SIARD Format Specification

Name	SIARD-2.2 Format Specification
Category	Standard
Maturity level	Approved
Version	2.2
Status	Reviewed version
Approval date	2021-08-31
Issue date	2021-08-31
Replaces	SIARD-2.1.1
Prerequisite	None
Attachments	metadata.xsd
Languages	English, German (awaits translation), French (awaits translation), Italian (awaits translation)
Publisher / distributor	DILCIS Board, https://dilcis.eu/ Swiss Federal Archives, https://www.bar.admin.ch/

Summary

This document contains the specification for the SIARD file format version 2.2.
SIARD stands for Software Independent Archival of Relational Databases. The format version 1.0 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.

Version history

Relationship of the present version to previous versions:

eCH-0165 v1.0	Replaced by version 2.1.1: Version 1.0 has been replaced by a more recent and current version. Its use remains possible, but it is recommended to use the present version.
eCH-0165 v2.0	Abrogated: Version 2.0 has been displaced because of errors and ambiguities that might have led to practical problems in the long term. This old issue must no longer be used, and either the present version or version 1.0 must be used.
SIARD-2.1	Replaced by version 2.1.1: Version 2.1 corrects errors and ambiguities in version 2.1. It has been developed by the eCH Fachgruppe on Digital Preservation but is no official standards by the eCH.
SIARD-2.1.1	Replaced by version 2.2: Version 2.1.1 documents the current state of the SIARD file format. It is identical to version 2.1 save for a few precisions in the wording.
SIARD-2.2	Version 2.2 adds support for files outside the database according to part 9 of SQL:2008 (ISO/IEC 9075-9:2008 – SQL/MED) as well as scalability supporting large objects stored outside the SIARD file. Apart from this it is identical to version 2.1.1. It has been developed by the DILCIS Board during the E-ARK3 project.[1]

[1] SIARD 2.2 is expected to be supported by SIARD Suite, Database Preservation Toolkit and other applications

Table of Contents

Es wurden keine Einträge für das Inhaltsverzeichnis gefunden.

1 Introduction

1.1 Status

This document was approved by the DILCIS Board and the Swiss Federal Archives.

1.2 Area of application

1.2.1 Addressees / target group

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

1.2.2 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)[2].

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] 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

1.2.3 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[3].

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.

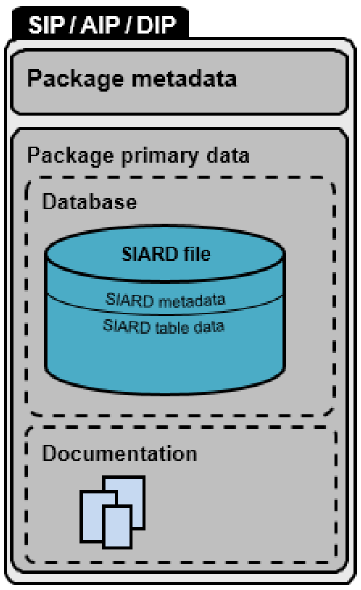


Fig SEQ Fig * ARABIC 1: Diagram of an information package containing a SIARD file

[3] http://public.ccsds.org/pubs/650x0m2.pdf.

2 Structure of the document

2.1 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

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

2.2 ID for requirements

The requirements are unambiguously identifiable by means of an ID.

ID
G_2.2-1

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
	L_	=	Requirements for LOB structure outside the SIARD file
	S_	=	Requirements for segmentation of LOBs and SIARD file

2.1-1	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.

2.3 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.

2.4 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

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 files in XML 1.0 format[4] that conform to schema definitions according to XML Schema 1.0[5]. 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. This file may reference externally stored Large Objects that belong to the database in a larger sense. In rare cases a SIARD file may need to be segmented due to size.	M

3.3 Character sets and characters

ID	Description of requirement	M/O
G_3.3-1	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, NCHAR VARYING, 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

[4] https://www.w3.org/TR/REC-xml/

[5] https://www.w3.org/TR/xmlschema-1/, https://www.w3.org/TR/xmlschema-2/, https://www.w3.org/TR/xmlschema-ref/

G_3.3-4	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, greater than, and ampersand are represented in XML as entity references.	M

Original characters	Characters in the SIARD format
0 to 8	\u0000 to \u0008
14 to 31	\u000E to \u001F
32	\u0020, for multiple spaces
"	&quot;
&	&amp;
'	&apos;
<	&lt;
>	&gt;
\	\u005c
127 to 159	\u007F to \u009F

3.4 File URI Scheme

The file URI scheme used for referencing externally stored large objects is defined in RFC 8089[6].

ID	Description of requirement	M/O
G_3.4-1	All externally referenced files are specified using a file URI as specified in RFC 8089.	M
G_3.4-2	File URIs are stored as URL-encoded ASCII strings in a SIARD archive.	M
G_3.4-3	Externally stored Large Objects can be integrated into ZIP files if the file URI is based on a file system that allows addressing individual entries within a ZIP file. 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

[6] http://en.wikipedia.org/wiki/File_URI_scheme , https://tools.ietf.org/html/rfc8089

3.5 Identifiers and regular identifiers

In SQL:2008 there are regular identifiers[7] 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 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 or a keyword 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 all other identifiers are 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	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

[7] 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.

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[8].	M
G_4.1-2	SIARD files must either be uncompressed or else compressed using the “deflate” algorithm as described in RFC 1951[9]. ***Recommendation*** *It is advisable to use the deflate algorithm.*	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 in the root folder.
	
	Example
	Structure of the SIARD file (schematic)
	`ech-0165_oe.siard`
	        `content/`
	        `header/`
		M

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.	M

[8] ZIP files were originally defined by Phil Katz and are today extensively used as a de facto standard. The current version 6.3.9 of the specification published by PkWare can be found at https://support.pkware.com/display/PKZIP/Application+Note+Archives.

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

	Example
	Structure of the SIARD file (schematic)
	`ech-0165_oe.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).*	

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, TXT, or XML files, or a file extension associated with the MIME type of the lob files in case this is known, e.g. JPG), no other folders or files are permitted.
	
	Example
	Structure of the SIARD file (schematic)
	`ech-0165_oe.siard`
	`    content/`
	`...`
	`        schema1/`
	`...`
	`            table6/`
	`                table6.xml`
	`                table6.xsd`
	`            table7/`
	`                table7.xml`
	`                table7.xsd`
		M

	`lob1`[10]`/`
	`record0.xml`
	`record1.xml`
	`...`
	Recommendation
	It is advisable to normalise the lob folders and lob files and to use, for example, `lob1/` *and* `record0.bin, record0.txt`, or `record0.xml` *instead of the actual name,* or a file extension associated with the MIME type of the lob files in case this is known *(see restrictions under P_4.2-6).*	

P_4.2-4	In order to facilitate the recognition of the SIARD Format (e.g. by PRONOM) an empty folder `/header/siardversion/2.2/` identifying the version of the SIARD Format must exist.	M

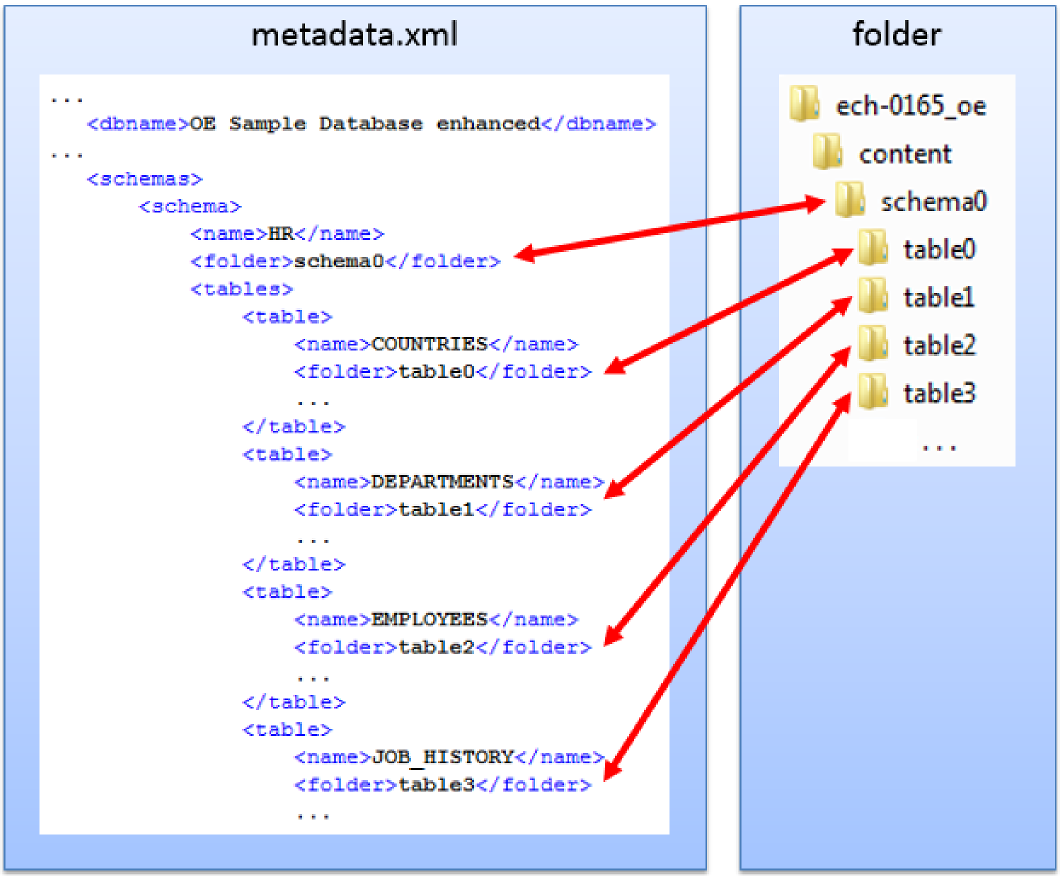
P_4.2-5	The `metadata.xml` and `metadata.xsd` files must be present in the `header/` folder. Additional files, such as style sheets, are permitted.
	
	Example
	Structure of the SIARD file (schematic)
	`ech-0165_oe.siard`
	`    content/`
	`...`
	`    header/`
	`        metadata.xml`
	`        metadata.xsd`
	`        siardversion/`
	`            2.2/`
	`...`
		M

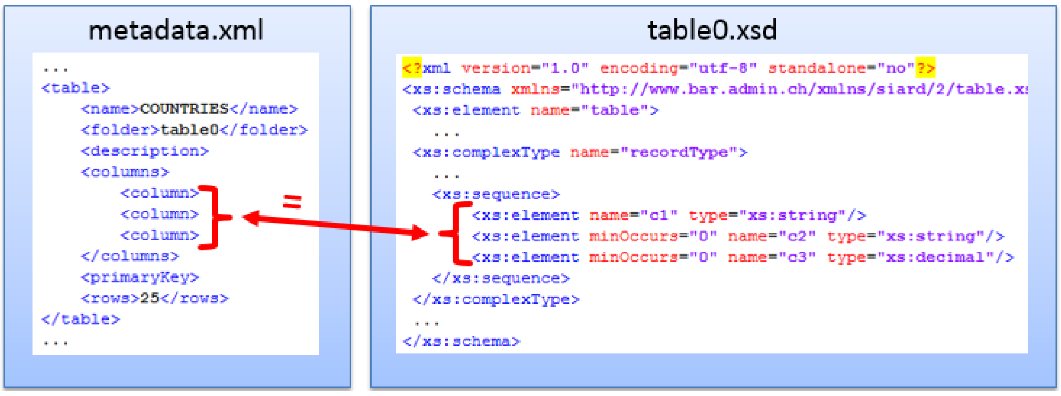
[10] In this example, column 2 contains additional lob files that are accordingly stored in `lob1/`.

P_4.2-6	All file and folder names referring to elements inside the SIARD (ZIP64) file 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

ID	Description of requirement	M/O

P_4.3-1	The structure prescribed in the `metadata.xml` must be identical to that in the `content/` folder.
	
