to the SIARD archive. **Note** The optional LOB folder name is only needed for columns of the large object types (e.g. BLOB, CLOB or XML) which are stored internally in the SIARD archive. The files that the large object fields represent are stored in these folders and are called `record0.txt`, `record1.txt`, and `record0.bin`, `record1.bin`, … or a1/record2.bin, r1/record234.txt, … These are referenced in the data XML file.	O
lobFolder	Name of the LOB folder given as a relative or absolute “file:” URI – possibly in the external file system. It may be used for internal as well as external storage of large objects. **Note** This entry is only meaningful, if the column is a LOB column (e.g. type BLOB, CLOB or XML). If it is missing, it is assumed to equal “.”, e.g. to refer to the lobFolder element of the enclosing ROW or UDT or else to same folder as the *externalBase* on the database level. Otherwise its value must be a – preferably relative – “file:” URI, indicating the folder, where the files of this LOB column are to be stored. If this value is a relative URI, it is assumed to be relative to the lobFolder element of the enclosing ROW or UDT or else the global *externalBase* entry at the database level. The *file* attributes of the cells in this column are interpreted as being relative to this folder.	O
type	SQL:2008 built-in data type of the column **Note** If the data type of this column is a built-in data type, this field must be used. Otherwise the field `typeName` must refer to a defined type in the types list.	O

Identifier	Meaning	M/O
typeOriginal	Original column type **Note** As the various database programs that describe themselves as SQL-compliant permit very different data types, the *original* type is listed here as well as the SQL:2008 type. A translation of the proprietary types to SQL:2008 types is to be defined and documented in the corresponding application for each database program that supports the SIARD format.	O
nullable	Optional entry	O
typeSchema	Schema of named type if the column is not a built-in data type and the named data type is not defined in the same schema as the table of this column.	O
typeName	Name of the advanced or structured data type of this column	O
fields	List of fields in the column, if the column is a structured data type of category “row” or “udt”	O
defaultValue	Default value of the column	O
mimeType	MIME type of this column, if it is a BLOB column and all records contain files of the same MIME type in this column. This purely advisory element helps choosing the correct viewer for binary objects. It can be filled in manually or by the downloading program, which uses some format recognition mechanism.	O
description	Description of the meaning and content of the column	O

5.7 Field metadata

ID	Description of requirement	M/O
M_5.7-1	The field metadata of a column or a field can be archived in the `metadata.xml` file	O

The following field metadata are stored in the `metadata.xml` file if a column or a field is an advanced or structured data type of category “row” or “udt”:

Identifier	Meaning	M/O
folder	Entry name of the LOB folder in the table folder, if the column or field is stored internally to the SIARD archive. **Note** The optional LOB folder name is only needed for columns or fields of the large object types (e.g. BLOB, CLOB or XML) which are stored internally in the SIARD archive. The files that the large object fields represent are stored in these folders and are called `record0.txt`, `record1.txt`, and `record0.bin`, `record1.bin`, … or a1/record2.bin, r1/record234.txt, … These are referenced in the data XML file.	O
lobFolder	Name of the LOB folder given as a relative or absolute “file:” URI – possibly in the external file system. It may be used for internal as well as external storage of large objects. **Note** This entry is only meaningful, if the column or field is a LOB (e.g. type BLOB, CLOB or XML). If it is missing, it is assumed to equal “.”, e.g. to refer to the `lobFolder` element of the enclosing column or field element. Otherwise its value must be a – preferably relative – “file:” URI, indicating the folder, where the files of this LOB column are to be stored. If this value is a relative URI, it is assumed to be relative to the `lobFolder` element of the enclosing element. The *file* attributes of the cells in this column or field are interpreted as being relative to this folder.	O
fields	List of fields in the field, if the field is a structured data type of category “row” or “udt”	O
mimeType	MIME type of this field, if it is a BLOB column and all records contain files of the same MIME type in this field. This purely advisory element helps choosing the correct viewer for binary objects. It can be filled in manually or by the downloading program, which uses some format recognition mechanism.	O
description	Description of the meaning and content of the field	O

5.8 Primary key metadata

ID	Description of requirement	M/O
M_5.8-1	The primary key metadata of a table can be archived in the `metadata.xml` file	O

The following primary key metadata are stored in the `metadata.xml` file if a primary key is archived:

Identifier	Meaning	M/O
Name	Name of the primary key	M
column	List of the columns in the primary key	M
description	Description of the meaning and content of the primary key	O

5.9 Foreign key metadata

ID	Description of requirement	M/O
M_5.9-1	The foreign key metadata in a table can be archived in the `metadata.xml` file	O

The following foreign key metadata are stored in the `metadata.xml`file if a foreign key is archived:

Identifier	Meaning	M/O
Name	Name of the foreign key	M
referencedSchema	Schema of the table referenced	M
referencedTable	Table that is referenced **Note** The external table name referenced can be of the table or schema.table type. Delimited identifiers are enclosed in quotation marks.	M
reference	List of columns and referenced columns	M
matchType	Match type (FULL, PARTIAL or SIMPLE)	O
deleteAction	Delete action, e.g. CASCADE **Note** The delete and change actions contain the actions permitted by the SQL:2008 standard.	O
updateAction	Change action, e.g. SET DEFAULT	O
description	Description of the meaning and content of the foreign key	O

5.10 Reference metadata

ID	Description of requirement	M/O
M_5.10-1	The metadata of the references used in the foreign key can be archived in the `metadata.xml` file	O

The following reference metadata are stored in the `metadata.xml`file if a foreign key is archived:

Identifier	Meaning	M/O
column	Name of the column	M
referenced	Name of the referenced column	M

5.11 Candidate key metadata

ID	Description of requirement	M/O
M_5.11-1	The metadata of the candidate key of a table can be archived in the `metadata.xml`file	O

The following candidate key metadata are stored in the `metadata.xml`file if a candidate key is archived:

Identifier	Meaning	M/O
name	Name of the candidate key	M
column	List of the columns in the candidate key	M
description	Description of the meaning and content of the candidate key	O

5.12 Check constraint metadata

The check constraint consists of a condition that is to be examined. This is indicated as an BOOLEAN expression (having the value *true*, *false* or *unknown*) in SQL:2008 syntax.

ID	Description of requirement	M/O
M_5.12-1	The metadata of the check constraint of a table can be archived in the `metadata.xml`file	O

The following check constraint metadata are stored in the `metadata.xml`file if a check constraint is archived:

Identifier	Meaning	M/O
Name	Name of the check constraint	M
condition	Condition of the check constraint	M
description	Description of the meaning and content of the check constraint	O

5.13 Trigger level metadata

ID	Description of requirement	M/O
M_5.13-1	The trigger metadata of a table can be archived in the `metadata.xml`file	O

The following trigger metadata are stored in the `metadata.xml`file if a trigger is archived:

Identifier	Meaning	M/O
Name	Trigger name in the table	M
actionTime	BEFORE or AFTER	M
triggerEvent	INSERT, DELETE, UPDATE [OF <trigger column list>]	M
aliasList	<old or new value alias list>	O
triggeredAction	<triggered action>	M
description	Description of the meaning and content of the trigger	O

5.14 View level metadata

ID	Description of requirement	M/O
M_5.14-1	The view metadata of a schema can be archived in the `metadata.xml`file	O

The following view metadata are stored in the `metadata.xml`file if a view is archived:

Identifier	Meaning	M/O
Name	Name of the view in the schema	M
columns	List of the column names in the view **Note** The column metadata of a view are structured identically to those of a table.	M
Query	SQL:2008 query that defines the view	O
queryOriginal	Original SQL query that defines the view **Note** As the various database programs that describe themselves as SQL-compliant permit very different query syntaxes, the original query is listed here as well as the SQL:2008 query. A translation of the proprietary query syntax to SQL:2008 types is to be defined and documented in the corresponding application for each database program that supports the SIARD format.	O

Identifier	Meaning	M/O
description	Description of the meaning and content of the view	O

5.15 Routine level metadata

ID	Description of requirement	M/O
M_5.15-1	The routine metadata of a schema can be archived in the `metadata.xml`file	O

The following routine metadata are stored in the `metadata.xml`file if a routine is archived:

Identifier	Meaning	M/O
Name	Routine name in the schema	M
description	Description of the meaning and content of the routine	O
source	Original source code of the routine (VBA, PL/SQL, JAVA) **Note** Since many database programs have proprietary routines that cannot be transformed into an SQL:2008-compliant query, the original source code of the routine (e.g. in PL/SQL for Oracle databases, VBA for MS Access modules) can be archived here.	O
body	SQL:2008-compliant source code of the routine	O
characteristic	Characteristic of the routine	O
returnType	Return type of the routine (if it is a function)	O
parameters	List of parameters	O

5.16 Parameter metadata

ID	Description of requirement	M/O
M_5.16-1	The parameter metadata that are used in the routine can be archived in the `metadata.xml` file	O

The following parameter metadata are stored in the `metadata.xml`file if a routine is archived:

Identifier	Meaning	M/O
name	Name of the parameter	M
mode	Mode of the parameter (IN, OUT or INOUT)	M

Identifier	Meaning	M/O
Type	SQL:2008 built-in data type of the parameter **Note** If the data type of this column is a built-in data type, this field must be used. Otherwise the field `typeName` must refer to a defined type in the types list.	O
typeOriginal	Original parameter type **Note** As the various database programs that describe themselves as SQL-compliant permit very different data types, the *original* type is listed here as well as the SQL:2008 type. A translation of the proprietary types to SQL:2008 types is to be defined and documented in the corresponding application for each database program that supports the SIARD format.	O
typeSchema	Schema of named type if the parameter is not a built-in data type and the named data type is not defined in the same schema as the table of this column.	O
typeName	Name of the advanced or structured data type of this parameter	O
parameterFields	List of fields in the parameter, if the parameter is a structured data type of category “row” or “udt”	O
Description	Description of the meaning and function of the routine	O

5.17 ParameterField metadata

ID	Description of requirement	M/O
M_5.17-1	The field metadata of a parameter or a parameterField can be archived in the `metadata.xml` file	O

The following parameterField metadata are stored in the `metadata.xml` file if a parameter or a parameterField is an advanced or structured data type of category “row” or “udt”:

Identifier	Meaning	M/O
parameterFields	List of parameterFields in the parameterField, if the parameterField is a structured data type of category “row” or “udt”	O
description	Description of the meaning and content of the parameterField	O

Neither name nor type of the parameter field need be recorded, because they are stored in the corresponding attribute list of the type. If the parameter is a simple type, the parameterFields list is omitted. However, a container for the description is still needed.