	Example
	
	 metadata.xml
	folder
	ech-0165_oe
	content
	schema0
	table0
	table1
	table2
	table3
	...
	`...`
	`<dbname>OE Sample Database enhanced</dbname>`
	`...`
	`<schemas>`
	`<schema>`
	`<name>HR</name>`
	`<folder>schema0</folder>`
	`<tables>`
	`<table>`
	`<name>COUNTRIES</name>`
	`<folder>table0</folder>`
	`...`
	`</table>`
	`<table>`
	`<name>DEPARTMENTS</name>`
	`<folder>table1</folder>`
	`...`
	`</table>`
	`<table>`
	`<name>EMPLOYEES</name>`
	`<folder>table2</folder>`
	`...`
	`</table>`
	`<table>`
	`<name>JOB_HISTORY</name>`
	`<folder>table3</folder>`
	`...`
		M

P_4.3-2	The number of columns in a table specified in `metadata.xml` must be identical to that in the corresponding `table[number].xsd` file.
	
	Example
	
	
	`...`
	`<table>`
	`<name>COUNTRIES</name>`
	`<folder>table0</folder>`
	`<description>`
	`<columns>`
	`<column>`
	`<column>`
	`<column>`
	`</columns>`
	`<primaryKey>`
	`<rows>25</rows>`
	`</table>`
	`...`
	
	`<?xml version="1.0" encoding="utf-8" standalone="no"?>`
	`<xs:schema xmlns="http://www.bar.admin.ch/xmlns/siard/2/table.x`
	`<xs:element name="table">`
	`...`
	`<xs:complexType name="recordType">`
	`...`
	`<xs:sequence>`
	`<xs:element name="c1" type="xs:string"/>`
	`<xs:element minOccurs="0" name="c2" type="xs:string"/>`
	`<xs:element minOccurs="0" name="c3" type="xs:decimal"/>`
	`</xs:sequence>`
	`</xs:complexType>`
	`...`
	`</xs:schema>`
		M

P_4.3-3	The data type information on the column definitions in `metadata.xml` must be identical to that in the corresponding `table[number].xsd` file.	M

The SQL:2008 built-in data types[11] are converted into XML data types in the `table[number].xsd` schema files in accordance with the following table.

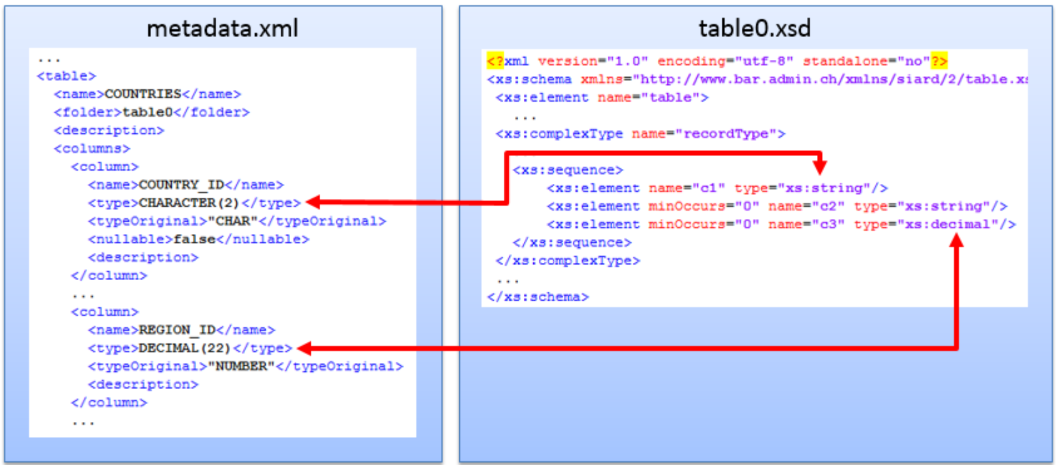
SQL:2008	XML
BIGINT	xs:integer
BINARY LARGE OBJECT(...), BLOB(...)	blobType[12]
BINARY VARYING(...), VARBINARY(...)	xs:hexBinary / clobType[13]
BINARY(…)	xs:hexBinary / blobType[13]
BOOLEAN	xs:boolean
CHARACTER LARGE OBJECT(...), CLOB(...)	clobType[13]
CHARACTER VARYING(...), CHAR VARYING(...), VARCHAR(...)	xs:string / clobType[13]
CHARACTER(...), CHAR(...)	xs:string / clobType[13]
DATE	dateType
DATALINK	blobType / clobType[13]
DECIMAL(...), DEC(...)	xs:decimal
DOUBLE PRECISION	xs:double
FLOAT(p)	xs:double
INTEGER, INT	xs:integer
INTERVAL <start> [TO <end>]	xs:duration
NATIONAL CHARACTER LARGE OBJECT(...), NCHAR LARGE OBJECT(...), NCLOB(...)	clobType[13]
NATIONAL CHARACTER VARYING(...), NATIONAL CHAR VARYING(...), NCHAR VARYING(...)	xs:string / clobType[13]
NATIONAL CHARACTER(...), NCHAR(...), NATIONAL CHAR(...),	xs:string / clobType[13]
NUMERIC(...)	xs:decimal

[11] BIT and BIT VARYING are data types that originated in prior SQL definitions and have been replaced by BOOLEAN and BINARY in SQL:2008. BIT(1) is converted to BOOLEAN, BIT(n) is converted to BINARY((n+7/8).

[12] For the XML data types *blobType* and *clobType* see G_3.1-1, T_6.2-1 and metadata.xsd.
DATALINK is represented in XML as blobType (or clobType) using the optional DLURLPATHONLY

REAL	xs:float
SMALLINT	xs:integer
TIME(...)	timeType
TIME WITH TIME ZONE(...)	timeType
TIMESTAMP(...)	dateTimeType
TIMESTAMP WITH TIME ZONE(...)	dateTimeType
XML	clobType[11]

Example



```
...
<table>
  <name>COUNTRIES</name>
  <folder>table0</folder>
  <description>
  <columns>
    <column>
      <name>COUNTRY_ID</name>
      <type>CHARACTER(2)</type>
      <typeOriginal>"CHAR"</typeOriginal>
      <nullable>false</nullable>
      <description>
    </column>
    ...
    <column>
      <name>REGION_ID</name>
      <type>DECIMAL(22)</type>
      <typeOriginal>"NUMBER"</typeOriginal>
      <description>
    </column>
    ...
```

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<xs:schema xmlns="http://www.bar.admin.ch/xmlns/siard/2/table.xs
 <xs:element name="table">
   ...
 <xs:complexType name="recordType">
   <xs:sequence>
       <xs:element name="c1" type="xs:string"/>
       <xs:element minOccurs="0" name="c2" type="xs:string"/>
       <xs:element minOccurs="0" name="c3" type="xs:decimal"/>
   </xs:sequence>
 </xs:complexType>
 ...
</xs:schema>
```

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).

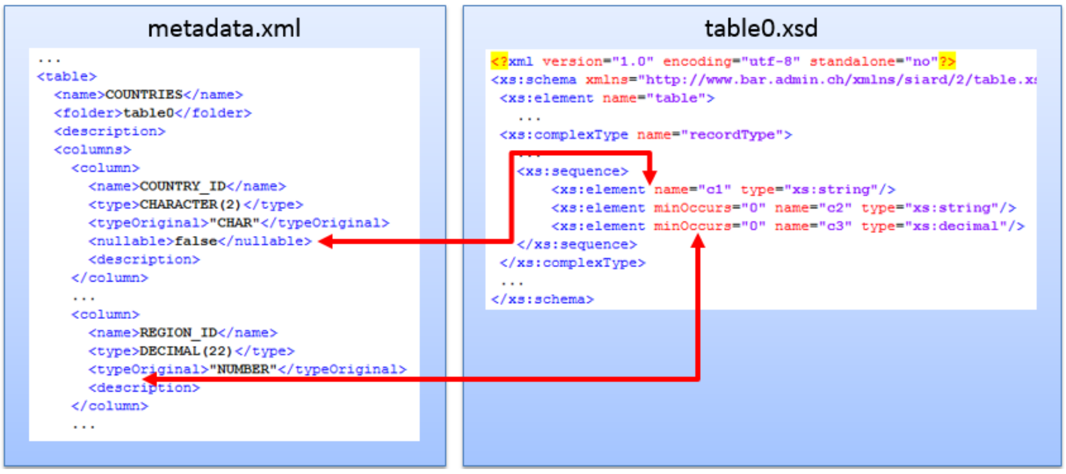
The timeType is a restriction of xs:time to the UTC “time zone”.

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. No additional attributes are required for inlined values, where the CLOB value is stored directly.
If the CLOB value is stored in a separate file the *file* and *length* attributes are mandatory, and the *digestType* and *digest* attributes are optional. 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 attribute contains integrity information according to the equally optional digestType attribute.

The blobType is an extension of xs:hexBinary. No additional attributes are required for inlined values, where the BLOB value is stored directly.
If the BLOB value is stored in a separate file the *file* and *length* attributes are mandatory, and the *digestType* and *digest* attributes are optional. 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 attribute contains integrity information according to the equally optional digestType attribute.
For the SQL:2008 data type DATALINK the blobType is used with attribute *dlurlpathonly* (section 6.4 in ISO/IEC 9075-9:2008). It is a restriction of xs:anyURI following ISO/IEC 9075-9:2008, section 8 URLs. The specification is a direct translation of the format of

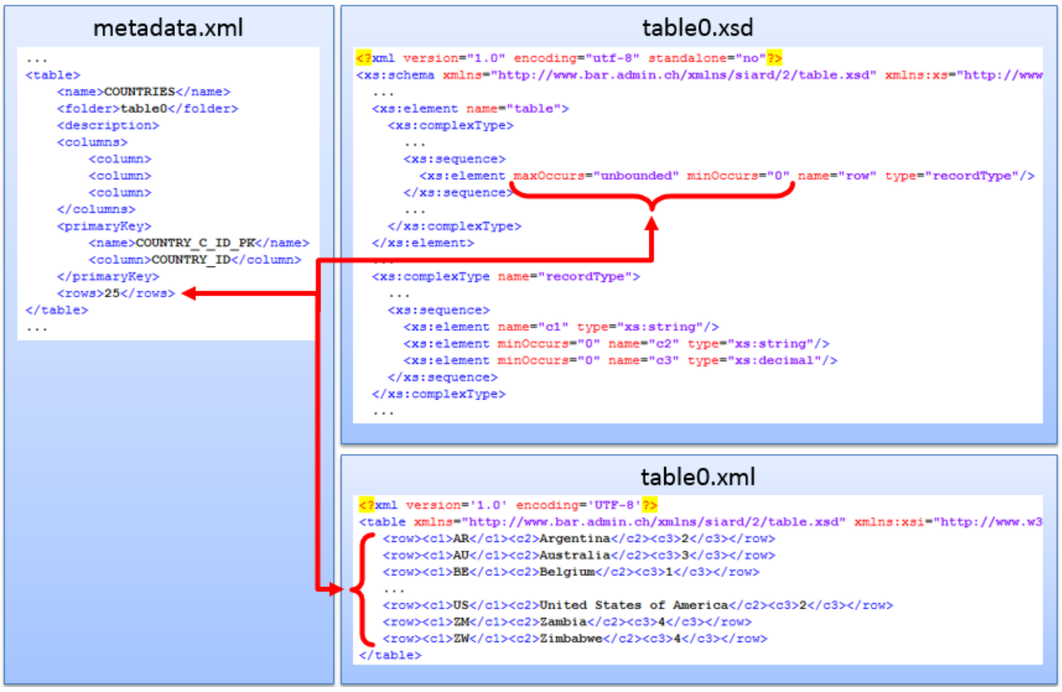
	HTTP and FILE URLs specified in [RFC3986], except that “localhost” has been omitted from the format of FILE URL. blobType is defined in metadata.xsd.	
P_4.3-4	The named DISTINCT data types are converted to the XML data type in the `table[number].xsd` schema files which would be used for representing their base types.	M
P_4.3-5	Arrays are 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. **Example** See example `table0.xsd` in appendix D.3c.	M
P_4.3-6	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. **Example** See example `table0.xsd` in appendix D.3c.	M

P_4.3-7	The nullable information on the column definitions in the `metadata.xml` must be identical to that in the corresponding `table[number].xsd` file.
	
	Example
	
	metadata.xml
	`...`
	`<table>`
	`  <name>COUNTRIES</name>`
	`  <folder>table0</folder>`
	`  <description>`
	`  <columns>`
	`    <column>`
	`      <name>COUNTRY_ID</name>`
	`      <type>CHARACTER(2)</type>`
	`      <typeOriginal>"CHAR"</typeOriginal>`
	`      <nullable>false</nullable>`
	`      <description>`
	`    </column>`
	`    ...`
	`    <column>`
	`      <name>REGION_ID</name>`
	`      <type>DECIMAL(22)</type>`
	`      <typeOriginal>"NUMBER"</typeOriginal>`
	`      <description>`
	`    </column>`
	`    ...`
	
	table0.xsd
	`<?xml version="1.0" encoding="utf-8" standalone="no"?>`
	`<xs:schema xmlns="http://www.bar.admin.ch/xmlns/siard/2/table.x`
	` <xs:element name="table">`
	`   ...`
	` <xs:complexType name="recordType">`
	`   ...`
	`   <xs:sequence>`
	`       <xs:element name="c1" type="xs:string"/>`
	`       <xs:element minOccurs="0" name="c2" type="xs:string"/>`
	`       <xs:element minOccurs="0" name="c3" type="xs:decimal"/>`
	`   </xs:sequence>`
	` </xs:complexType>`
	` ...`
	`</xs:schema>`
	
	Note
	The SQL:2008 notation "<nullable>true</nullable>" becomes “minOccurs="0"" in XML. "<nullable>false</nullable>" corresponds to "minOccurs="1"" in XML. However, as “minOccurs=”1”” is the default value, it is often omitted. No indication of “<nullable>” implies “<nullable>true</nullable>”.
		M

P_4.3-8	The column sequence in the `metadata.xml` must be identical to that in the corresponding `table[number].xsd`.	M
P_4.3-9	The field sequence in the table definition of `metadata.xml` must be identical to the field sequence in the corresponding `table[number].xsd`.	M

P_4.3-10	The number of lines in a table in metadata.xml must be identical to the number of lines in the corresponding table[number].xml. The number of lines in a table in metadata.xml must fit into the area specified in the corresponding table[number].xsd.	M

Example