5.18 User level metadata

ID	Description of requirement	M/O
M_5.18-1	The user metadata can be archived in the `metadata.xml` file	O

The following user metadata are stored in the `metadata.xml`file:

Identifier	Meaning	M/O
name	Name of the user	M
description	Description of the significance and function of the user	O

5.19 Role level metadata

ID	Description of requirement	M/O
M_5.19-1	The role metadata can be archived in the `metadata.xml`file	O

The following role metadata are stored in the `metadata.xml`file:

Identifier	Meaning	M/O
name	Name of the role	M
admin	Administrator of the role (user or role)	M
description	Description of the meaning and function of the role	O

5.20 Privilege level metadata

ID	Description of requirement	M/O
M_5.20-1	The privilege metadata can be archived in the `metadata.xml`file	O

The following privilege metadata are stored in the `metadata.xml`file:

Identifier	Meaning	M/O
type	Privilege granted (e.g. SELECT)	M
object	Object to which the privilege is to be applied	O
grantor	Authorised grantor of the privilege	M
grantee	Recipient of the privilege (user or role)	M
option	Grant option (ADMIN or GRANT)	O

Identifier	Meaning	M/O
description	Description of the significance and function of the grant	O

6 Requirements for table data

As described above, the table data of an archived relational database are located in the `content/` folder in the document root of the SIARD archive. They are filed there in the schema and table folder concerned.

Table data are always stored in an XML file. An XML schema definition is generated for each table that indicates the XML storage format of the table data. This means that for each table there is a `table[number].xml` file to the schema definition `table[number].xsd`.

ID	Description of requirement	M/O
T_6.0-1	All the table data (primary data) must meet the consistency requirements of SQL:2008. A SIARD file that validates syntactically against the various XSDs but infringes the SQL standard semantically is not compliant with this format description. In particular, the table values must correspond to the constraints of the SQL types in the metadata. Additionally, the primary, candidate and foreign key conditions and nullability conditions stored in the metadata must all be met.	M
T_6.0-2	The schema definition `table[number].xsd` must be complied with for the `table[number].xml` file. This means that `table[number].xml` must be capable of being positively validated against `table[number].xsd`.	M

6.1 Table schema definition

The `table[number].xsd` file contains the following schema definitions for a table:

ID	Description of requirement	M/O
T_6.1-1	There must be an XML schema definition for each table that indicates the XML storage format of the table data.	M

ID	Description of requirement	M/O
T_6.1-2	This schema definition reflects the SQL schema metadata of the table and indicates that the table is stored as a sequence of lines containing a sequence of column entries with various XML types. The name of the table tag is *table*, that of the dataset tag is *row*, while the column tags are called *c1, c2, ...*. **Example** content – table0.xsd 	M
T_6.1-3	The type mapping to be used in table schema definitions is specified in P_4.3-3. Apart from XML Schema standard types the following special type are used: clobType, blobType, dateType, dateTimeType.	M
T_6.1-4	Multiple cell values of advanced or structured types are to be stored as separate elements inside the cell tags. The names of the individual elements of an ARRAY are a1, a2, …. The names of the individual elements of a ROW are r1, r2, …. The names of the individual elements of a UDT are u1, u2, …. See appendix D3 for an example.	?

6.2 Large object data cells

ID	Description of requirement	M/O
T_6.2-1	Large objects can be stored inline in the `table[number].xml` file, as separate file entries inside the SIARD archive, or externally as files in the file system	M

The following large object data are stored in a lob cell of the `table[number].xsd` file:

Identifier	Meaning	M/O
file	If the large object is not inlined, this element indicates the location and name of the large object file in this cell or cell attribute as a “file:” URI. If it is a relative URI it is interpreted as relative to the enclosing element’s (column or attribute) *lobFolder*.	O
length	Length (for BLOBs in bytes, for CLOBs and XML in characters)	O
messageDigest	Recommended for large objects stored externally. Message digest (MD5 or SHA-1) over the bytes of the large object.	O

6.3 Date and timestamp data cells

ID	Description of requirement	M/O
T_6.3-1	Dates and timestamps must be restricted to the years 0001-9999 according to the SQL:2008 specification. This restriction id enforced in the definitions of *dateType* and *dateTimeType*.	M

6.4 Table data

The `table[number].xml` file contains the following table data for this table:

ID	Description of requirement	M/O
T_6.4-1	The table data for each table must be stored in an XML file.	M
T_4.2-2	The *table* file consists of *row* elements containing the data of a line subdivided into the various columns (*c1, c2 ...*). **Example** content – table0.xml <?xml version="1.0" encoding="utf-8"?> <table xsi:schemaLocation="http://www.admin.ch/xmlns/siard/1.0/schema0/table0.xsd table0.xsd" xmlns="http://www.admin.ch/xmlns/siard/1.0/schema0/table0.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <row><c1>1</c1><c2>Chai</c2><c3>1</c3><c4>1</c4> ... <c10>false</c10></row> <row><c1>2</c1><c2>Chang</c2><c3>1</c3><c4>1</c4> ... <c10>false</c10></row> <row><c1>3</c1><c2>Aniseed Syrup</c2><c3>1</c3><c4>2</c4> ... <c10>false</c10></row> ... <row><c1>75</c1><c2>Rhönbräu Klosterbier</c2><c3>12</c3><c4>1</c4> ... <c10>false</c10></row> <row><c1>76</c1><c2>Lakkalikööri</c2><c3>23</c3><c4>1</c4> ... <c10>false</c10></row> <row><c1>77</c1><c2>Frankfurter grüne Soße</c2><c3>12</c3><c4>2</c4> ... <c10>false</c10></row> </table>	M

ID	Description of requirement	M/O
T_6.4-3	If a cell of a column does not contain a value, this may be omitted or empty. **Example** content – table1.xml <?xml version="1.0" encoding="utf-8"?> <table xsi:schemaLocation="http://www.admin.ch/xmlns/siard/1.0/schema0/table1.xsd table1.xsd" xmlns="http://www.admin.ch/xmlns/siard/1.0/schema0/table1.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <row><c1>1</c1><c2>Speedy Express</c2></row> <row><c1>2</c1><c2>United Package</c2><c3/></row> <row><c1>3</c1><c2>Federal Shipping</c2><c3></c3></row> </table>	O
T_6.4-4	If a cell of a column contains a complex value (ARRAY, ROW, UDT), it is represented by a sequence of sub elements of the cell (a1,a2, … for ARRAYs, r1, r2, … for ROWs, u1, u2, … for UDTs) which in turn contain their respective values. These values may again be complex. See appendix D4 for an example.	?
T_6.4-5	If a table contains data of the large object types (BLOB, CLOB, or XML ...) separate files may be produced for these and the storage location of the file is stored instead of the cell content. The decision, when to store large object data in separate files rather than inlining them is at the discretion of the software producing the SIARD archive. To avoid creating empty folders, folders are only created when they are necessary, i.e. contain data. If a large object is stored in a separate file, its cell element must have attributes *file*, *length* and *digest*. Here *file* is a “file:” URI relative to the *lobFolder* element of the column or attribute metadata. The *length* contains the length in bytes (for BLOBs) or characters (for CLOBs or XMLs). The digest records a message digest over the LOB file, making it possible to check integrity of the SIARD archive, even when some LOBs are stored externally. **Example** content – table3.xml <?xml version="1.0" encoding="utf-8"?> <table xsi:schemaLocation="http://www.admin.ch/xmlns/siard/1.0/schema0/table3.xsd table3.xsd" xmlns="http://www.admin.ch/xmlns/siard/1.0/schema0/table3.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <row><c1>1</c1><c2>Beverages</c2><c3>Soft drinks, coffees, teas, beers, and ales</c3><c4 length="10746" file="content/schema0/table3/lob4/record0.bin"/></row> <row><c1>2</c1><c2>Condiments</c2><c3>Sweet and savory sauces, relishes, spreads, and seasonings</c3><c4 length="10746" file="content/schema0/table3/lob4/record1.bin"/></row> ... <row><c1>7</c1><c2>Produce</c2><c3>Dried fruit and bean curd</c3><c4 length="10746" file="content/schema0/table3/lob4/record6.bin"/></row> <row><c1>8</c1><c2>Seafood</c2><c3>Seaweed and fish</c3><c4 length="10746" file="content/schema0/table3/lob4/record7.bin"/></row> </table> ***Recommendation*** *It is strongly recommended, to either inline all large objects in one column or none.* *It is advisable to normalise the lob folders and lob files and to use, for example,* `lob4/` *and* `record0.bin` *or* `record0.txt` *instead of the actual name.*	M

7 Version and validity of the specification

The current version of the Specification is 2.0. The content of the Specification is reviewed periodically by the eCH digital archiving expert group and amended as necessary.

8 Change management process

The change management process for this standard follows [eCH-0150], scenario 3. The *change manager* is the head of the expert group; the *change board* is a committee mandated by the expert group or a subject group.

9 Exclusion of liability / notice regarding third-party rights

eCH standards which are made available to users by the **eCH** association (Verein **eCH**) for use free of charge, or which **eCH** references, only have the status of recommendations. The **eCH** association accepts no liability for decisions or measures taken by the user on the basis of these documents. It is the responsibility of the user to review the documents before using them and obtain advice where necessary. **eCH** standards cannot, nor are they intended to, replace technical, organisational or legal advice in specific cases.

Documents, processes, methods, products and standards referenced in **eCH** standards may be protected as trademarks or by copyright or patent law. It is the sole responsibility of the user to obtain any rights that may be necessary from the individuals and/or organisations that hold them.

Although the **eCH** association takes every care to draw up the **eCH** standards diligently, it can offer no warranty or guarantee that the information and documents made available are up to date, complete, correct or free from errors. The content of **eCH** standards may be amended at any time without notice.

To the extent permitted by law, no liability is accepted in respect of loss or damage incurred by the user as the result of using the **eCH** standards.

10 Copyright

Any person who draws up **eCH** standards retains intellectual property rights in respect thereof. However, such person undertakes to make his/her relevant intellectual property or rights in respect of the intellectual property of others available, to the extent possible, to the respective expert groups and the **eCH** association free of charge for unrestricted use and further development within the context of the association's purpose.

The standards drawn up by the expert groups may be used, disseminated and further developed free of charge and without restriction, provided the authors at **eCH** are cited.

eCH standards are fully documented and free from licence and/or patent restrictions. The associated documentation may be obtained free of charge.

These provisions apply solely to the standards drawn up by **eCH**, and not to standards or products of third parties that are referred to in the **eCH** standards. The standards contain corresponding references to the rights of third parties.

Appendix A – Participation and Review

Hedi Bruggisser, Thurgau State Archives

Georg Büchler, KOST

Alain Dubois, Valais State Archives

Martin Kaiser, KOST

Lambert Kansy, Basel Stadt State Archives

Markus Lischer, Lucerne State Archives

Claire Röthlisberger-Jourdan, KOST

Hartwig Thomas, Enter AG

Andreas Voss, Swiss Federal Archives

Appendix B – Abbreviations and Glossary

Term	Description
AIP	Archival Information Package. AIPs result from SIPs during the process of archiving digital documents. They represent the form of information packages in which digital documents are stored in the digital repository.
Archive	1. Institution or body responsible for cataloguing, keeping and preserving archive records and making them available. 2. Archived documents of an organisation. 3. Building or institution that was constructed or established for the purpose of archiving documents. 4. Term for a file that contains other files. See also archive file and the synonym container file.
Archive records	Refers to documents that have been accepted by the archive for safekeeping, or that are independently archived by other bodies in accordance with the same principles.
Database	A database normally consists of one or more database schemas as well as defined access rights of individual users and roles to certain parts of the database. In SQL:2008 users and roles can be holders of privileges. A relational database therefore consists of a number of structured database objects (e.g. schema, view) and the table content. A database schema is a kind of namespace prefix. A database catalogue contains the metadata of all the schemas in the catalogue. The catalogue level in SQL:2008 corresponds to the database “document” that can be converted into an archival format using SIARD.
DIP	Dissemination Information Package: According to OAIS, a DIP is a container for dossiers that are requested by a user via an ordering procedure.
DNS	Domain Name System, a distributed database that administers the name space in the Internet.
Documents	Documents are all recorded information, irrespective of the medium, that is received or produced in the fulfilment of public duties, as well as all finding aids and supplementary data that are required in order to understand and use this information.
Dossier	All the documents relating to a specific business matter. A dossier basically corresponds to a business matter. However, by combining similar business matters or dividing dossiers into subdossiers, this basic structure can be adapted to meet the corresponding needs. The compilation of dossiers is carried out on the basis of the classification system.
Information package	A conceptual container made up of optional content information and optional associated preservation metadata. It includes packaging information that distinguishes the content information and the package description from each other, identifies them and enables the content information to be searched for.

Term	Description
LOB	“Large object” used generically for cell content of a CLOB, BLOB or XML column which might be represented by a separate file.
Long-term archiving	Storing digital information and maintaining its long-term availability, normally without a time limit. In addition to retaining the bit stream of the archived information it also includes the ability to interpret and display it in human-readable and understandable form at all times.
MD5	Message-Digest Algorithm 5
Metadata	Metadata can be described as “information about primary data” (data about data), since they have a descriptive nature.
OAIS	Open Archival Information System, ISO 14721:2003. A reference model that describes an archive as an organisation in which people and systems work together to preserve information and make it available to a designated community.
Primary data	Primary data are the data that make up the content of documents. Within a SIARD file, the table data perform the function of primary data
Records creator	Refers to the authority or organisational unit that created and managed the documents.
Routines	SQL routines (also known as stored procedures) are mainly important to understanding the view queries in which they can occur as partial expressions.
Schemas	Schemas are containers for the tables, views and routines.
SFA	Swiss Federal Archives
SHA1	Secure Hash Algorithm 1
SIP	Submission Information Package: According to OAIS, SIPs are information packages that are submitted to the archive by the records-creating authorities. They contain digital documents (primary data and metadata).
Tables	Tables consist of a table definition with fields that assign a name and type to each column in the table, datasets that contain the actual table data, an optional primary key, foreign keys that ensure referential integrity, candidate keys that serve to identify a dataset, and constraints that guarantee consistency. Triggers may optionally be defined for a table.
UTF	Unicode Transformation Format
Views	Views are standard queries stored in the database. The query result is a table that also contains fields and data sets.
XSD	XML Schema Definition

Appendix C – List of Standards Used

eCH-0150	eCH-0150 Change und Release Management von eCH-Standards http://www.ech.ch/
SQL:2008	ISO/IEC 9075(1-4,9-11,13,14):2008: Information technology -- Database languages – SQL http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=53681
Unicode	Unicode 6.1.0 Unicode, Inc. http://www.unicode.org/versions/Unicode6.1.0/ (corresponds to ISO/IEC 10646:2012: Information technology -- Universal Coded Character Set (UCS) http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56921)
XML	Extensible Markup Language (XML), 1.1 (Second Edition) W3C Recommendation 16 August 2006, edited in place 29 September 2006 http://www.w3.org/TR/2006/REC-xml11-20060816/ (corresponds to ISO/IEC 19503:2005: Information technology -- XML Metadata Interchange (XMI), http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=32622)
ZIP	ZIP File Format Specification, Version 6.3.3 September 1, 2012 PKWARE Inc. http://www.pkware.com/documents/casestudies/APPNOTE.TXT
RFC 1738	URL specification – in particular the “file:” URL/URI https://www.ietf.org/rfc/rfc1738.txt
RFC 1951	Specification of the “deflate” algorithm. https://www.ietf.org/rfc/rfc1951.txt

Appendix D – XML Schema Definitions

D.1 metadata.xsd

The XML schema definition `metadata.xsd` defines the structure of the `metadata.xml` file in the `header/` folder.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- $Workfile: metadata.xsd $  ======================================
Metadata schema for SIARD-E (SIARD 2.0)
Version    : $Id: metadata.xsd 1205 2010-06-17 16:54:52Z hartwig $
Application: Software-Independent Archival of Relational Databases
Platform   : XML 1.0, XML Schema 2001
Description: This XML schema definition defines the structure
             of the metadata in the SIARD format
=====================================================================
Copyright  : 2007, 2014, 2015, Swiss Federal Archives, Berne, Switzerland
===================================================================== -->
<xs:schema id="metadata"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.bar.admin.ch/xmlns/siard/1.0/metadata.xsd"
  targetNamespace="http://www.bar.admin.ch/xmlns/siard/1.0/metadata.xsd"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

  <!-- root element of an XML file conforming to this XML schema -->
  <xs:element name="siardArchive">
    <xs:complexType>
      <xs:annotation>
        <xs:documentation>
          Root element of meta data of the SIARD archive
        </xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <!-- name of the archived database -->
        <xs:element name="dbname" type="mandatoryString"/>
        <!-- short free form description of the database content -->
        <xs:element name="description" type="xs:string" minOccurs="0"/>
        <!-- name of person responsible for archiving the database -->
        <xs:element name="archiver" type="xs:string" minOccurs="0"/>
        <!-- contact data (telephone number or email address) of archiver -->
        <xs:element name="archiverContact" type="xs:string" minOccurs="0"/>
        <!-- name of data owner (section and institution responsible for data)
             of database when it was archived -->
        <xs:element name="dataOwner" type="mandatoryString"/>
        <!-- time span during which data where entered into the database -->
        <xs:element name="dataOriginTimespan" type="mandatoryString"/>
        <!--  root folder for external files (new in version 2.0) -->
        <xs:element name="lobFolder" type="xs:anyURI"  minOccurs="0"/>
        <!-- name and version of program that generated the metadata file -->
        <xs:element name="producerApplication" type="xs:string" minOccurs="0"/>
        <!-- date of creation of archive (automatically generated by SIARD) -->
        <xs:element name="archivalDate" type="xs:date"/>
        <!-- message digest code over all primary data in folder "content" -->
        <xs:element name="messageDigest" type="xs:string" minOccurs="0" max-
Occurs="unbounded"/>
```

```
        <!-- DNS name of client machine from which SIARD was running for archiving
-->
        <xs:element name="clientMachine" type="xs:string" minOccurs="0"/>
        <!-- name of database product and version from which database originates -
->
        <xs:element name ="databaseProduct" type="xs:string" minOccurs="0"/>
        <!-- connection string used for archiving -->
        <xs:element name="connection" type="xs:string" minOccurs="0"/>
        <!-- database user used for archiving -->
        <xs:element name="databaseUser" type="xs:string" minOccurs="0"/>
        <!--  list of schemas in database  -->
        <xs:element name="schemas" type="schemasType"/>
        <!-- list of users in the archived database -->
        <xs:element name="users" type="usersType"/>
        <!-- list of roles in the archived database -->
        <xs:element name="roles" type="rolesType" minOccurs="0"/>
        <!-- list of privileges in the archived database -->
        <xs:element name="privileges" type="privilegesType" minOccurs="0"/>
      </xs:sequence>
      <!-- constraint: version number must be 1.0 or 2.0 -->
      <xs:attribute name="version" type="versionType" use="required" />
    </xs:complexType>
  </xs:element>

  <!-- complex type schemas -->
  <xs:complexType name="schemasType">
    <xs:annotation>
      <xs:documentation>
        List of schemas
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="schema" type="schemaType" minOccurs="1" maxOccurs="un-
bounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type schema -->
  <xs:complexType name="schemaType">
    <xs:annotation>
      <xs:documentation>
        Schema element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the schema -->
      <xs:element name="name" type="xs:string" />
      <!-- archive name of the schema folder -->
      <xs:element name="folder" type="fsName"/>
      <!-- description of the schema's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
      <!--  list of advanced and structured types in the schema (new in version
2.0) -->
      <xs:element name="types" type="typesType" minOccurs="0"/>
      <!-- list of tables in the schema -->
      <xs:element name="tables" type="tablesType"/>
      <!-- list of views in the schema -->
      <xs:element name="views" type="viewsType" minOccurs="0"/>
```