```
...
<table>
    <name>COUNTRIES</name>
    <folder>table0</folder>
    <description>
    <columns>
        <column>
        <column>
        <column>
    </columns>
    <primaryKey>
        <name>COUNTRY_C_ID_PK</name>
        <column>COUNTRY_ID</column>
    </primaryKey>
    <rows>25</rows>
</table>
...
```

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<xs:schema xmlns="http://www.bar.admin.ch/xmlns/siard/2/table.xsd" xmlns:xs="http://www
  ...
  <xs:element name="table">
    <xs:complexType>
      ...
      <xs:sequence>
        <xs:element maxOccurs="unbounded" minOccurs="0" name="row" type="recordType"/>
      </xs:sequence>
      ...
    </xs:complexType>
  </xs:element>
  ...
  <xs:complexType name="recordType">
    ...
    <xs:sequence>
      <xs:element name="c1" type="xs:string"/>
      <xs:element minOccurs="0" name="c2" type="xs:string"/>
      <xs:element minOccurs="0" name="c3" type="xs:decimal"/>
    </xs:sequence>
  </xs:complexType>
  ...
```

```
<?xml version='1.0' encoding='UTF-8'?>
<table xmlns="http://www.bar.admin.ch/xmlns/siard/2/table.xsd" xmlns:xsi="http://www.w3
   <row><c1>AR</c1><c2>Argentina</c2><c3>2</c3></row>
   <row><c1>AU</c1><c2>Australia</c2><c3>3</c3></row>
   <row><c1>BE</c1><c2>Belgium</c2><c3>1</c3></row>
   ...
   <row><c1>US</c1><c2>United States of America</c2><c3>2</c3></row>
   <row><c1>ZM</c1><c2>Zambia</c2><c3>4</c3></row>
   <row><c1>ZW</c1><c2>Zimbabwe</c2><c3>4</c3></row>
</table>
```

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`.

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. This 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 pass validation 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
dataOrigin-Timespan	Origination period of the data in the database; approximate indication in text form	M

lobFolder	A “file:” URI representing the base URI for relative URIs indicating the possible 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 global 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	Consists of `digestType` (MD5, SHA-1, or SHA-256) and the corresponding `digest`. The `digest` represents a binary buffer as a hexadecimal string, or alternatively, in the case of SHA-1 or SHA-256, a base64 string. Whether hexadecimal or a base64 encoding is used depends on the length of the digest and of the string. The digest is calculated over the `content/` folder. 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[13]. **Example** See example `metadata.xml` in appendix D.2. ***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.* Note that the messageDigest indicates hexadecimal values and it is therefore not of strict importance whether they are set in upper or lower case. However, lower case is mostly used and enforced, see e.g. RFC 2831 https://www.ietf.org/rfc/rfc2831.txt	O
clientMachine	DNS name of the (client) computer on which the archiving was carried out	O

[13] A message digest code that is stored inside the SIARD file is not, on its own, a guarantee of integrity, for it can be counterfeited by an attacker counterfeiting the SIARD file. This can only be circumvented by storing the message digest code externally. However, internally producing a metadata-independent message digest code at downloading time can support external storage.

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
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 schema	O
views	List of the views stored in the schema	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” or “udt”).	M
underSchema	If the type is based on a super type, schema name of super type	O
underType	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 predefined base type if category is “distinct”	O
attributes	List of attributes, if category is “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	All metadata that are designed as mandatory in `metadata.xml` at attribute level must be completed accordingly.	M

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

Identifier	Meaning	M/O
name	Name of the attribute	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
nullable	Nullable element of the attribute ***Recommendation*** *Use of the nullable element is discouraged.*	O
typeSchema	Schema of advanced or structured data type	O
typeName	Name of advanced or structured data type	O
defaultValue	Default value of attribute	O
description	Description of the meaning and function of the routine	O
cardinality	(Maximum) number of elements if the attribute is an array.	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
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 or view. The column name must be unambiguous within a given table.	M
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. **Example** See the example `metadata.xml` in appendix D.2. **Note** This entry is only meaningful, if the column is a LOB column (e.g. type BLOB, CLOB, DATALINK or XML). If it is missing, it is assumed to equal “.”, e.g. to refer to the same folder as the *lobFolder* 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 global *lobFolder* entry at the database level. The relative *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.	M
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 either an array or a structured data type of category “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
cardinality	(Maximum) number of elements if it is an array.	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 array or an advanced or structured data type of category “udt”:

Identifier	Meaning	M/O
name	Field name in the column or field. The field name must be unique within the same column. ***Recommendation*** *For type “udt” containers (column or field) the fieldname should be identical to the corresponding attribute name. For array containers the field name should be the name of the container followed by the array index in square brackets, starting at 1, i.e. for instance “Punkt[1]”, “Punkt[2]”, etc.*	M
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 field is a LOB field (e.g. type BLOB, CLOB, DATALINK 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 field 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 an array or a structured data type of category or “udt”	O
mimeType	MIME type of this field, if it is a BLOB field 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

Primary key is defined as a unique key that identifies a record.

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

Candidate key is defined as a unique key that is a candidate for a primary key. Primary keys as well as candidate keys are of the same type uniqueKeyType in metadata.xsd. Therefore, the requirements for candidate keys are the same as those for primary keys (see M_5.8-1).

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, AFTER or INSTEAD OF	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
rows	Number of records	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
specificName	Specific name that uniquely identifies the routine in the schema[14].	M
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

[14] The introduction of object-oriented elements into SQL:1999 has made “overloading” possible, allowing two different routines (procedures or functions) to have the same name, as long as their parameter lists differ. Therefore the requirement that a routine be unique within a schema cannot stand. The “specific name” has been introduced instead in order to uniquely identify the routine in the schema.

name	Name of the parameter	M
mode	Mode of the parameter (IN, OUT or INOUT)	M
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
description	Description of the meaning and function of the routine	O
cardinality	(Maximum) number of elements if the parameter is an array.	O

5.17 User level metadata

ID	Description of requirement	M/O
M_5.17-1	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

5.18 Role level metadata

ID	Description of requirement	M/O
M_5.18-1	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

5.19 Privilege level metadata

ID	Description of requirement	M/O
M_5.19-1	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
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 positioned there in the schema and table folder as per the data type.

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 pass validation 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
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, ...* The column tags always start with c1 and increase by 1. There must be no gap, because a NULL value is expressed by a missing corresponding column in the XML file. **Example** See the example `table2.xsd` in Appendix D.3a.	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, datalinkType, dateType, timeType, 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 UDT are u1, u2, …. The names always start with a1 or u1, respectively, and increase by 1. There must be no	M

	gap, because a NULL value is expressed by a missing corresponding column in the XML file. **Example** See the example `table0.xsd` in appendix D.3c.	

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, internally as separate file entries inside the SIARD archive, or externally as standalone files in the file system	M

If the Large Object is stored as a separate file (either internally or externally) the *file* and *length* attributes must be stored in a LOB cell in the `table[number].xml` file. These attributes are optional if the Large Object is stored inline.

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.	M[15]
length	Length (for BLOB and DATALINK in bytes, for CLOB and XML in characters)	M[17]
digestType	Contains the type of integrity information (digest): "MD5", "SHA-1" or "SHA-256". ***Recommendation*** *This attribute, as well as the digest attribute, should be completed for all Large Objects stored as a separate file.*	O
digest	Contains the integrity information of the Large Object. ***Recommendation*** *This attribute, as well as the digestType attribute, should be completed for all Large Objects stored as a separate file.*	O
dlurlpathonly	The file path (including the file name) of the file reference as given in the RDBMS from where the LOB has been exported to SIARD. (ISO/IEC 9075-9:2008 6.4 <string value function>). Applies only to external LOBs.	O

6.3 Date and timestamp data cells

[15] O if the Large Object is stored inline in the `table[number].xml` file.

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 is enforced in the definitions of *dateType* and *dateTimeType*.	M
T_6.3-2	Dates, times and timestamps may need to be stored with terminating Z in UTC. This limitation is enforced in the definitions of dateType, timeType, and dateTimeType. ***Recommendation*** *Store all dates, times, and timestamps with terminating Z.*	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_6.4-2	The *table* file consists of *row* elements containing the data of a line subdivided into the various columns (*c1, c2 ...*). **Example** See the example table2.xml in Appendix D.4a.	M
T_6.4-3	If a cell of a column or field is NULL, it must be omitted. If it equals ‘’, the string of length 0, it must be present but empty.	M
T_6.4-4	If a cell of a column contains a complex value (ARRAY, UDT), it is represented by a sequence of sub elements of the cell (a1, a2, … for ARRAYs, u1, u2, … for UDTs) which in turn contain their respective values. These values may again be complex. **Example** See the example `table0.xml` in appendix D.4c.	O
T_6.4-5	If a table contains data of the large object types (BLOB, CLOB, DATALINK 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 BLOB or DATALINK) or characters (for CLOB or XML). 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**	M

	See the example `table7.xml` in Appendix D.4b. ***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.*	

7 Requirements for the folder structure for LOBs stored outside the SIARD file

These requirements are for the folder structure for LOBs stored outside the SIARD file.

LOBs can be stored outside the SIARD file regardless of whether they are internal (e.g. datatype BLOB) or external (e.g. datatype DATALINK).

LOBs are stored as files in folders for each schema and column.