```
      <!-- list of routines in the schema -->
      <xs:element name="routines" type="routinesType" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type types (new in version 2.0) -->
  <xs:complexType name="typesType">
    <xs:annotation>
      <xs:documentation>
        List of advanced or structured data types types
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="type" type="typeType" minOccurs="1" maxOccurs="unbounded"
/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type type (new in version 2.0) -->
  <xs:complexType name="typeType">
    <xs:annotation>
      <xs:documentation>
        Advanced or structured data tape type
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!--  name of data type  -->
      <xs:element name="name" type="xs:string"/>
      <!--  category of data type -->
      <xs:element name="category" type="categoryType"/>
      <!--  schema of supertype -->
      <xs:element name="underSchema" type="xs:string" minOccurs="0"/>
      <!--  name of supertype -->
      <xs:element name="underType" type="xs:string" minOccurs="0"/>
      <!-- instantiability if data type (never true for DISTINCT) -->
      <xs:element name="instantiable" type="xs:boolean"/>
      <!-- finality (always true for DISTINCT, never true for structured UDTs) -->
      <xs:element name="final" type="xs:boolean"/>
      <!--  primitive base SQL:2008 type of (DISTINCT, ARRAY) type -->
      <xs:element name="base" type="xs:string" minOccurs="0"/>
      <!--  SQL_1999 cardinality of ARRAY type -->
      <xs:element name="cardinality" type="xs:integer" minOccurs="0"/>
      <!--  alternatively list of attributes (ROW, UDT) -->
      <xs:element name="attributes" type="attributesType" minOccurs="0"/>
      <!-- description of the parameter's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type attributes (new in version 2.0) -->
  <xs:complexType name="attributesType">
    <xs:annotation>
      <xs:documentation>
        List of attributes of a type
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
```

```
      <xs:element name="attribute" type="attributeType" minOccurs="1"
maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type attribute (new in version 2.0) -->
  <xs:complexType name="attributeType">
    <xs:annotation>
      <xs:documentation>
        Attribute of a type
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the attribute -->
      <xs:element name="name" type="xs:string" />
      <xs:choice> <!-- new in version 2.0: either built-in or structured -->
        <xs:sequence>
          <!-- SQL:2008 data type of the column -->
          <xs:element name="type" type="builtinTypeType" />
              <!-- original data type of the column -->
              <xs:element name="typeOriginal" type="xs:string" minOccurs="0"/>
        </xs:sequence>
            <xs:sequence>
              <!-- SQL:2008 schema of advanced or structured data type of the at-
tribute -->
              <xs:element name="typeSchema" type="xs:string" minOccurs="0" />
              <!-- SQL:2008 name of advanced or structured data type of the at-
tribute -->
              <xs:element name="typeName" type="xs:string" />
              <!-- SQL:2008 attribute list of the column (recursive) -->
              <xs:element name="attributes" type="attributesType" minOccurs="0"/>
            </xs:sequence>
      </xs:choice>
      <!-- default value -->
      <xs:element name="defaultValue" type="xs:string" minOccurs="0"/>
      <!-- description of the attributes's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type tables -->
  <xs:complexType name="tablesType">
    <xs:annotation>
      <xs:documentation>
        List of tables
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="table" type="tableType" minOccurs="1" maxOccurs="un-
bounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type table -->
  <xs:complexType name="tableType">
    <xs:annotation>
      <xs:documentation>
        Table element in siardArchive
```

```
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the table -->
      <xs:element name="name" type="xs:string"/>
      <!-- archive name of the table folder -->
      <xs:element name="folder" type="fsName"/>
      <!-- description of the table's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
      <!-- list of columns of the table -->
      <xs:element name="columns" type="columnsType"/>
      <!--  primary key -->
      <xs:element name="primaryKey" type="primaryKeyType" minOccurs="0"/>
      <!--  foreign keys  -->
      <xs:element name="foreignKeys" type="foreignKeysType" minOccurs="0"/>
      <!--  candidate keys (unique constraints)  -->
      <xs:element name="candidateKeys" type="candidateKeysType" minOccurs="0"/>
      <!-- list of (check) constraints -->
      <xs:element name="checkConstraints" type="checkConstraintsType" minOc-
curs="0"/>
      <!--  list of triggers  -->
      <xs:element name="triggers" type="triggersType" minOccurs="0"/>
      <!--  number of rows in the table -->
      <xs:element name="rows" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type views -->
  <xs:complexType name="viewsType">
    <xs:annotation>
      <xs:documentation>
        List of views
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="view" type="viewType" minOccurs="1" maxOccurs="unbounded"
/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type view -->
  <xs:complexType name="viewType">
    <xs:annotation>
      <xs:documentation>
        View element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the view -->
      <xs:element name="name" type="xs:string" />
      <!-- SQL query string defining the view -->
      <xs:element name="query" type="xs:string" minOccurs="0"/>
      <!-- original query string defining the view -->
      <xs:element name="queryOriginal" type="xs:string" minOccurs="0"/>
      <!-- description of the view's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
      <!-- list of columns of the view -->
      <xs:element name="columns" type="columnsType"/>
```

```
      <!--  number of rows in the view - added in 2014! -->
      <xs:element name="rows" type="xs:integer" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type columns -->
  <xs:complexType name="columnsType">
    <xs:annotation>
      <xs:documentation>
        List of columns
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="column" type="columnType" minOccurs="1" maxOccurs="un-
bounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type column -->
  <xs:complexType name="columnType">
    <xs:annotation>
      <xs:documentation>
        Column element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the column -->
      <xs:element name="name" type="xs:string" />
      <!-- archive name of the internally stored LOBs within containing
           element -->
      <xs:element name="folder" type="fsName" minOccurs="0"/>
      <!--  folder for LOBs relative to lobFolder of nearest containing
            element for internally or externally stored LOBs
            (new in version 2.0) -->
      <xs:element name="lobFolder" type="xs:anyURI"  minOccurs="0"/>
      <xs:choice> <!-- new in version 2.0: either built-in or structured -->
        <xs:sequence>
          <!-- SQL:2008 data type of the column -->
          <xs:element name="type" type="builtinTypeType" />
              <!-- original data type of the column -->
              <xs:element name="typeOriginal" type="xs:string" minOccurs="0"/>
                  <!-- nullability (default: true) -->
                  <xs:element name="nullable" type="xs:boolean" minOccurs="0"/>
        </xs:sequence>
            <xs:sequence> <!--  new in version 2.0 -->
              <!-- SQL:2008 schema of UDT name of the column (new in version 2.0)
-->
              <xs:element name="typeSchema" type="xs:string" minOccurs="0" />
              <!-- SQL:2008 name of UDT of the column (new in version 2.0) -->
              <xs:element name="typeName" type="xs:string" />
              <!-- SQL:2008 attribute list of the column (new in version 2.0) -->
              <xs:element name="fields" type="fieldsType" minOccurs="0"/>
            </xs:sequence>
      </xs:choice>
      <!-- default value -->
      <xs:element name="defaultValue" type="xs:string" minOccurs="0"/>
      <!-- unique, references, check column constraints
           are stored as table constraints -->
```

```
      <!--  mimeType makes sense only for LOBs and is only informatory-->
      <xs:element name="mimeType" type="xs:string" minOccurs="0"/>
      <!-- description of the column's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type fields -->
  <xs:complexType name="fieldsType">
    <xs:annotation>
      <xs:documentation>
        List of fields of a column or field
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="field" type="fieldType" minOccurs="1" maxOccurs="un-
bounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type for type of a column or a field -->
  <xs:complexType name="fieldType">
    <xs:annotation>
      <xs:documentation>
        Field element describing the type of a field
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- archive name of the internally stored LOBs within containing
           element -->
      <xs:element name="folder" type="fsName" minOccurs="0"/>
      <!--  folder for LOBs relative to lobFolder of nearest containing
            element for internally or externally stored LOBs
            (new in version 2.0) -->
      <xs:element name="lobFolder" type="xs:anyURI"  minOccurs="0"/>
      <!-- SQL:2008 attribute list of the column (new in version 2.0) -->
      <xs:element name="fields" type="fieldsType" minOccurs="0"/>
      <!--  mimeType makes sense only for LOBs and is only informatory-->
      <xs:element name="mimeType" type="xs:string" minOccurs="0"/>
      <!-- description of the field's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type primaryKey -->
  <xs:complexType name="primaryKeyType">
    <xs:annotation>
      <xs:documentation>
        primaryKey element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the primary key -->
      <xs:element name="name" type="xs:string" minOccurs="0" />
       <!-- description of the primary key's meaning and content -->
       <xs:element name="description" type="xs:string" minOccurs="0"/>
      <!-- columns belonging to the primary key -->
```

```
      <xs:element name="column" type="xs:string"  minOccurs="1" maxOccurs="un-
bounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type foreignKeys -->
  <xs:complexType name="foreignKeysType">
    <xs:annotation>
      <xs:documentation>
        List of foreign key constraints
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="foreignKey" type="foreignKeyType" minOccurs="1" max-
Occurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type foreignKey -->
  <xs:complexType name="foreignKeyType">
    <xs:annotation>
      <xs:documentation>
        foreignKey element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the foreign key -->
      <xs:element name="name" type="xs:string" />
      <!--  referenced schema -->
      <xs:element name="referencedSchema" type="xs:string"/>
      <!-- referenced table -->
      <xs:element name="referencedTable" type="xs:string"/>
      <!--  references -->
      <xs:element name="reference" type="referenceType" minOccurs="1" max-
Occurs="unbounded"/>
      <!-- match type (FULL, PARTIAL, SIMPLE) -->
      <xs:element name="matchType" type="matchTypeType" minOccurs="0"/>
      <!-- ON DELETE action e.g. ON DELETE CASCADE -->
      <xs:element name="deleteAction" type="xs:string" minOccurs="0"/>
      <!-- ON UPDATE action e.g. ON UPDATE SET DEFAULT -->
      <xs:element name="updateAction" type="xs:string" minOccurs="0"/>
      <!-- description of the foreign key's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type reference -->
  <xs:complexType name="referenceType">
    <xs:annotation>
      <xs:documentation>
        reference element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- referencing column -->
      <xs:element name="column" type="xs:string"/>
      <!-- referenced column (table.column) -->
      <xs:element name="referenced" type="xs:string"/>
```

```
    </xs:sequence>
  </xs:complexType>

  <!-- complex type candidateKeys -->
  <xs:complexType name="candidateKeysType">
    <xs:annotation>
      <xs:documentation>
        List of candidate key (unique) constraints
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="candidateKey" type="candidateKeyType" minOccurs="1" max-
Occurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type candidateKey -->
  <xs:complexType name="candidateKeyType">
    <xs:annotation>
      <xs:documentation>
        candiate key (unique) element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the candidate key -->
      <xs:element name="name" type="xs:string"/>
       <!-- description of the candidate key's meaning and content -->
       <xs:element name="description" type="xs:string" minOccurs="0"/>
      <!-- columns belonging to the candidate key -->
      <xs:element name="column" type="xs:string"  minOccurs="1" maxOccurs="un-
bounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type check constraints -->
  <xs:complexType name="checkConstraintsType">
    <xs:annotation>
      <xs:documentation>
        List of check constraints
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="checkConstraint" type="checkConstraintType" minOccurs="1"
maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type check constraint -->
  <xs:complexType name="checkConstraintType">
    <xs:annotation>
      <xs:documentation>
        Check constraint element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the constraint -->
      <xs:element name="name" type="xs:string"/>
      <!-- check condition -->
```

```
      <xs:element name="condition" type="xs:string"/>
      <!-- description of the constraint's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type triggers -->
  <xs:complexType name="triggersType">
    <xs:annotation>
      <xs:documentation>
        List of triggers
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="trigger" type="triggerType" minOccurs="1" maxOccurs="un-
bounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type trigger -->
  <xs:complexType name="triggerType">
    <xs:annotation>
      <xs:documentation>
        Trigger element in siardArchive
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of the trigger -->
      <xs:element name="name" type="xs:string" />
      <!--  action time -->
      <xs:element name="actionTime" type="actionTimeType"/>
      <!--  trigger event INSERT, DELETE, UPDATE [OF <trigger column list>] -->
      <xs:element name="triggerEvent" type="xs:string"/>
      <!--  alias list <old or new values alias> -->
      <xs:element name="aliasList" type="xs:string" minOccurs="0"/>
      <!--  triggered action -->
      <xs:element name="triggeredAction" type="xs:string"/>
      <!-- description of the trigger's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type routines -->
  <xs:complexType name="routinesType">
    <xs:annotation>
      <xs:documentation>
        List of routines
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="routine" type="routineType" minOccurs="1" maxOccurs="un-
bounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type routine -->
  <xs:complexType name="routineType">
    <xs:annotation>
```

```
      <xs:documentation>
        Routine
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- database name of routine in schema -->
      <xs:element name="name" type="xs:string"/>
      <!-- description of the routines's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
      <!-- original source code (VBA, PL/SQL, ...) defining the routine -->
      <xs:element name="source" type="xs:string" minOccurs="0"/>
      <!--  SQL:2008 body of routine  -->
      <xs:element name="body" type="xs:string" minOccurs="0"/>
      <!--  routine characteristic -->
      <xs:element name="characteristic" type="xs:string" minOccurs="0"/>
      <!--  SQL:2008 data type of the return value (for functions) -->
      <xs:element name="returnType" type="xs:string" minOccurs="0"/>
      <!--  list of parameters -->
      <xs:element name="attributes" type="parametersType" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type parameters -->
  <xs:complexType name="parametersType">
    <xs:annotation>
      <xs:documentation>
        List of parameters of a routine
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="parameter" type="parameterType" minOccurs="1" max-
Occurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type parameter -->
  <xs:complexType name="parameterType">
    <xs:annotation>
      <xs:documentation>
        Parameter of a routine
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!--  name of parameter  -->
      <xs:element name="name" type="xs:string"/>
      <!--  mode of parameter (IN, OUT, INOUT) -->
      <xs:element name="mode" type="xs:string"/>
      <xs:choice> <!-- new in version 2.0: either built-in or structured -->
        <xs:sequence>
          <!-- SQL:2008 data type of the column -->
          <xs:element name="type" type="builtinTypeType" />
              <!-- original data type of the column -->
              <xs:element name="typeOriginal" type="xs:string" minOccurs="0"/>
        </xs:sequence>
            <xs:sequence> <!--  new in version 2.0 -->
              <!-- SQL:2008 schema of UDT name of the column (new in version 2.0)
-->
              <xs:element name="typeSchema" type="xs:string" minOccurs="0" />
```

```
              <!-- SQL:2008 name of UDT of the column (new in version 2.0) -->
              <xs:element name="typeName" type="xs:string" />
              <!-- SQL:2008 attribute list of the column (new in version 2.0) -->
              <xs:element name="parameterFields" type="parameterFieldsType" minOc-
curs="0"/>
            </xs:sequence>
      </xs:choice>
      <!-- description of the parameter's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type parameterFields (new in version 2.0) -->
  <xs:complexType name="parameterFieldsType">
    <xs:annotation>
      <xs:documentation>
        List of fields of a parameter
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="parameterField" type="parameterFieldType" minOccurs="1"
maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type for type of a parameter or a parameter field (new in version
2.0) -->
  <xs:complexType name="parameterFieldType">
    <xs:annotation>
      <xs:documentation>
        Field element describing the type of a parameter or a parameter field
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- SQL:2008 attribute list of the column (new in version 2.0) -->
      <xs:element name="parameterFields" type="parameterFieldsType" minOc-
curs="0"/>
      <!-- description of the parameter field's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type users -->
  <xs:complexType name="usersType">
    <xs:annotation>
      <xs:documentation>
        List of users
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="user" type="userType" minOccurs="1" maxOccurs="unbounded"
/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type user -->
  <xs:complexType name="userType">
    <xs:annotation>
```

```
      <xs:documentation>
        User
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- user name -->
      <xs:element name="name" type="xs:string"/>
      <!-- description of the user's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type roles -->
  <xs:complexType name="rolesType">
    <xs:annotation>
      <xs:documentation>
        List of roles
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="role" type="roleType" minOccurs="1" maxOccurs="unbounded"
/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type role -->
  <xs:complexType name="roleType">
    <xs:annotation>
      <xs:documentation>
        Role
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- role name -->
      <xs:element name="name" type="xs:string"/>
      <!-- role ADMIN (user or role) -->
      <xs:element name="admin" type="xs:string"/>
      <!-- description of the role's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- complex type privileges -->
  <xs:complexType name="privilegesType">
    <xs:annotation>
      <xs:documentation>
        List of grants
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="privilege" type="privilegeType" minOccurs="1"
maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- complex type privilege -->
  <xs:complexType name="privilegeType">
    <xs:annotation>
```

```
      <xs:documentation>
        Grant (incl. grant of role)
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- privilege type (incl. ROLE privilege or "ALL PRIVILEGES" -->
      <xs:element name="type" type="xs:string"/>
      <!-- privilege object (may be omitted for ROLE privilege) -->
      <xs:element name="object" type="xs:string" minOccurs="0"/>
      <!-- GRANTED BY -->
      <xs:element name="grantor" type="xs:string"/>
      <!-- user list of users or roles or single value "PUBLIC" -->
      <xs:element name="grantee" type="xs:string"/>
      <!-- optional option "GRANT" or "ADMIN" -->
      <xs:element name="option" type="privOptionType" minOccurs="0"/>
      <!-- description of the grant's meaning and content -->
      <xs:element name="description" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>


  <xs:simpleType name="builtinTypeType">
    <xs:annotation>
      <xs:documentation>
        builtinTypeType is constrained to valid SQL:2008 data type values
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <!-- exact numerics (BIGINT from SQL:2008) -->
      <xs:pattern value="INTEGER|INT|SMALLINT|BIGINT"/>
      <xs:pattern value="NUMERIC(\([1-9]\d*(,[1-9]\d*)?\))?"/>
      <xs:pattern value="DECIMAL(\([1-9]\d*(,[1-9]\d*)?\))?"/>
      <!-- approximate numerics -->
      <xs:pattern value="REAL|DOUBLE PRECISION"/>
      <xs:pattern value="FLOAT(\([1-9]\d*\))?"/>
      <!-- character strings -->
      <xs:pattern value="(CHARACTER|CHAR)(\([1-9]\d*\))?"/>
      <xs:pattern value="(CHARACTER VARYING|CHAR VARYING|VARCHAR)(\([1-9]\d*\))"/>
      <xs:pattern value="(CHARACTER LARGE OBJECT|CLOB)(\([1-9]\d*(K|M|G)?\))?"/>
      <xs:pattern value="(NATIONAL CHARACTER|NATIONAL CHAR|NCHAR)(\([1-
9]\d*\))?"/>
      <xs:pattern value="(NATIONAL CHARACTER VARYING|NATIONAL CHAR VARYING|NCHAR
VARYING)(\([1-9]\d*\))"></xs:pattern>
      <xs:pattern value="(NATIONAL CHARACTER LARGE OBJECT|NCHAR LARGE
OBJECT|NCLOB)(\([1-9]\d*(K|M|G)?\))?"/>
      <xs:pattern value="XML"/>
      <!-- BIT and BINARY strings -->
      <xs:pattern value="BIT(\([1-9]\d*\))?"/>
      <xs:pattern value="BIT VARYING(\([1-9]\d*\))"/>
      <xs:pattern value="(BINARY LARGE OBJECT|BLOB)(\([1-9]\d*(K|M|G)?\))?"/>
      <!-- BINARY strings from SQL:2008 -->
      <xs:pattern value="BINARY(\([1-9]\d*\))?"/>
      <xs:pattern value="(BINARY VARYING|VARBINARY)(\([1-9]\d*\))"/>
      <!-- datetimes -->
      <xs:pattern value="DATE"/>
      <xs:pattern value="(TIME|TIME WITH TIME ZONE)(\([1-9]\d*\))?"/>
      <xs:pattern value="(TIMESTAMP|TIMESTAMP WITH TIME ZONE)(\((0|([1-
9]\d*))\))?"/>
```

```
      <!--  intervals -->
      <xs:pattern value="INTERVAL (YEAR|MONTH|DAY|HOUR|MINUTE|SECOND)(\([1-
9]\d*\))?( TO (MONTH|DAY|HOUR|MINUTE|SECOND)(\([1-9]\d*\))?)?"/>
      <!-- BOOLEAN -->
      <xs:pattern value="BOOLEAN"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- simple type for version number -->
  <xs:simpleType name="versionType">
    <xs:annotation>
      <xs:documentation>
        versionType must be constrained to "1.0" or "2.0"
        for conformity with this XML schema
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="collapse"/>
      <xs:enumeration value="1.0"/>
      <xs:enumeration value="2.0"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- simple type for privilege option -->
  <xs:simpleType name="privOptionType">
    <xs:annotation>
      <xs:documentation>
        privOptionType must be "ADMIN" or "GRANT"
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="collapse"/>
      <xs:enumeration value="ADMIN"/>
      <xs:enumeration value="GRANT"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- simple type for mandatory string
       which must contain at least 1 character -->
  <xs:simpleType name="mandatoryString">
    <xs:annotation>
      <xs:documentation>
        mandatoryString must contain at least 1 character
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="preserve"/>
      <xs:minLength value="1" />
    </xs:restriction>
  </xs:simpleType>