These folders are defined in the value of the lobFolder element at the siardArchive level (e.g. for all schemas) and in the value of the lobFolder element at the column level (e.g. for every single column).

External LOBs are data stored outside the RDB following part 9 of SQL:2008 (SQL/MED). According to the ISO/IEC 9075-9:2008 (SQL/MED) standard only one link between a cell of type DATALINK and an external file can exists. Otherwise a “datalink exception — external file already linked” is raised[16].

[16] see SQL:2008, part 9, 15.2 Effect of inserting tables into base tables, 1, b, ii)).

The standard compliant way of handling many references to the same file is therefore to create a one-to-many table between the cell of type DATALINK and the cells referring to the file. It is not by creating several cells of type DATALINK referring to the same file

7.1 Folder structure for LOBs stored outside the SIARD file

ID	Description of requirement	M/O

L_7.1-0	LOBs are stored as files in folders for each schema, table and column.
	These folders are defined in metadata.xml according to the value of the element <siardArchive/> <lobFolder/> and the values of the elements <column/> <lobFolder/>.
	The LOB file name should be normalised (compliant with *P_4.2-3)* as defined below.
	The folder structure and folder and file naming should be as follows:
	● A main LOB folder named `[databaseName]_lobs`
	● A LOB folder for each column named after the schema no. i, table no. j, column no. k; i. e.: `s[i]_t[j]_c[k]`
	● A folder named `seg_0`
	● A LOB file named after the table no. j, column no. k and row no. l of the LOB i.e. `t[j]_c[k]_r[l]`
	● A LOB file name suffix named `bin` (or a file extension associated with the MIME type of the lob file in case this is known *(see restrictions under P_4.2-6).)*
	
	Note that the base value for tables is 0, whereas it is 1 for columns and rows.
	
	Example
	`Northwind.siard`
	`Northwind_lobs/`
	`    s0_t2_c4/`
	`        seg_0/`
	`            t2_c4_r1.bin`
	`            t2_c4_r2.bin`
	`            t2_c4_r3.bin`
	`            t2_c4_r4.bin`
	`            t2_c4_r5.bin`
	`            t2_c4_r6.bin`
	`            t2_c4_r7.bin`
	`            t2_c4_r8.bin`
	`    s0_t2_c8/                <!-- new column -->`
	`        seg_0/`
	`            t2_c8_r3.bin`
	`    s0_t11_c6/               <!-- new column -->`
	`        seg_0/`
	`            t11_c6_r7.bin`
	
	metadata.xml
	<?xml version="1.0" encoding="UTF-8" standalone="yes"?>...
	<siardArchive>...<lobFolder>./Northwind_lobs/</lobFolder>
	..
	<column>...<lobFolder>s0_t2_c4/</lobFolder>...</column>
	<column>...<lobFolder>s0_t2_c8/</lobFolder>...</column>
	..
	<column>...<lobFolder>s0_t11_c6/</lobFolder>...</column>
	
	table2.xml
	<row><c1>1</c1><c2>Beverages</c2><c3>Soft drinks, coffees, teas, beers, and ales</c3>
	<c4 file="seg_0/t2_c4_r1.bin" length="10151" /></row>
	<row><c1>5</c1><c2>Seafood</c2><c3></c3><c4 file="seg_0/t2_c4_r5.bin" ... /></row>
	<row><c1>8</c1><c2>Candy/c2><c3></c3><c4 file="seg_0/t2_c4_r8.bin" ... /></row>
		M

L_7.1-1	The `[databaseName]_lobs/seg_[]/` folders may be packaged as ZIP files as well, named with the suffix `.zip`	O

8 Scalability issues

Scalability issues may occur with many and large LOBs stored outside the SIARD file and in rare cases even with the SIARD file itself. These scalability issues are normally implementation dependent, but millions of LOBs and TBs of LOBs can present challenges.
This chapter specifies how these scalability issues can be managed in order to improve interoperability.

8.1 Segmenting LOBs stored outside the SIARD file

In order to achieve effective and easy file and folder handling (such as copying, hashing, validating etc) the number of and size of LOBs in one folder may be limited, and the LOBs may by segmented into different folders. The threshold values for the number limit and size limit are implementation dependent.

ID	Description of requirement	M/O

S_8.1-0	The structure and naming of folders for segmenting LOBs follow L_7.1-0.
	The folder structure and naming should be as follows:
	● A new folder named `seg_[s]` when the file number limit per folder is reached
	● A new folder named `seg_[s]` when the file size limit per folder is reached
	
	Note that the base value for tables is 0, whereas it is 1 for columns and rows.
	The threshold values for file number limit and file size limit are implementation dependent.
	
	Example
	`Northwind.siard`
	`Northwind_lobs/`
	`     s0_t2_c4/`
	`          seg_0/`
	`               t2_c4_r1.bin`
	`               t2_c4_r2.bin`
	`               t2_c4_r3.bin`
	`               t2_c4_r4.bin`
	`          seg_1/          <!-- folder file number limit (4) -->`
	`               t2_c4_r5.bin`
	`               t2_c4_r6.bin`
	`               t2_c4_r7.bin`
	`          seg_2/          <!-- folder file size limit (8 GB) -->`
	`               t2_c4_r8.bin`
	`     s0_t2_c8/`
	`          seg_0/`
	`               t2_c8_r3.bin`
	`     s0_t11_c6/`
	`          seg_0/`
	`               t11_c6_r7.bin`
	metadata.xml
	<?xml version="1.0" encoding="UTF-8" standalone="yes"?>...
	<siardArchive>...<lobFolder>./Northwind_lobs/</lobFolder>
	..
	<column>...<lobFolder>s0_t2_c4/</lobFolder>...</column>
	<column>...<lobFolder>s0_t2_c8/</lobFolder>...</column>
	..
	<column>...<lobFolder>s0_t11_c6/</lobFolder>...</column>
		M[17]

[17] Mandatory only if LOBs are segmented

	table2.xml
	`<row><c1>1</c1><c2>Beverages</c2><c3>Soft drinks, coffees, teas, beers, and ales</c3>`
	`<c4 file="seg_0/t2_c4_r1.bin" length="10151"  /></row>`
	`<row><c1>5</c1><c2>Seafood</c2><c3></c3><c4 file="seg_1/t2_c4_r5.bin" ... /></row>`
	`<row><c1>8</c1><c2>Candy</c2><c3></c3><c4 file="seg_2/t2_c4_r8.bin" ... /></row>`	

8.1.1 Splitting LOBs as binary parts

It may happen that a LOB is larger than the limit size for a folder (for example, it could be a long movie). In that case it will be necessary to carry out binary division on the file itself to divide it into smaller parts using a naming convention.

ID	Description of requirement	M/O

S_8.1.1-0	If a LOB is larger than the folder size limit the LOB must be binary divided (split) into file parts.
	Each file part must have the suffix `_part[nnn]`, with nnn beginning with 001.
	File parts must be stored in order.
	
	Example
	`Northwind.siard`
	`Northwind_lobs/`
	`     s0_t2_c4/`
	`          seg_0/`
	`               t2_c4_r1.bin`
	`               t2_c4_r2.bin`
	`               t2_c4_r3.bin`
	`               t2_c4_r4.bin`
	`          seg_1/              <!-- folder file number limit (4) -->`
	`               t2_c4_r5.bin`
	`               t2_c4_r6.bin_part001    <!-- file size limit (8 GB) -->`
	`          seg_2/`
	`               t2_c4_r6.bin_part002`
	`               t2_c4_r7.bin`
	`               t2_c4_r8.bin`
	`     s0_t2_c8/`
	`          seg_0/`
	`               t2_c8_r3.bin`
	`     s0_t11_c6/`
	`          seg_0/`
	`               t11_c6_r7.bin`
		M[18]

[18] Mandatory only if LOBs are split

8.1.2 Mapping file for segment folders

In case of scalability issues with many LOBs the segmentation into folders may not be sufficient as these LOBs will often be in the same column (e.g. driver license images for 5 mio people will be in one table in one column with 5 mio rows having 5 mio LOBs of drivers license images).
Therefore having the same folder (path) for one column may not be sufficient, for example during creation of the SIARD file or during transfer and packaging into a SIP.
In such a case an optional mapping file may be used. Note that this mapping will only be of relevance to point of creation. During transport or after arrival at an archive the mapping may have to revisited.

ID	Description of requirement	M/O

S_8.1.2-0	If an optional mapping file is used it must be named mapping.txt.
	mapping.txt must be located at the same folder level as the SIARD file.
	mapping.txt must contain the path for the LOB segmented folder followed by a space and then the URI for the location of the segmented folder (URI according to RFC 3986).
	
	Example
	`Northwind_lobs/s0_t2_c4/seg_0/ file://storagesrv1.sfa.ch/Home/stor/`
	`Northwind_lobs/s0_t2_c4/seg_1/ file://storagesrv1.sfa.ch/Home/stor/`
	`Northwind_lobs/s0_t2_c4/seg_2/ file://storagesrv2.sfa.ch/Home/stor/`
	
	i.e. the folder `s0_t2_c4/seg_0/` is mapped to this location:
	`file://storagesrv1.sfa.ch/Home/stor/Northwind_lobs/s0_t2_c4/seg_0/`
		O

8.1.3 Manifest file for LOBs stored outside the SIARD file

The purpose of the optional manifest file for LOBs stored outside the SIARD file is to ease the handling of these files, i.e. during creation of the SIARD file, transfer, packaging into a SIP.

The purpose of specifying the format for the manifest file is to improve interoperability.

The information in the manifest file is already available in other parts of SIARD, and must be based on that information. The manifest file is therefore only a convenience.

The information about the location of a LOB stored outside the SIARD file are available from the combined path of the value of the database lobFolder element and the value of the column lobFolder element in metadata.xml and the value of the file attribute of the column element c[c] in the table[t].xml file.
The information about digest value is available from the value of the digest attribute (See T_6.4-5)

ID	Description of requirement	M/O
S_8.1.3-0	If a manifest file for LOBs stored outside the SIARD file is used it must follow the structure based on GNU md5sum-invocation regardless of checksum algorithm. https://www.gnu.org/software/coreutils/manual/coreutils.html#md5sum-invocation https://www.gnu.org/software/coreutils/manual/coreutils.html#sha2-utilities Checksum, a space, a flag indicating binary or text input mode, and the file name. Binary mode is indicated with ‘*’, text mode with ‘ ’ (space) Example: `2de1ac4c4e8ebb853e17db01af3fb7c3 */Northwind_lobs/s0_t2_c4/seg_2/t2_c4_r8.bin`	O

8.2 Splitting the SIARD file as binary parts

In rare cases it may happen that a SIARD file itself is larger than a given implementation dependent limit, e.g., due to a low folder limit combined with a large total size of tables.
In that case it will be necessary to carry out binary division on the file itself to divide it into smaller parts using a naming convention.
The purpose of specifying the format for splitting is to improve interoperability.

ID	Description of requirement	M/O

S_8.2-0	If a SIARD file is larger than a given limit the file must be binary divided (split) into file parts.
	Each file part must have the suffix `_part[nnn]`, with nnn beginning with 001.
	File parts must be stored in order.
	
	Example
	`Northwind.siard_part001  <!-- file size limit -->`
	`Northwind.siard_part002`
	`Northwind.siard_part003`
	
	The threshold values for file number limit and file size limit are implementation dependent.
		M[19]

[19] Mandatory only if the SIARD file is splitted

9 Version and validity of the specification

The current version of the Specification is 2.2.

10 Change management process

The change management process for this standard is managed by DILCIS.

11 Exclusion of liability / notice regarding third-party rights

DILCIS and SFA accept no liability for decisions or measures taken by the user on the basis of this document. DILCIS and SFA can offer no warranty or guarantee that the information and documents made available are up to date, complete, correct or free from errors. 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 this standard.

12 Copyright

The authors of the present standard retain intellectual property rights in respect thereof. However, they undertake to make their relevant intellectual property or rights in respect of the intellectual property of others available, to the extent possible, free of charge for unrestricted use and further development.

Appendix A – Participation and Review

Authors

Krystyna Ohnesorge, Swiss Federal Archives, krystyna.ohnesorge@bar.admin.ch

Hartwig Thomas, Enter AG, hartwig.thomas@enterag.ch

Andreas Voss †, Swiss Federal Archives

Marcel Büchler, Swiss Federal Archives, marcel.buechler@bar.admin.ch

Audun Lund, Swiss Federal Archives, audun.lund@bar.admin.ch

Claire Röthlisberger-Jourdan, KOST, claire.roethlisberger@kost.admin.ch

Anders Bo Nielsen, Danish National Archives, abn@sa.dk

Arne-Kristian Groven, National Archives of Norway, arngro@arkivverket.no

Luis Faria, KEEP SOLUTIONS, LDA, lfaria@keep.pt

Contributors

Hedi Bruggisser, Thurgau State Archives, hedi.bruggisser@tg.ch

Georg Büchler, KOST, georg.buechler@kost.admin.ch

Boris Domajnko, Slovenian National Archives, boris.domajnko@gov.si

Alain Dubois, Valais State Archives, alain.dubois@admin.vs.ch

Bruno Ferreira, KEEP SOLUTIONS, LDA

Miguel Guimarães, KEEP SOLUTIONS, LDA

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

Zoltán Lux, National Archives of Hungary, lux.zoltan@mnl.gov.hu

Rebekka Plüss, Zurich State Archives, rebekka.pluess@ji.zh.ch

Lauri Rätsep, National Archives of Estonia, lauri.ratsep@ra.ee

Hélder Silva, KEEP SOLUTIONS, LDA, hsilva@keep.pt

Mario Spuler, Fachlabor Gubler, m.spuler@fachlabor-gubler.ch

Martin Dew-Hattens, Danish National Archives, mdh@sa.dk

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.
DATALINK	A data type according to SQL:2008 part 9 SQL/MED (ISO/IEC 9075-9:2008). It contains a reference to a LOB in a system outside of the RDB, but partly controlled by the RDBMS. In SIARD it is treated as a LOB with information about the original path. (DLURLPATHONLY).
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.

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/
RFC 1738	URL specification – in particular the “file:” URL/URI https://www.ietf.org/rfc/rfc1738.txt
RFC 8089	URL specification – The "file" URI Scheme https://tools.ietf.org/html/rfc8089
RFC 1951	Specification of the “deflate” algorithm. https://www.ietf.org/rfc/rfc1951.txt
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 6.3.9 July 15, 20 PKWARE Inc. http://www.pkware.com/documents/casestudies/APPNOTE.TXT

Appendix D – Excerpts of example file ech-0165_oe.siard

The listings in Appendix D are taken from the SIARD file `ech-0165_oe.siard` that is attached to the SIARD Format Specification v2.2 (and 2.1.1).

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"?>
<!-- ====================================================================
XML schema for meta data of the SIARD Format 2.2 RFC
Application: Software-Independent Archival of Relational Databases
Platform   : XML 1.0, XML Schema 2001
Description: This XML schema definition defines the structure of the meta data in the SIARD format 2.2.
======================================================================
Copyright  : 2007, 2014, 2018, Swiss Federal Archives, Berne, Switzerland, 2020 DILCIS and SFA
==================================================================== -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.bar.admin.ch/xmlns/siard/2/metadata.xsd"
targetNamespace="http://www.bar.admin.ch/xmlns/siard/2/metadata.xsd" elementFormDefault="qualified" attributeFormDefault="unqualified" version="2.2"
id="metadata">
  <!-- 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 -->
        <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 codes over all primary data in folder "content" -->
        <xs:element name="messageDigest" type="messageDigestType" minOccurs="0" maxOccurs="unbounded"/>
        <!-- DNS name of client machine from which connection to the database was established 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 (JDBC URL) 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>
      <xs:attribute name="version" type="versionType" use="required"/>
      <!-- constraint: version number with release -->
    </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" maxOccurs="unbounded"/>
    </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 -->
      <xs:element name="types" type="typesType" minOccurs="0"/>
      <!-- list of tables in the schema -->
      <xs:element name="tables" type="tablesType" minOccurs="0"/>
      <!-- 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 -->
  <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" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- complex type type -->
  <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"/>
      <!--  predefined base SQL:2008 type of (DISTINCT) type -->
      <xs:element name="base" type="predefinedTypeType" minOccurs="0"/>
      <!--  alternatively list of attributes (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 -->
```

```
<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" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- complex type attribute -->
<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>
      <!-- either predefined or structured -->
      <xs:sequence>
        <!-- SQL:2008 data type of the column -->
        <xs:element name="type" type="predefinedTypeType"/>
      </xs:sequence>
      <xs:sequence>
        <!-- SQL:2008 schema of advanced or structured data type of the attribute -->
        <xs:element name="typeSchema" type="xs:string" minOccurs="0"/>
        <!-- SQL:2008 name of advanced or structured data type of the attribute -->
        <xs:element name="typeName" type="xs:string"/>
      </xs:sequence>
    </xs:choice>
    <!-- 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"/>
    <!-- default value -->
    <xs:element name="defaultValue" type="xs:string" minOccurs="0"/>
    <!--  SQL_1999 cardinality for ARRAY type -->
    <xs:element name="cardinality" type="xs:integer" 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" maxOccurs="unbounded"/>
  </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="uniqueKeyType" 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" minOccurs="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" 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" maxOccurs="unbounded"/>
    </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"/>
      <!--  folder for LOBs relative to lobFolder of nearest containing
        element for internally or externally stored LOBs -->
      <xs:element name="lobFolder" type="xs:anyURI" minOccurs="0"/>
      <xs:choice>
        <!-- either predefined or structured -->
        <xs:sequence>
          <!-- SQL:2008 predefined data type of the column -->
          <xs:element name="type" type="predefinedTypeType"/>
          <!--  mimeType makes sense only for LOBs and is only informatory -->
          <xs:element name="mimeType" type="xs:string" minOccurs="0"/>
        </xs:sequence>
        <xs:sequence>
          <!-- SQL:2008 schema of UDT name of the column -->
          <xs:element name="typeSchema" type="xs:string" minOccurs="0"/>
          <!-- SQL:2008 name of UDT of the column -->
          <xs:element name="typeName" type="xs:string"/>
        </xs:sequence>
      </xs:choice>
      <!-- original data type of the column -->
      <xs:element name="typeOriginal" type="xs:string" minOccurs="0"/>
      <!-- SQL:2008 attribute list of the column -->
      <xs:element name="fields" type="fieldsType" minOccurs="0"/>
      <!-- nullability (default: true) -->
      <xs:element name="nullable" type="xs:boolean" minOccurs="0"/>
      <!-- default value -->
      <xs:element name="defaultValue" type="xs:string" minOccurs="0"/>
      <!--  SQL_1999 cardinality for ARRAY type -->
      <xs:element name="cardinality" type="xs:integer" minOccurs="0"/>
      <!-- unique, references, check column constraints
       are stored as table constraints -->
      <!-- 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" maxOccurs="unbounded"/>
    </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>
      <!-- attribute name or array element position (1-based) -->
      <xs:element name="name" type="xs:string"/>
      <!--  folder for LOBs relative to lobFolder of nearest containing
        element for internally or externally stored LOBs -->
      <xs:element name="lobFolder" type="xs:anyURI" minOccurs="0"/>
      <!-- SQL:2008 sub field list of the field -->
      <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 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" maxOccurs="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" maxOccurs="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="referentialActionType" minOccurs="0"/>
      <!-- ON UPDATE action e.g. ON UPDATE SET DEFAULT -->
      <xs:element name="updateAction" type="referentialActionType" 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="uniqueKeyType" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- complex type uniqueKey -->
<xs:complexType name="uniqueKeyType">
  <xs:annotation>
    <xs:documentation>unique (primary or candidate) key element in siardArchive</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!-- database name of the unique key -->
    <xs:element name="name" type="xs:string"/>
    <!-- description of the unique key's meaning and content -->
    <xs:element name="description" type="xs:string" minOccurs="0"/>
    <!-- columns belonging to the unique key -->
    <xs:element name="column" type="xs:string" maxOccurs="unbounded"/>
  </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" 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" maxOccurs="unbounded"/>
  </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 BEFORE, AFTER or INSTEAD OF -->
    <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" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- complex type routine -->
<xs:complexType name="routineType">
  <xs:annotation>
    <xs:documentation>Routine</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <!--  specific (unique) name of routine in schema -->
    <xs:element name="specificName" type="xs:string"/>
    <!-- database (possible overloaded) 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="parameters" 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" maxOccurs="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>
      <!-- either predefined or structured -->
      <xs:sequence>
        <!-- SQL:2008 data type of the column -->
        <xs:element name="type" type="predefinedTypeType"/>
      </xs:sequence>
      <xs:sequence>
```

```
          <!-- SQL:2008 schema of UDT name of the column -->
          <xs:element name="typeSchema" type="xs:string" minOccurs="0"/>
          <!-- SQL:2008 name of UDT of the column -->
          <xs:element name="typeName" type="xs:string"/>
        </xs:sequence>
      </xs:choice>
      <!-- original data type of the column -->
      <xs:element name="typeOriginal" type="xs:string" minOccurs="0"/>
      <!--  SQL_1999 cardinality for ARRAY type -->
      <xs:element name="cardinality" type="xs:integer" 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 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="0" 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" 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" 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>
  <!--  complex type for messageDigest with separate algorithm field -->
  <xs:complexType name="messageDigestType">
    <xs:annotation>
      <xs:documentation>Message digests with algorithm ("MD5", "SHA-1" or "SHA-256") and hexadecimal or - for the SHA variants - Base64
code.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="digestType" type="digestTypeType"/>
      <xs:element name="digest" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <!-- simple type fpr predefined SQL:2008 types -->
  <xs:simpleType name="predefinedTypeType">
    <xs:annotation>
      <xs:documentation>predefinedTypeType is constrained to valid SQL:2008 data type values</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:pattern value="INTEGER|INT|SMALLINT|BIGINT"/>
      <xs:pattern value="(NUMERIC|DECIMAL|DEC)(\s*\(\s*[1-9]\d*\s*(,\s*\d+\s*)?\))?"/>
      <xs:pattern value="REAL|DOUBLE PRECISION"/>
      <xs:pattern value="FLOAT(\s*\(\s*[1-9]\d*\s*\))?"/>
      <xs:pattern value="(CHARACTER|CHAR)(\s*\(\s*[1-9]\d*\s*\))?"/>
      <xs:pattern value="(CHARACTER\s+VARYING|CHAR\s+VARYING|VARCHAR)(\s*\(\s*[1-9]\d*\s*\))?"/>
      <xs:pattern value="(CHARACTER\s+LARGE\s+OBJECT|CLOB)(\s*\(\s*[1-9]\d*(\s*(K|M|G))?\s*\))?"/>
      <xs:pattern value="(NATIONAL\s+CHARACTER|NATIONAL\s+CHAR|NCHAR)(\s*\(\s*[1-9]\d*\s*\))?"/>
      <xs:pattern value="(NATIONAL\s+CHARACTER\s+VARYING|NATIONAL\s+CHAR\s+VARYING|NCHAR VARYING)(\s*\(\s*[1-9]\d*\s*\))?"/>
      <xs:pattern value="(NATIONAL\s+CHARACTER\s+LARGE\s+OBJECT|NCHAR\s+LARGE\s+OBJECT|NCLOB)(\s*\(\s*[1-9]\d*(\s*(K|M|G))?\s*\))?"/>
      <xs:pattern value="XML"/>
      <xs:pattern value="BINARY(\s*\(\s*[1-9]\d*\s*\))?"/>
      <xs:pattern value="(BINARY\s+VARYING|VARBINARY)(\s*\(\s*[1-9]\d*\s*\))?"/>
      <xs:pattern value="(BINARY\s+LARGE\s+OBJECT|BLOB)(\s*\(\s*[1-9]\d*(\s*(K|M|G))?\s*\))?"/>
      <xs:pattern value="DATE"/>
      <xs:pattern value="(TIME|TIME\s+WITH\s+TIME\s+ZONE)(\s*\(\s*[1-9]\d*\s*\))?"/>
      <xs:pattern value="(TIMESTAMP|TIMESTAMP\s+WITH\s+TIME\s+ZONE)(\s*\(\s*(0|([1-9]\d*))\s*\))?"/>
      <xs:pattern value="INTERVAL\s+(((YEAR|MONTH|DAY|HOUR|MINUTE)(\s*\(\s*[1-
9]\d*\s*\))?(\s+TO\s+(MONTH|DAY|HOUR|MINUTE|SECOND)(\s*\(\s*[1-9]\d*\s*\))?)?)|(SECOND(\s*\(\s*[1-9]\d*\s*(,\s*\d+\s*)?\))?))"/>
      <xs:pattern value="BOOLEAN"/>
      <xs:pattern value="DATALINK"/>
      <!-- exact numerics (BIGINT from SQL:2008) -->
      <!-- approximate numerics -->
      <!-- character strings -->
      <!-- BINARY strings from SQL:2008 -->
      <!-- datetimes -->
      <!-- DATALINK from SQL:2008 part 9 (SQL/MED) -->
    </xs:restriction>
  </xs:simpleType>
  <!--  type for message digest type -->
  <xs:simpleType name="digestTypeType">
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="collapse"/>
      <xs:enumeration value="MD5"/>
      <xs:enumeration value="SHA-1"/>
      <xs:enumeration value="SHA-256"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- simple type for version number -->
  <xs:simpleType name="versionType">
    <xs:annotation>
```