  <!-- simple type of a filesystem (file or folder) name -->
  <xs:simpleType name="fsName">
    <xs:annotation>
      <xs:documentation>
        fsNames may only consist of ASCII characters and digits
        and must start with a non-digit
      </xs:documentation>
```

```
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z]|[A-Z])([a-z]|[A-Z]|[0-9]).*" />
      <xs:minLength value="1" />
    </xs:restriction>
  </xs:simpleType>

  <!-- simple type for action time of a trigger -->
  <xs:simpleType name="actionTimeType">
    <xs:annotation>
      <xs:documentation>
        actionTime is BEFORE or AFTER
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="BEFORE" />
      <xs:enumeration value="AFTER" />
    </xs:restriction>
  </xs:simpleType>

  <!-- simple type for match type of a foreign key -->
  <xs:simpleType name="matchTypeType">
    <xs:annotation>
      <xs:documentation>
        matchType is FULL, PARTIAL or SIMPLE
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="FULL" />
      <xs:enumeration value="PARTIAL" />
      <xs:enumeration value="SIMPLE" />
    </xs:restriction>
  </xs:simpleType>

  <!-- simple type for the category of a column or a parameter (new in version
2.0) -->
  <xs:simpleType name="categoryType">
    <xs:annotation>
      <xs:documentation>
        category of advanced or structured data types is "distinct",
        "row", "array", or "udt"
        for conformity with this XLM schema
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="collapse"/>
      <xs:enumeration value="distinct"/>
      <xs:enumeration value="row"/>
      <xs:enumeration value="array"/>
      <xs:enumeration value="udt"/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

D.2 Example of a `metadata.xml`

The following is an example of a metadata description of a database that is compliant with the XML schema for SIARD:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="metadata.xsl"?>
<siardArchive
   xmlns="http://www.bar.admin.ch/xmlns/siard/1.0/metadata.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.bar.admin.ch/xmlns/siard/1.0/metadata.xsd
metadata.xsd"
   version="2.0">
  <dbname>SIARD Format 2 with SQL:2008 Standard Types</dbname>
  <dataOwner>Enter AG, Zurich</dataOwner>
  <dataOriginTimespan>2015</dataOriginTimespan>
  <lobFolder>file:///D:/Projekte/SIARD/SIARD%20Suite/</lobFolder>
  <producerApplication>Manually constructed</producerApplication>
  <archivalDate>2015-01-19</archivalDate>
  <messageDigest>MD5D1C411FC45542DCA86EEB1CC9B4B596C</messageDigest>
  <clientMachine>celsius.enterag.ch</clientMachine>
  <databaseProduct>SQL:2011</databaseProduct>
  <connection>jdbc:manually:SQL:2011</connection>
  <databaseUser>SIARDLOGIN</databaseUser>
  <schemas>
    <schema>
      <name>SIARDSCHEMA</name>
      <folder>schema0</folder>

      <!-- version 2.0 advanced or structured types -->
      <types>
        <type>
          <name>TDISTINCT</name>
          <category>distinct</category>
          <instantiable>false</instantiable>
          <final>true</final>
          <base>INTEGER</base>
          <description>Example of a DISTINCT type</description>
        </type>
        <type>
          <name>TROW</name>
          <category>row</category>
          <instantiable>true</instantiable>
          <final>true</final>
          <attributes>
            <attribute>
              <name>TAPEID</name>
              <type>INTEGER</type>
              <defaultValue>0</defaultValue>
            </attribute>
            <attribute>
              <name>TRANSCRIPTION</name>
              <type>CLOB</type>
            </attribute>
            <attribute>
              <name>SOUND</name>
```

```
        <type>BLOB</type>
      </attribute>
    </attributes>
    <description>Example of a ROW type</description>
  </type>
  <type>
    <name>TARRAY</name>
    <category>array</category>
    <instantiable>false</instantiable>
    <final>true</final>
    <base>CHARACTER VARYING(255)</base>
    <cardinality>4</cardinality>
    <description>Example of an ARRAY type</description>
  </type>
  <type>
    <name>TUDT</name>
    <category>udt</category>
    <instantiable>true</instantiable>
    <final>true</final>
    <attributes>
      <attribute>
        <name>ID</name>
        <type>INTEGER</type>
      </attribute>
      <attribute>
        <name>NESTEDROW</name>
        <typeName>TROW</typeName>
        <attributes>
          <attribute>
            <name>first</name>
            <type>INTEGER</type>
          </attribute>
          <attribute>
            <name>second</name>
            <type>VARCHAR(255)</type>
          </attribute>
        </attributes>
      <description>Example of nested attributes</description>
      </attribute>
    </attributes>
    <description>Example of a UDT</description>
  </type>
</types>

<tables>
  <!--  version 1.0 table -->
  <table>
    <name>TABLETEST1</name>
    <folder>table0</folder>
    <description />
    <columns>
      <!-- binary types -->
      <column>
        <name>CBIT</name>
        <type>BIT</type>
        <typeOriginal>BIT</typeOriginal>
        <nullable>true</nullable>
      </column>
```

```
<column>
  <name>CBIT_32</name>
  <type>BIT(32)</type>
  <typeOriginal>BIT(32)</typeOriginal>
  <nullable>true</nullable>
</column>
<column>
  <name>CBIT_VARYING_160</name>
  <type>BIT VARYING(160)</type>
  <typeOriginal>BIT VARYING(160)</typeOriginal>
  <nullable>true</nullable>
</column>
<column>
  <name>CBINARY</name>
  <type>BINARY</type>
  <typeOriginal>BINARY</typeOriginal>
  <nullable>true</nullable>
</column>
<column>
  <name>CBINARY_5</name>
  <type>BINARY(5)</type>
  <typeOriginal>BINARY(5)</typeOriginal>
  <nullable>true</nullable>
</column>
<column>
  <name>CBINARY_VARYING_32</name>
  <type>BINARY VARYING(32)</type>
  <typeOriginal>BINARY VARYING(32)</typeOriginal>
  <nullable>true</nullable>
</column>
<column>
  <name>CBINARY_LARGE_OBJECT</name>
  <!-- <folder>lob8</folder>  -->
  <lobFolder>schema0/table0/lob8/</lobFolder>
  <type>BINARY LARGE OBJECT</type>
  <typeOriginal>BINARY LARGE OBJECT</typeOriginal>
  <nullable>true</nullable>
</column>
<!-- character types -->
<column>
  <name>CCHARACTER</name>
  <type>CHARACTER</type>
  <typeOriginal>CHARACTER</typeOriginal>
  <nullable>false</nullable>
</column>
<column>
  <name>CCHARACTER_5</name>
  <type>CHARACTER(5)</type>
  <typeOriginal>CHARACTER(5)</typeOriginal>
  <nullable>true</nullable>
</column>
<column>
  <name>CCHARACTER_VARYING_32</name>
  <type>CHARACTER VARYING(32)</type>
  <typeOriginal>CHARACTER VARYING(32)</typeOriginal>
  <nullable>true</nullable>
</column>
<column>
```

```
    <name>CCHARACTER_LARGE_OBJECT</name>
    <!-- <folder>lob12</folder> -->
    <lobFolder>schema0/table0/lob12/</lobFolder>
    <type>CHARACTER LARGE OBJECT</type>
    <typeOriginal>CHARACTER LARGE OBJECT</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>CNATIONAL_CHARACTER</name>
    <type>NATIONAL CHARACTER</type>
    <typeOriginal>NATIONAL_CHARACTER</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>CNATIONAL_CHARACTER_5</name>
    <type>NATIONAL CHARACTER(5)</type>
    <typeOriginal>NATIONAL_CHARACTER(5)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>CNATIONAL_CHARACTER_VARYING_32</name>
    <type>NATIONAL CHARACTER VARYING(32)</type>
    <typeOriginal>NATIONAL_CHARACTER VARYING(32)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>CNATIONAL_CHARACTER_LARGE_OBJECT</name>
    <!--  <folder>lob16</folder> -->
    <lobFolder>schema0/table0/lob16/</lobFolder>
    <type>NATIONAL CHARACTER LARGE OBJECT</type>
    <typeOriginal>NATIONAL CHARACTER LARGE OBJECT</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>CXML</name>
    <!-- <folder>lob17</folder> -->
    <lobFolder>schema0/table0/lob17/</lobFolder>
    <type>XML</type>
    <typeOriginal>XML</typeOriginal>
    <nullable>true</nullable>
  </column>
  <!-- integer types -->
  <column>
    <name>CSMALLINT</name>
    <type>SMALLINT</type>
    <typeOriginal>SMALLINT</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>CINTEGER</name>
    <type>INTEGER</type>
    <typeOriginal>INTEGER</typeOriginal>
    <nullable>false</nullable>
  </column>
  <column>
    <name>CBIGINT</name>
    <type>BIGINT</type>
    <typeOriginal>BIGINT</typeOriginal>
```

```
    <nullable>true</nullable>
  </column>
  <!-- fixed numeric types -->
  <column>
    <name>CDECIMAL</name>
    <type>DECIMAL</type>
    <typeOriginal>DECIMAL</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>CDECIMAL_3</name>
    <type>DECIMAL(3)</type>
    <typeOriginal>DECIMAL(3)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>CDECIMAL_5_2</name>
    <type>DECIMAL(5,2)</type>
    <typeOriginal>DECIMAL(5,2)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>CNUMERIC</name>
    <type>NUMERIC</type>
    <typeOriginal>NUMERIC</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>CNUMERIC_3</name>
    <type>NUMERIC(3)</type>
    <typeOriginal>NUMERIC(3)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>CNUMERIC_5_2</name>
    <type>NUMERIC(5,2)</type>
    <typeOriginal>NUMERIC(5,2)</typeOriginal>
    <nullable>true</nullable>
  </column>
  <!--  approximate numerics types -->
  <column>
    <name>CREAL</name>
    <type>REAL</type>
    <typeOriginal>REAL</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>CFLOAT</name>
    <type>FLOAT</type>
    <typeOriginal>FLOAT</typeOriginal>
    <nullable>true</nullable>
  </column>
  <column>
    <name>CFLOAT_13</name>
    <type>FLOAT(13)</type>
    <typeOriginal>FLOAT(13)</typeOriginal>
    <nullable>true</nullable>
  </column>
```