```
      <xs:documentation>versionType is constrained to "2.2" for conformity with this XML schema</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="collapse"/>
      <xs:enumeration value="2.2"/>
      <!--  to be extended later with
   <xs.enumeration value="2.2"/>
   etc. -->
    </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:minLength value="1"/>
      <xs:pattern value="([a-z]|[A-Z])([a-z]|[A-Z]|[0-9]).*"/>
    </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="INSTEAD OF"/>
      <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 referential action of a foreign key -->
  <xs:simpleType name="referentialActionType">
    <xs:annotation>
      <xs:documentation>referential action is CASCADE, SET NULL, SET DEFAULT, RESTRICT, or NO ACTION</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="CASCADE"/>
      <xs:enumeration value="SET NULL"/>
      <xs:enumeration value="SET DEFAULT"/>
      <xs:enumeration value="RESTRICT"/>
      <xs:enumeration value="NO ACTION"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- simple type for the category of a column or a parameter -->
  <xs:simpleType name="categoryType">
    <xs:annotation>
      <xs:documentation>category of advanced or structured data types is "distinct" or "udt" for conformity with this XML schema</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="collapse"/>
      <xs:enumeration value="distinct"/>
      <xs:enumeration value="udt"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- complex type for the character large object - to be used in table[n].xsd - T_6.2-1 -->
  <xs:complexType name="clobType">
    <xs:annotation>
      <xs:documentation source="T_6.2-1" xml:lang="en">a character large object can either be stored inline or as a file internally or externally.
      The length is in characters, not in bytes.</xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="file" type="xs:anyURI"/>
        <xs:attribute name="length" type="xs:integer"/>
        <xs:attribute name="digestType" type="digestTypeType"/>
        <xs:attribute name="digest" type="xs:string"/>
        <xs:attribute name="dlurlpathonly" type="xs:anyURI"/> <!-- applies only to external LOBs -->
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!-- complex type for the binary large object - to be used in table[n].xsd -->
  <xs:complexType name="blobType">
    <xs:annotation>
      <xs:documentation source="T_6.2-1" xml:lang="en">a binary large object can either be stored inline or as a file internally or externally.
      The length is in bytes.</xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="xs:hexBinary">
        <xs:attribute name="file" type="xs:anyURI"/>
        <xs:attribute name="length" type="xs:integer"/>
        <xs:attribute name="digestType" type="digestTypeType"/>
        <xs:attribute name="digest" type="xs:string"/>
        <xs:attribute name="dlurlpathonly" type="xs:anyURI"/> <!-- applies only to external LOBs -->
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</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="yes"?>
<siardArchive xmlns="http://www.bar.admin.ch/xmlns/siard/2/metadata.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.2"
xsi:schemaLocation="http://www.bar.admin.ch/xmlns/siard/2/metadata.xsd metadata.xsd">
  <dbname>OE Sample Database enhanced</dbname>
  <description>Record with PRODUCT_ID 4000 in table PRODUCT_INFORMATION has a picture.version with updated .xsd</description>
  <archiver>Claire Roethlisberger</archiver>
  <archiverContact>claire.roethlisberger@kost.admin.ch</archiverContact>
  <dataOwner>Oracle (OE database) and Swiss Federal Archives (enhancement)</dataOwner>
  <dataOriginTimespan>2000-2007</dataOriginTimespan>
  <producerApplication>SiardGui 2.1.89 Swiss Federal Archives, Berne, Switzerland, 2007-2018</producerApplication>
  <archivalDate>2018-01-30Z</archivalDate>
  <clientMachine>VMW10.enterag.ch</clientMachine>
  <databaseProduct>Oracle Oracle Database 12c Release 12.1.0.1.0 - 64bit Production</databaseProduct>
  <connection>jdbc:oracle:thin:@localhost:1521:ORCL</connection>
  <databaseUser>OE</databaseUser>
  <schemas>
    <schema>
      <name>HR</name>
      <folder>schema0</folder>
      <tables>
        ...
        <table>
          <name>EMPLOYEES</name>
          <folder>table2</folder>
          <description>employees table. Contains 107 rows. References with departments, jobs, job_history tables. Contains a self reference.
          </description>
          <columns>
            <column>
              <name>EMPLOYEE_ID</name>
              <type>DECIMAL(6)</type>
              <typeOriginal>"NUMBER"</typeOriginal>
              <nullable>false</nullable>
              <description>Primary key of employees table.</description>
            </column>
            <column>
              <name>FIRST_NAME</name>
              <type>VARCHAR(20)</type>
              <typeOriginal>"VARCHAR2"</typeOriginal>
              <description>First name of the employee. A not null column.</description>
            </column>
            <column>
              <name>LAST_NAME</name>
              <type>VARCHAR(25)</type>
              <typeOriginal>"VARCHAR2"</typeOriginal>
              <nullable>false</nullable>
              <description>Last name of the employee. A not null column.</description>
            </column>
            <column>
              <name>EMAIL</name>
              <type>VARCHAR(25)</type>
              <typeOriginal>"VARCHAR2"</typeOriginal>
              <nullable>false</nullable>
              <description>Email id of the employee</description>
            </column>
            <column>
              <name>PHONE_NUMBER</name>
              <type>VARCHAR(20)</type>
              <typeOriginal>"VARCHAR2"</typeOriginal>
              <description>Phone number of the employee; includes country code and area code</description>
            </column>
            <column>
              <name>HIRE_DATE</name>
              <type>DATE</type>
              <typeOriginal>"DATE"</typeOriginal>
              <nullable>false</nullable>
              <description>Date when the employee started on this job. A not null column.</description>
            </column>
            <column>
              <name>JOB_ID</name>
              <type>VARCHAR(10)</type>
              <typeOriginal>"VARCHAR2"</typeOriginal>
              <nullable>false</nullable>
              <description>Current job of the employee; foreign key to job_id column of the jobs table. A not null column.</description>
            </column>
            <column>
              <name>SALARY</name>
              <type>DECIMAL(8, 2)</type>
              <typeOriginal>"NUMBER"</typeOriginal>
              <description>Monthly salary of the employee. Must be greater than zero (enforced by constraint emp_salary_min)</description>
            </column>
            <column>
              <name>COMMISSION_PCT</name>
              <type>DECIMAL(2, 2)</type>
              <typeOriginal>"NUMBER"</typeOriginal>
              <description>Commission percentage of the employee; Only employees in sales department elgible for commission percentage
              </description>
            </column>
            <column>
              <name>MANAGER_ID</name>
              <type>DECIMAL(6)</type>
              <typeOriginal>"NUMBER"</typeOriginal>
              <description>Manager id of the employee; has same domain as manager_id in departments table. Foreign key to employee_id
                column of employees table. (useful for reflexive joins and CONNECT BY query)</description>
            </column>
            <column>
              <name>DEPARTMENT_ID</name>
              <type>DECIMAL(4)</type>
              <typeOriginal>"NUMBER"</typeOriginal>
              <description>Department id where employee works; foreign key to department_id column of the departments table</description>
            </column>
          </columns>
          <primaryKey>
            <name>EMP_EMP_ID_PK</name>
            <column>EMPLOYEE_ID</column>
          </primaryKey>
          <foreignKeys>
            <foreignKey>
              <name>EMP_DEPT_FK</name>
              <referencedSchema>HR</referencedSchema>
              <referencedTable>DEPARTMENTS</referencedTable>
              <reference>
                <column>DEPARTMENT_ID</column>
                <referenced>DEPARTMENT_ID</referenced>
              </reference>
              <deleteAction>RESTRICT</deleteAction>
              <updateAction>CASCADE</updateAction>
            </foreignKey>
            <foreignKey>
              <name>EMP_MANAGER_FK</name>
              <referencedSchema>HR</referencedSchema>
              <referencedTable>EMPLOYEES</referencedTable>
              <reference>
                <column>MANAGER_ID</column>
                <referenced>EMPLOYEE_ID</referenced>
              </reference>
              <deleteAction>RESTRICT</deleteAction>
              <updateAction>CASCADE</updateAction>
            </foreignKey>
            <foreignKey>
              <name>EMP_JOB_FK</name>
              <referencedSchema>HR</referencedSchema>
              <referencedTable>JOBS</referencedTable>
              <reference>
                <column>JOB_ID</column>
                <referenced>JOB_ID</referenced>
              </reference>
              <deleteAction>RESTRICT</deleteAction>
              <updateAction>CASCADE</updateAction>
```