```
      <column>
        <name>CDOUBLE_PRECISION</name>
        <type>DOUBLE PRECISION</type>
        <typeOriginal>DOUBLE PRECISION</typeOriginal>
        <nullable>true</nullable>
      </column>
      <!-- date and time types  -->
      <column>
        <name>CDATE</name>
        <type>DATE</type>
        <typeOriginal>DATE</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>CTIME</name>
        <type>TIME</type>
        <typeOriginal>TIME</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>CTIME_5</name>
        <type>TIME(5)</type>
        <typeOriginal>TIME(5)</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>CTIMESTAMP</name>
        <type>TIMESTAMP</type>
        <typeOriginal>TIMESTAMP</typeOriginal>
        <nullable>true</nullable>
      </column>
      <column>
        <name>CTIMESTAMP_7</name>
        <type>TIMESTAMP(7)</type>
        <typeOriginal>TIMESTAMP(7)</typeOriginal>
        <nullable>true</nullable>
      </column>
    </columns>
    <primaryKey>
      <name>TABLETEST1PK</name>
      <column>CCHARACTER</column>
      <column>CINTEGER</column>
    </primaryKey>
    <rows>1</rows>
  </table>

  <!--  advanced table -->
  <table>
    <name>TABLETEST2</name>
    <folder>table1</folder>
    <description />
    <columns>
      <column>
        <name>ID</name>
        <type>INTEGER</type>
        <nullable>false</nullable>
      </column>
      <column>
```

```
    <name>CDISTINCT</name>
    <typeName>TDISTINT</typeName>
  </column>
  <column>
    <name>CROW</name>
    <lobFolder>lob2</lobFolder>
    <typeName>TROW</typeName>
    <fields>
      <!-- TAPEID -->
      <field>
        <description>Tape ID</description>
      </field>
      <!-- TRANSCRIPTION -->
      <field>
        <lobFolder>field1</lobFolder>
        <description>Tape transcription</description>
      </field>
      <!-- SOUND -->
      <field>
        <lobFolder>field2</lobFolder>
        <mimeType>audio/mpeg</mimeType>
        <description>Digitized tape</description>
      </field>
    </fields>
    <description>Tape recordings</description>
  </column>
  <column>
    <name>CARRAY</name>
    <typeName>TARRAY</typeName>
    <description>Up to 4 fields for complex foreign names</description>
  </column>
  <column>
    <name>CUDT</name>
    <lobFolder>lob4</lobFolder>
    <typeName>TUDT</typeName>
    <fields>
      <!-- ID -->
      <field></field>
      <!-- NESTEDROW -->
      <field>
        <lobFolder>field1</lobFolder>
        <fields>
          <!-- TAPEID -->
          <field>
            <description>Tape ID</description>
          </field>
          <!-- TRANSCRIPTION -->
          <field>
            <lobFolder>field1</lobFolder>
            <description>Tape transcription</description>
          </field>
          <!-- SOUND -->
          <field>
            <lobFolder>field2</lobFolder>
            <mimeType>audio/mpeg</mimeType>
            <description>Digitized tape</description>
          </field>
        </fields>
```

```
                <description>Tape recordings nested</description>
              </field>
            </fields>
            <description>UDT type with field of advanced type</description>
          </column>
        </columns>
        <primaryKey>
          <name>TABLETEST2PK</name>
          <column>ID</column>
        </primaryKey>
        <rows>1000000</rows>
      </table>

    </tables>
  </schema>
</schemas>
<users>
  <user>
    <name>SIARDLOGIN</name>
  </user>
  <user>
    <name>SIARDUSER</name>
  </user>
</users>
<roles>
  <role>
    <name>public</name>
    <admin />
  </role>
</roles>
<privileges>
  <privilege>
    <type>UPDATE</type>
    <object>TABLE SIARDSCHEMA.TABLETEST2</object>
    <grantor>dbo</grantor>
    <grantee>SIARDUSER</grantee>
    <option>ADMIN</option>
  </privilege>
  <privilege>
    <type>INSERT</type>
    <object>TABLE SIARDSCHEMA.TABLETEST2</object>
    <grantor>dbo</grantor>
    <grantee>SIARDUSER</grantee>
    <option>ADMIN</option>
  </privilege>
  <privilege>
    <type>REFERENCES</type>
    <object>TABLE SIARDSCHEMA.TABLETEST2</object>
    <grantor>dbo</grantor>
    <grantee>SIARDUSER</grantee>
    <option>ADMIN</option>
  </privilege>
  <privilege>
    <type>SELECT</type>
    <object>TABLE SIARDSCHEMA.TABLETEST2</object>
    <grantor>dbo</grantor>
    <grantee>SIARDUSER</grantee>
    <option>ADMIN</option>
```

```
    </privilege>
    <privilege>
      <type>DELETE</type>
      <object>TABLE SIARDSCHEMA.TABLETEST2</object>
      <grantor>dbo</grantor>
      <grantee>SIARDUSER</grantee>
      <option>ADMIN</option>
    </privilege>
  </privileges>
</siardArchive>
```

D.3 Example of the XML schema definition of a table: `table0.xsd`

SIARD generates an XML schema definition for each table that assigns the correct XML data types to the columns.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.admin.ch/xmlns/siard/1.0/schema0/table0.xsd"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  targetNamespace="http://www.admin.ch/xmlns/siard/1.0/schema0/table0.xsd">
  <xs:element name="table">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="row"
type="rowType"></xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="rowType">
    <xs:sequence>
      <xs:element minOccurs="0" name= "c1" type="xs:boolean" /> <!-- CBOOLEAN -->
      <xs:element minOccurs="0" name= "c2" type="xs:hexBinary" /> <!-- CBIT -->
      <xs:element minOccurs="0" name= "c3" type="xs:hexBinary" /> <!-- CBIT_32 -->
      <xs:element minOccurs="0" name= "c4" type="xs:hexBinary" /> <!--
CBIT_VARYING_160 -->
      <xs:element minOccurs="0" name= "c5" type="xs:hexBinary" /> <!-- CBINARY -->
      <xs:element minOccurs="0" name= "c6" type="xs:hexBinary" /> <!-- CBINARY_5 -
->
      <xs:element minOccurs="0" name= "c7" type="xs:hexBinary" /> <!--
CBINARY_VARYING_32 -->
      <xs:element minOccurs="0" name= "c8" type="blobType" /> <!--
CBINARY_LARGE_OBJECT -->
      <xs:element               name= "c9" type="xs:string" /> <!-- CCHARACTER -->
      <xs:element minOccurs="0" name="c10" type="xs:string" /> <!-- CCHARACTER_5 -
->
      <xs:element minOccurs="0" name="c11" type="xs:string" /> <!--
CCHARACTER_VARYING_32 -->
      <xs:element minOccurs="0" name="c12" type="clobType" /> <!--
CCHARACTER_LARGE_OBJECT -->
      <xs:element minOccurs="0" name="c13" type="xs:string" /> <!--
CNATIONAL_CHARACTER -->
```

```
      <xs:element minOccurs="0" name="c14" type="xs:string" /> <!--
CNATIONAL_CHARACTER_5 -->
      <xs:element minOccurs="0" name="c15" type="xs:string" /> <!--
CNATIONAL_CHARACTER_VARYING_32 -->
      <xs:element minOccurs="0" name="c16" type="clobType" /> <!--
CNATIONAL_CHARACTER_LARGE_OBJECT -->
      <xs:element minOccurs="0" name="c17" type="clobType" /> <!-- CXML -->
      <xs:element minOccurs="0" name="c18" type="xs:integer" /> <!-- CSMALLINT -->
      <xs:element               name="c19" type="xs:integer" /> <!-- CINTEGER -->
      <xs:element minOccurs="0" name="c20" type="xs:integer" /> <!-- CBIGINT -->
      <xs:element minOccurs="0" name="c21" type="xs:decimal" /> <!-- CDECIMAL -->
      <xs:element minOccurs="0" name="c22" type="xs:decimal" /> <!-- CDECIMAL_3 --
>
      <xs:element minOccurs="0" name="c23" type="xs:decimal" /> <!-- CDECIMAL_5_2
-->
      <xs:element minOccurs="0" name="c24" type="xs:decimal" /> <!-- CNUMERIC -->
      <xs:element minOccurs="0" name="c25" type="xs:decimal" /> <!-- CNUMERIC_3 --
>
      <xs:element minOccurs="0" name="c26" type="xs:decimal" /> <!-- CNUMERIC_5_2
-->
      <xs:element minOccurs="0" name="c27" type="xs:float" /> <!-- CREAL -->
      <xs:element minOccurs="0" name="c28" type="xs:float" /> <!-- CFLOAT -->
      <xs:element minOccurs="0" name="c29" type="xs:float" /> <!-- CFLOAT_13 -->
      <xs:element minOccurs="0" name="c30" type="xs:float" /> <!--
CDOUBLE_PRECISION -->
      <xs:element minOccurs="0" name="c31" type="dateType" /> <!-- CDATE -->
      <xs:element minOccurs="0" name="c32" type="xs:time" /> <!-- CTIME -->
      <xs:element minOccurs="0" name="c33" type="xs:time" /> <!-- CTIME_5 -->
      <xs:element minOccurs="0" name="c34" type="dateTimeType" /> <!-- CTIMESTAMP
-->
      <xs:element minOccurs="0" name="c35" type="dateTimeType" /> <!--
CTIMESTAMP_7 -->
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="clobType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="file" type="xs:anyURI" use="required" />
        <xs:attribute name="length" type="xs:integer" use="required" />
        <xs:attribute name="messageDigest" type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="blobType">
    <xs:simpleContent>
      <xs:extension base="xs:hexBinary">
        <xs:attribute name="file" type="xs:anyURI" use="required" />
        <xs:attribute name="length" type="xs:integer" use="required" />
        <xs:attribute name="messageDigest" type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!-- date type between 0001 and 9999 restricted to UTC -->
  <xs:simpleType name="dateType">
    <xs:annotation>
      <xs:documentation>
        dateType restricts xs:date to dates between 0001 and 9999
        and is in UTC (no +/- but an optional Z)
```

```
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:date">
      <xs:minInclusive value="0001-01-01Z"/>
      <xs:maxExclusive value="10000-01-01Z"/>
      <xs:pattern value="\d{4}-\d{2}-\d{2}Z?"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- time type restricted to UTC -->
  <xs:simpleType name="timeType">
    <xs:annotation>
      <xs:documentation>
        timeType restricts xs:date to dates between 0001 and 9999
        and is in UTC (no +/- but an optional Z)
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:time">
      <xs:pattern value="\d{2}:\d{2}:\d{2}Z?"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- dateTime type between 0001 and 9999 restricted to UTC -->
  <xs:simpleType name="dateTimeType">
    <xs:annotation>
      <xs:documentation>
        dateTimeType restricts xs:dateTime to dates between 0001 and 9999
        and is in UTC (no +/- after the T but an optional Z)
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:dateTime">
      <xs:minInclusive value="0001-01-01T00:00:00.000000000Z"/>
      <xs:maxExclusive value="10000-01-01T00:00:00.000000000Z"/>
      <xs:pattern value="\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}(\.\d*)Z?"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

D.4 Example of the table data of a table: `table0.xml`

The table data are stored in an XML file that satisfies the XML schema definition of the table.