```
        </foreignKey>
      </foreignKeys>
      <candidateKeys>
        <candidateKey>
          <name>EMP_EMAIL_UK</name>
          <column>EMAIL</column>
        </candidateKey>
      </candidateKeys>
      <rows>107</rows>
    </table>
    ...
  </tables>
</schema>
<schema>
  <name>OE</name>
  <folder>schema1</folder>
  <types>
    <type>
      <name>CUST_ADDRESS_TYP</name>
      <category>udt</category>
      <instantiable>true</instantiable>
      <final>true</final>
      <attributes>
        <attribute>
          <name>STREET_ADDRESS</name>
          <type>VARCHAR(40)</type>
        </attribute>
        <attribute>
          <name>POSTAL_CODE</name>
          <type>VARCHAR(10)</type>
        </attribute>
        <attribute>
          <name>CITY</name>
          <type>VARCHAR(30)</type>
        </attribute>
        <attribute>
          <name>STATE_PROVINCE</name>
          <type>VARCHAR(10)</type>
        </attribute>
        <attribute>
          <name>COUNTRY_ID</name>
          <type>CHARACTER(2)</type>
        </attribute>
      </attributes>
    </type>
    <type>
      <name>ORDER_TYP</name>
      <category>udt</category>
      <instantiable>true</instantiable>
      <final>true</final>
      <attributes>
        <attribute>
          <name>ORDER_ID</name>
          <type>DECIMAL(12)</type>
        </attribute>
        <attribute>
          <name>ORDER_MODE</name>
          <type>VARCHAR(8)</type>
        </attribute>
        <attribute>
          <name>CUSTOMER_REF</name>
          <typeSchema>OE</typeSchema>
          <typeName>CUSTOMER_TYP</typeName>
        </attribute>
        <attribute>
          <name>ORDER_STATUS</name>
          <type>DECIMAL(2)</type>
        </attribute>
        <attribute>
          <name>ORDER_TOTAL</name>
          <type>DECIMAL(8, 2)</type>
        </attribute>
        <attribute>
      <name>SALES_REP_ID</name>
      <type>DECIMAL(6)</type>
    </attribute>
    <attribute>
      <name>ORDER_ITEM_LIST</name>
      <typeSchema>OE</typeSchema>
      <typeName>ORDER_ITEM_TYP</typeName>
      <cardinality>2147483647</cardinality>
    </attribute>
  </attributes>
</type>
<type>
  <name>CUSTOMER_TYP</name>
  <category>udt</category>
  <instantiable>true</instantiable>
  <final>true</final>
  <attributes>
    <attribute>
      <name>CUSTOMER_ID</name>
      <type>DECIMAL(6)</type>
    </attribute>
    <attribute>
      <name>CUST_FIRST_NAME</name>
      <type>VARCHAR(20)</type>
    </attribute>
    <attribute>
      <name>CUST_LAST_NAME</name>
      <type>VARCHAR(20)</type>
    </attribute>
    <attribute>
      <name>CUST_ADDRESS</name>
      <typeSchema>OE</typeSchema>
      <typeName>CUST_ADDRESS_TYP</typeName>
    </attribute>
    <attribute>
      <name>PHONE_NUMBERS</name>
      <type>VARCHAR(25)</type>
      <cardinality>5</cardinality>
    </attribute>
    <attribute>
      <name>NLS_LANGUAGE</name>
      <type>VARCHAR(3)</type>
    </attribute>
    <attribute>
      <name>NLS_TERRITORY</name>
      <type>VARCHAR(40)</type>
    </attribute>
    <attribute>
      <name>CREDIT_LIMIT</name>
      <type>DECIMAL(9, 2)</type>
    </attribute>
    <attribute>
      <name>CUST_EMAIL</name>
      <type>VARCHAR(40)</type>
    </attribute>
    <attribute>
      <name>CUST_ORDERS</name>
      <typeSchema>OE</typeSchema>
      <typeName>ORDER_TYP</typeName>
      <cardinality>2147483647</cardinality>
    </attribute>
  </attributes>
</type>
<type>
  <name>ORDER_ITEM_TYP</name>
  <category>udt</category>
  <instantiable>true</instantiable>
  <final>true</final>
  <attributes>
    <attribute>
      <name>ORDER_ID</name>
      <type>DECIMAL(12)</type>
    </attribute>
```

```
      <attribute>
        <name>LINE_ITEM_ID</name>
        <type>DECIMAL(3)</type>
      </attribute>
      <attribute>
        <name>UNIT_PRICE</name>
        <type>DECIMAL(8, 2)</type>
      </attribute>
      <attribute>
        <name>QUANTITY</name>
        <type>DECIMAL(8)</type>
      </attribute>
      <attribute>
        <name>PRODUCT_REF</name>
        <typeSchema>OE</typeSchema>
        <typeName>PRODUCT_INFORMATION_TYP</typeName>
      </attribute>
    </attributes>
  </type>
  <type>
    <name>PRODUCT_INFORMATION_TYP</name>
    <category>udt</category>
    <instantiable>true</instantiable>
    <final>true</final>
    <attributes>
      <attribute>
        <name>PRODUCT_ID</name>
        <type>DECIMAL(6)</type>
      </attribute>
      <attribute>
        <name>PRODUCT_NAME</name>
        <type>VARCHAR(50)</type>
      </attribute>
      <attribute>
        <name>PRODUCT_DESCRIPTION</name>
        <type>VARCHAR(2000)</type>
      </attribute>
      <attribute>
        <name>CATEGORY_ID</name>
        <type>DECIMAL(2)</type>
      </attribute>
      <attribute>
        <name>WEIGHT_CLASS</name>
        <type>DECIMAL(1)</type>
      </attribute>
      <attribute>
        <name>WARRANTY_PERIOD</name>
        <type>INTERVAL YEAR TO MONTH</type>
      </attribute>
      <attribute>
        <name>SUPPLIER_ID</name>
        <type>DECIMAL(6)</type>
      </attribute>
      <attribute>
        <name>PRODUCT_STATUS</name>
        <type>VARCHAR(20)</type>
      </attribute>
      <attribute>
        <name>LIST_PRICE</name>
        <type>DECIMAL(8, 2)</type>
      </attribute>
      <attribute>
        <name>MIN_PRICE</name>
        <type>DECIMAL(8, 2)</type>
      </attribute>
      <attribute>
        <name>CATALOG_URL</name>
        <type>VARCHAR(50)</type>
      </attribute>
      <attribute>
        <name>INVENTORY_LIST</name>
        <typeSchema>OE</typeSchema>
        <typeName>INVENTORY_TYP</typeName>
```

```
            <cardinality>2147483647</cardinality>
          </attribute>
        </attributes>
      </type>
      <type>
        <name>INVENTORY_TYP</name>
        <category>udt</category>
        <instantiable>true</instantiable>
        <final>true</final>
        <attributes>
          <attribute>
            <name>PRODUCT_ID</name>
            <type>DECIMAL(6)</type>
          </attribute>
          <attribute>
            <name>WAREHOUSE</name>
            <typeSchema>OE</typeSchema>
            <typeName>WAREHOUSE_TYP</typeName>
          </attribute>
          <attribute>
            <name>QUANTITY_ON_HAND</name>
            <type>DECIMAL(8)</type>
          </attribute>
        </attributes>
      </type>
      <type>
        <name>WAREHOUSE_TYP</name>
        <category>udt</category>
        <instantiable>true</instantiable>
        <final>true</final>
        <attributes>
          <attribute>
            <name>WAREHOUSE_ID</name>
            <type>DECIMAL(3)</type>
          </attribute>
          <attribute>
            <name>WAREHOUSE_NAME</name>
            <type>VARCHAR(35)</type>
          </attribute>
          <attribute>
            <name>LOCATION_ID</name>
            <type>DECIMAL(4)</type>
          </attribute>
        </attributes>
      </type>
    </types>
    <tables>
      <table>
        <name>CUSTOMERS</name>
        <folder>table0</folder>
        <description>Contains customers data either entered by an employee or by the customer him/herself over the Web.</description>
        <columns>
          <column>
            <name>CUSTOMER_ID</name>
            <type>DECIMAL(6)</type>
            <typeOriginal>"NUMBER"</typeOriginal>
            <nullable>false</nullable>
            <description>Primary key column.</description>
          </column>
          <column>
            <name>CUST_FIRST_NAME</name>
            <type>VARCHAR(20)</type>
            <typeOriginal>"VARCHAR2"</typeOriginal>
            <nullable>false</nullable>
            <description>NOT NULL constraint.</description>
          </column>
          <column>
            <name>CUST_LAST_NAME</name>
            <type>VARCHAR(20)</type>
            <typeOriginal>"VARCHAR2"</typeOriginal>
            <nullable>false</nullable>
            <description>NOT NULL constraint.</description>
          </column>
```

```xml
<column>
  <name>CUST_ADDRESS</name>
  <typeSchema>OE</typeSchema>
  <typeName>CUST_ADDRESS_TYP</typeName>
  <fields>
    <field>
      <name>STREET_ADDRESS</name>
    </field>
    <field>
      <name>POSTAL_CODE</name>
    </field>
    <field>
      <name>CITY</name>
    </field>
    <field>
      <name>STATE_PROVINCE</name>
    </field>
    <field>
      <name>COUNTRY_ID</name>
    </field>
  </fields>
  <description>Object column of type address_typ.</description>
</column>
<column>
  <name>PHONE_NUMBERS</name>
  <type>VARCHAR(25)</type>
  <typeOriginal>VARCHAR(25) ARRAY[5]</typeOriginal>
  <fields>
    <field>
      <name>PHONE_NUMBERS[1]</name>
    </field>
    <field>
      <name>PHONE_NUMBERS[2]</name>
    </field>
    <field>
      <name>PHONE_NUMBERS[3]</name>
    </field>
    <field>
      <name>PHONE_NUMBERS[4]</name>
    </field>
    <field>
      <name>PHONE_NUMBERS[5]</name>
    </field>
  </fields>
  <cardinality>5</cardinality>
  <description>Varray column of type phone_list_typ</description>
</column>
<column>
  <name>NLS_LANGUAGE</name>
  <type>VARCHAR(3)</type>
  <typeOriginal>"VARCHAR2"</typeOriginal>
</column>
<column>
  <name>NLS_TERRITORY</name>
  <type>VARCHAR(30)</type>
  <typeOriginal>"VARCHAR2"</typeOriginal>
</column>
<column>
  <name>CREDIT_LIMIT</name>
  <type>DECIMAL(9, 2)</type>
  <typeOriginal>"NUMBER"</typeOriginal>
  <description>Check constraint.</description>
</column>
<column>
  <name>CUST_EMAIL</name>
  <type>VARCHAR(40)</type>
  <typeOriginal>"VARCHAR2"</typeOriginal>
</column>
<column>
  <name>ACCOUNT_MGR_ID</name>
  <type>DECIMAL(6)</type>
  <typeOriginal>"NUMBER"</typeOriginal>
  <description>References hr.employees.employee_id.</description>
```

```
    </column>
    <column>
      <name>CUST_GEO_LOCATION</name>
      <typeSchema>MDSYS</typeSchema>
      <typeName>SDO_GEOMETRY</typeName>
      <fields>
        <field>
          <name>SDO_GTYPE</name>
        </field>
        <field>
          <name>SDO_SRID</name>
        </field>
        <field>
          <name>SDO_POINT</name>
          <fields>
            <field>
              <name>X</name>
            </field>
            <field>
              <name>Y</name>
            </field>
            <field>
              <name>Z</name>
            </field>
          </fields>
        </field>
        <field>
          <name>SDO_ELEM_INFO</name>
        </field>
        <field>
          <name>SDO_ORDINATES</name>
        </field>
      </fields>
      <description>SDO (spatial) column.</description>
    </column>
    <column>
      <name>DATE_OF_BIRTH</name>
      <type>DATE</type>
      <typeOriginal>"DATE"</typeOriginal>
    </column>
    <column>
      <name>MARITAL_STATUS</name>
      <type>VARCHAR(20)</type>
      <typeOriginal>"VARCHAR2"</typeOriginal>
    </column>
    <column>
      <name>GENDER</name>
      <type>VARCHAR(1)</type>
      <typeOriginal>"VARCHAR2"</typeOriginal>
    </column>
    <column>
      <name>INCOME_LEVEL</name>
      <type>VARCHAR(20)</type>
      <typeOriginal>"VARCHAR2"</typeOriginal>
    </column>
  </columns>
  <primaryKey>
    <name>CUSTOMERS_PK</name>
    <column>CUSTOMER_ID</column>
  </primaryKey>
  <foreignKeys>
    <foreignKey>
      <name>CUSTOMERS_ACCOUNT_MANAGER_FK</name>
      <referencedSchema>HR</referencedSchema>
      <referencedTable>EMPLOYEES</referencedTable>
      <reference>
        <column>ACCOUNT_MGR_ID</column>
        <referenced>EMPLOYEE_ID</referenced>
      </reference>
      <deleteAction>SET NULL</deleteAction>
      <updateAction>CASCADE</updateAction>
    </foreignKey>
  </foreignKeys>
```