```
<?xml version="1.0" encoding="utf-8"?>
<table
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.admin.ch/xmlns/siard/1.0/schema0/table0.xsd"
  xsi:schemaLocation="http://www.admin.ch/xmlns/siard/1.0/schema0/table0.xsd ta-
ble0.xsd">
  <row>

    <c1>true</c1> <!-- CBOOLEAN -->
    <c2>01</c2> <!-- CBIT -->
    <c3>31323334</c3> <!-- CBIT_32 -->
    <c4>1E1F202122232425262728293031323 3</c4> <!-- CBIT_VARYING_160 -->
    <c5>1E</c5> <!-- CBINARY -->
    <c6>1E1F202122</c6> <!-- CBINARY_5 -->
    <c7>1E1F2021222324252627</c7> <!-- CBINARY_VARYING_32 -->
```

```
    <c8 file="record0.bin" length="16000" messageDi-
gest="MD5D1C411FC45542DCA86EEB1CC9B4B596C" /> <!-- CBINARY_LARGE_OBJECT -->
    <c9>A</c9> <!-- CCHARACTER -->
    <c10>ABCDE</c10> <!-- CCHARACTER_5 -->
    <c11>A varchar(32) text</c11> <!-- CCHARACTER_VARYING_32 -->
    <c12 file="record0.txt" length="84000" messageDi-
gest="MD5D1341187455adfCA86E5666C974B5950"/> <!-- CCHARACTER_LARGE_OBJECT -->
    <c13>Ä</c13> <!-- CNATIONAL_CHARACTER -->
    <c14>ABCDÖ</c14> <!-- CNATIONAL_CHARACTER_5 -->
    <c15>A national varchar(32) text</c15> <!-- CNATIONAL_CHARACTER_VARYING -->
    <c16 file="record0.txt" length="84000" messageDi-
gest="MD5243561FE455EBFCAA1256667774B5831" /> <!--
CNATIONAL_CHARACTER_LARGE_OBJECT -->
    <c17 file="record0.xml" length="10424" messageDi-
gest="MD55234Fd874EFADFCA86E5CDDC97398991" /> <!-- CXML -->
    <c18>4321</c18> <!-- SMALLINT -->
    <c19>5</c19> <!-- CINTEGER -->
    <c20>98765432101234</c20> <!-- CBIGINT -->
    <c21>12345</c21> <!-- CDECIMAL -->
    <c22>123</c22> <!-- CDECIMAL_3 -->
    <c23>123.45</c23> <!-- CDECIMAL_5_2 -->
    <c24>54321</c24> <!-- NUMERIC -->
    <c25>123</c25> <!-- NUMERIC_3 -->
    <c26>123.45</c26> <!-- NUMERIC_5_2 -->
    <c27>3.14159</c27> <!-- CREAL -->
    <c28>3.14159</c28> <!-- CFLOAT -->
    <c29>3.141</c29> <!-- CFLOAT_13 -->
    <c30>987654321.0900</c30> <!-- CDOUBLE_PRECISION -->
    <c31>2013-10-18</c31> <!-- CDATE -->
    <c32>17:24:33</c32> <!-- CTIME -->
    <c33>17:24:33.12345</c33> <!-- CTIME_5 -->
    <c34>2011-12-05T16:24:33.123456</c34> <!-- CTIMESTAMP -->
    <c35>2011-12-05T16:24:33.1234567</c35> <!-- CTIMESTAMP_7 -->
  </row>
</table>
```

D.5 Example of the XML schema definition of a table with advanced and structured data types: table1.xsd

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.admin.ch/xmlns/siard/1.0/schema0/table1.xsd"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  targetNamespace="http://www.admin.ch/xmlns/siard/1.0/schema0/table1.xsd">
  <xs:element name="table">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="row"
type="rowType"></xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
  <xs:complexType name="rowType">
    <xs:sequence>
      <xs:element               name= "c1" type="xs:integer" /> <!-- ID -->
      <xs:element minOccurs="0" name= "c2" type="xs:integer" /> <!-- CDISTINCT -->
      <xs:element minOccurs="0" name= "c3"> <!-- CROW -->
        <xs:complexType>
          <xs:sequence>
          <xs:element minOccurs="0" name="r1" type="xs:integer" /> <!-- TABLEID --
>
          <xs:element minOccurs="0" name="r2" type="clobType" /> <!--
TRANSCRIPTION -->
          <xs:element minOccurs="0" name="r3" type="blobType "/> <!-- SOUND -->
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element minOccurs="0" name= "c4"> <!-- CARRAY -->
        <xs:complexType>
          <xs:sequence>
          <xs:element minOccurs="0" name="a1" type="xs:string" />
          <xs:element minOccurs="0" name="a2" type="xs:string" />
          <xs:element minOccurs="0" name="a3" type="xs:string" />
          <xs:element minOccurs="0" name="a4" type="xs:string" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element minOccurs="0" name= "c5" > <!-- CUDT -->
        <xs:complexType>
          <xs:sequence>
        <xs:element minOccurs="0" name="u1" type="xs:integer" /> <!-- ID -->
        <xs:element minOccurs="0" name="u2"> <!-- NESTEDROW -->
          <xs:complexType>
          <xs:sequence>
          <xs:element minOccurs="0" name="r1" type="xs:integer" /> <!-- TABLEID --
>
          <xs:element minOccurs="0" name="r2" type="clobType" /> <!--
TRANSCRIPTION -->
          <xs:element minOccurs="0" name="r3" type="blobType "/> <!-- SOUND -->
          </xs:sequence>
          </xs:complexType>
        </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="clobType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="file" type="xs:anyURI" use="required" />
        <xs:attribute name="length" type="xs:integer" use="required" />
        <xs:attribute name="messageDigest" type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="blobType">
    <xs:simpleContent>
      <xs:extension base="xs:hexBinary">
        <xs:attribute name="file" type="xs:anyURI" use="required" />
```

```
        <xs:attribute name="length" type="xs:integer" use="required" />
        <xs:attribute name="messageDigest" type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!-- date type between 0001 and 9999 restricted to UTC -->
  <xs:simpleType name="dateType">
    <xs:annotation>
      <xs:documentation>
        dateType restricts xs:date to dates between 0001 and 9999
        and is in UTC (no +/- but an optional Z)
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:date">
      <xs:minInclusive value="0001-01-01Z"/>
      <xs:maxExclusive value="10000-01-01Z"/>
      <xs:pattern value="\d{4}-\d{2}-\d{2}Z?"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- time type restricted to UTC -->
  <xs:simpleType name="timeType">
    <xs:annotation>
      <xs:documentation>
        timeType restricts xs:date to dates between 0001 and 9999
        and is in UTC (no +/- but an optional Z)
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:time">
      <xs:pattern value="\d{2}:\d{2}:\d{2}Z?"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- dateTime type between 0001 and 9999 restricted to UTC -->
  <xs:simpleType name="dateTimeType">
    <xs:annotation>
      <xs:documentation>
        dateTimeType restricts xs:dateTime to dates between 0001 and 9999
        and is in UTC (no +/- but an optional Z)
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:dateTime">
      <xs:minInclusive value="0001-01-01T00:00:00.000000000Z"/>
      <xs:maxExclusive value="10000-01-01T00:00:00.000000000Z"/>
      <xs:pattern value="\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}(\.\d*)Z?"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

D.6 Example of the table data of a table with advanced and structured data types: table1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<table
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.admin.ch/xmlns/siard/1.0/schema0/table1.xsd"
  xsi:schemaLocation="http://www.admin.ch/xmlns/siard/1.0/schema0/table1.xsd ta-
ble1.xsd">
```

```
  <row>
    <c1>5</c1> <!-- ID -->
    <c2>5555</c2> <!-- CDISTINCT -->
    <c3> <!-- CROW -->
      <r1>1234</r1> <!-- TAPEID -->
      <r2 file="record0.txt" length="84000" messageDi-
gest="MD5D1341187455adfCA86E5666C974B5950"/> <!-- TRANSCRIPTIOM -->
      <r3 file="record0.bin" length="16000" messageDi-
gest="MD5D1C411FC45542DCA86EEB1CC9B4B596C"/> <!-- SOUND -->
    </c3>
    <c4> <!-- CARRAY -->
      <a1>Vorname1</a1>
       <!--  a2 is NULL! and therefore missing -->
      <a3>Patronymic</a3>
      <a4>Name</a4>
    </c4>
    <c5> <!-- CUDT -->
      <u1>9876</u1> <!-- ID -->
      <u2> <!-- NESTEDROW -->
            <r1>71934576</r1> <!-- TAPEID -->
            <!-- r2 is NULL and therefore missing -->
            <r3 file="sub1000/record0.bin" length="16000" messageDi-
gest="MD5D1C411FC45542DCA86EEB1CC9B4B596C"/> <!-- SOUND -->
      </u2>
    </c5>
  </row>
</table>
```