```
    <rows>319</rows>
  </table>
  …
  <table>
    <name>WAREHOUSES</name>
    <folder>table7</folder>
    <description>Warehouse data unspecific to any industry.</description>
    <columns>
      <column>
        <name>WAREHOUSE_ID</name>
        <type>DECIMAL(3)</type>
        <typeOriginal>"NUMBER"</typeOriginal>
        <nullable>false</nullable>
        <description>Primary key column.</description>
      </column>
      <column>
        <name>WAREHOUSE_SPEC</name>
        <type>XML</type>
        <typeOriginal>"SYS"."XMLTYPE"</typeOriginal>
      </column>
      <column>
        <name>WAREHOUSE_NAME</name>
        <type>VARCHAR(35)</type>
        <typeOriginal>"VARCHAR2"</typeOriginal>
      </column>
      <column>
        <name>LOCATION_ID</name>
        <type>DECIMAL(4)</type>
        <typeOriginal>"NUMBER"</typeOriginal>
        <description>Primary key column, references hr.locations.location_id.</description>
      </column>
      <column>
        <name>WH_GEO_LOCATION</name>
        <typeSchema>MDSYS</typeSchema>
        <typeName>SDO_GEOMETRY</typeName>
        <fields>
          <field>
            <name>SDO_GTYPE</name>
          </field>
          <field>
            <name>SDO_SRID</name>
          </field>
          <field>
            <name>SDO_POINT</name>
            <fields>
              <field>
                <name>X</name>
              </field>
              <field>
                <name>Y</name>
              </field>
              <field>
                <name>Z</name>
              </field>
            </fields>
          </field>
          <field>
            <name>SDO_ELEM_INFO</name>
          </field>
          <field>
            <name>SDO_ORDINATES</name>
          </field>
        </fields>
        <description>SDO (spatial) column.</description>
      </column>
    </columns>
    <primaryKey>
      <name>WAREHOUSES_PK</name>
      <column>WAREHOUSE_ID</column>
    </primaryKey>
    <foreignKeys>
      <foreignKey>
        <name>WAREHOUSES_LOCATION_FK</name>
```

```
          <referencedSchema>HR</referencedSchema>
          <referencedTable>LOCATIONS</referencedTable>
          <reference>
            <column>LOCATION_ID</column>
            <referenced>LOCATION_ID</referenced>
          </reference>
          <deleteAction>SET NULL</deleteAction>
          <updateAction>CASCADE</updateAction>
        </foreignKey>
      </foreignKeys>
      <rows>9</rows>
    </table>
  </tables>
  <views>
    <view>
      <name>ACCOUNT_MANAGERS</name>
      <queryOriginal>SELECT .account_mgr_id _MGR, .region_id, .cust_address.country_id , .cust_address.state_province , (*)
        _CUSTOMERSFROM c, countries crWHERE .cust_address.country_id = cr.country_idGROUP BY ROLLUP (c.account_mgr_id,
        cr.region_id, c.cust_address.country_id, c.cust_address.state_province)</queryOriginal>
      <columns>
        <column>
          <name>ACCT_MGR</name>
          <type>DECIMAL(6)</type>
          <typeOriginal>"NUMBER"</typeOriginal>
        </column>
        <column>
          <name>REGION</name>
          <type>DECIMAL(22)</type>
          <typeOriginal>"NUMBER"</typeOriginal>
        </column>
        <column>
          <name>COUNTRY</name>
          <type>CHARACTER(2)</type>
          <typeOriginal>"CHAR"</typeOriginal>
        </column>
        <column>
          <name>PROVINCE</name>
          <type>VARCHAR(10)</type>
          <typeOriginal>"VARCHAR2"</typeOriginal>
        </column>
        <column>
          <name>NUM_CUSTOMERS</name>
          <type>DECIMAL(22)</type>
          <typeOriginal>"NUMBER"</typeOriginal>
        </column>
      </columns>
      <rows>0</rows>
    </view>
    …
  </views>
  <routines>
    <routine>
      <specificName>CATEGORY_DESCRIBE.CATALOG_TYP</specificName>
      <name>CATALOG_TYP</name>
    </routine>
    …
    <routine>
      <specificName>GET_PHONE_NUMBER_F</specificName>
      <name>GET_PHONE_NUMBER_F</name>
      <returnType>VARCHAR</returnType>
      <parameters>
        <parameter>
          <name>P_IN</name><mode>IN</mode><type>DECIMAL(38)</type><typeOriginal>NUMBER</typeOriginal>
        </parameter>
        <parameter>
          <name>P_PHONELIST</name><mode>IN</mode><type>VARCHAR(25)</type><cardinality>5</cardinality>
        </parameter>
      </parameters>
    </routine>
    …
  </routines>
</schema>
<schema>
```

```
      ...
    </schema>
  </schemas>
  <users>
    <user><name>OE</name></user>
    <user><name>HR</name></user>
  </users>
  <roles>
    <role><name>BI</name><admin>OE</admin></role>
    <role><name>PM</name><admin>OE</admin></role>
  </roles>
  <privileges>
    <privilege>
      <type>REFERENCES</type>
      <object>HR.COUNTRIES</object>
      <grantor>HR</grantor>
      <grantee>OE</grantee>
    </privilege>
      ...
  </privileges>
</siardArchive>
```

D.3 Examples of the XML schema definition of a table

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

D.3a table2.xsd (schema definition of a simple table)

```xml
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<xs:schema xmlns="http://www.bar.admin.ch/xmlns/siard/2/table.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.bar.admin.ch/xmlns/siard/2/table.xsd" elementFormDefault="qualified" attributeFormDefault="unqualified"
version="2.2">
  <!-- root element is the table element -->
  <xs:element name="table">
    <xs:complexType>
      <xs:annotation>
        <xs:documentation>Root element of a table of the SIARD archive. A table consists of rows.</xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:element name="row" type="recordType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="version" type="versionType" use="required"/>
    </xs:complexType>
  </xs:element>
  <!-- simple type for version number -->
  <xs:simpleType name="versionType">
    <xs:annotation>
      <xs:documentation>versionType is constrained to "2.2" for conformity with this XML schema</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="collapse"/>
      <xs:enumeration value="2.2"/>
      <!--  to be extended later with  <xs.enumeration value="2.2"/>  etc. -->
    </xs:restriction>
  </xs:simpleType>
  <!-- complex type record -->
  <xs:complexType name="recordType">
    <xs:annotation>
      <xs:documentation>row type of a table of the SIARD archive. A row consists of columns.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="c1" type="xs:decimal"/>
      <xs:element name="c2" type="xs:string" minOccurs="0"/>
      <xs:element name="c3" type="xs:string"/>
      <xs:element name="c4" type="xs:string"/>
      <xs:element name="c5" type="xs:string" minOccurs="0"/>
      <xs:element name="c6" type="dateType"/>
      <xs:element name="c7" type="xs:string"/>
      <xs:element name="c8" type="xs:decimal" minOccurs="0"/>
      <xs:element name="c9" type="xs:decimal" minOccurs="0"/>
      <xs:element name="c10" type="xs:decimal" minOccurs="0"/>
      <xs:element name="c11" type="xs:decimal" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <!-- date type between 0001 and 9999 restricted to UTC -->
  <xs:simpleType name="dateType">
    <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>
</xs:schema>
```

D.3b table7.xsd (schema definition of a table with internal large objects)

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<xs:schema xmlns="http://www.bar.admin.ch/xmlns/siard/2/table.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.bar.admin.ch/xmlns/siard/2/table.xsd" elementFormDefault="qualified" attributeFormDefault="unqualified"
version="2.2">
  <!-- root element is the table element -->
  <xs:element name="table">
    <xs:complexType>
      <xs:annotation>
        <xs:documentation>Root element of a table of the SIARD archive. A table consists of rows.</xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:element name="row" type="recordType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="version" type="versionType" use="required"/>
    </xs:complexType>
  </xs:element>
  <!-- simple type for version number -->
  <xs:simpleType name="versionType">
    <xs:annotation>
      <xs:documentation>versionType is constrained to "2.2" for conformity with this XML schema</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="collapse"/>
      <xs:enumeration value="2.2"/>
      <!--  to be extended later with  <xs.enumeration value="2.2"/>  etc. -->
    </xs:restriction>
  </xs:simpleType>
  <!-- complex type record -->
  <xs:complexType name="recordType">
    <xs:annotation>
      <xs:documentation>row type of a table of the SIARD archive. A row consists of columns.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="c1" type="xs:decimal"/>
      <xs:element name="c2" type="clobType" minOccurs="0"/>
      <xs:element name="c3" type="xs:string" minOccurs="0"/>
      <xs:element name="c4" type="xs:decimal" minOccurs="0"/>
      <xs:element name="c5" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="u1" type="xs:decimal" minOccurs="0"/>
            <xs:element name="u2" type="xs:decimal" minOccurs="0"/>
            <xs:element name="u3" minOccurs="0">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="u1" type="xs:decimal" minOccurs="0"/>
                  <xs:element name="u2" type="xs:decimal" minOccurs="0"/>
                  <xs:element name="u3" type="xs:decimal" minOccurs="0"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="u4" type="xs:decimal" minOccurs="0"/>
            <xs:element name="u5" type="xs:decimal" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <!-- type for text large objects -->
  <xs:complexType name="clobType">
    <xs:annotation>
      <xs:documentation>a text large object can either be stored inline (as xs:string) or externally (addressed by URI). The digest makes sure,
        that the connection to the external object is not completely lost. The length is in characters, not in bytes.</xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="file" type="xs:anyURI"/>
        <xs:attribute name="length" type="xs:integer"/>
        <xs:attribute name="digestType" type="digestTypeType"/>
        <xs:attribute name="digest" type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!--  type for message digest type -->
  <xs:simpleType name="digestTypeType">
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="collapse"/>
      <xs:enumeration value="MD5"/>
      <xs:enumeration value="SHA-1"/>
      <xs:enumeration value="SHA-256"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

D.3c table0.xsd (schema definition of a table with udt and array)

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<xs:schema xmlns="http://www.bar.admin.ch/xmlns/siard/2/table.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.bar.admin.ch/xmlns/siard/2/table.xsd" elementFormDefault="qualified" attributeFormDefault="unqualified" version="2.2">
  <!-- root element is the table element -->
  <xs:element name="table">
    <xs:complexType>
      <xs:annotation>
        <xs:documentation>Root element of a table of the SIARD archive. A table consists of rows.</xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:element name="row" type="recordType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="version" type="versionType" use="required"/>
    </xs:complexType>
  </xs:element>
  <!-- simple type for version number -->
  <xs:simpleType name="versionType">
    <xs:annotation>
      <xs:documentation>versionType is constrained to "2.2" for conformity with this XML schema</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="collapse"/>
      <xs:enumeration value="2.2"/>
      <!--  to be extended later with <xs.enumeration value="2.2"/>  etc. -->
    </xs:restriction>
  </xs:simpleType>
  <!-- complex type record -->
  <xs:complexType name="recordType">
    <xs:annotation>
      <xs:documentation>row type of a table of the SIARD archive. A row consists of columns.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="c1" type="xs:decimal"/>
      <xs:element name="c2" type="xs:string"/>
      <xs:element name="c3" type="xs:string"/>
      <xs:element name="c4" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="u1" type="xs:string" minOccurs="0"/>
            <xs:element name="u2" type="xs:string" minOccurs="0"/>
            <xs:element name="u3" type="xs:string" minOccurs="0"/>
            <xs:element name="u4" type="xs:string" minOccurs="0"/>
            <xs:element name="u5" type="xs:string" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="c5" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="a1" type="xs:string" minOccurs="0"/>
            <xs:element name="a2" type="xs:string" minOccurs="0"/>
            <xs:element name="a3" type="xs:string" minOccurs="0"/>
            <xs:element name="a4" type="xs:string" minOccurs="0"/>
            <xs:element name="a5" type="xs:string" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="c6" type="xs:string" minOccurs="0"/>
      <xs:element name="c7" type="xs:string" minOccurs="0"/>
      <xs:element name="c8" type="xs:decimal" minOccurs="0"/>
      <xs:element name="c9" type="xs:string" minOccurs="0"/>
      <xs:element name="c10" type="xs:decimal" minOccurs="0"/>
      <xs:element name="c11" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="u1" type="xs:decimal" minOccurs="0"/>
            <xs:element name="u2" type="xs:decimal" minOccurs="0"/>
            <xs:element name="u3" minOccurs="0">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="u1" type="xs:decimal" minOccurs="0"/>
                  <xs:element name="u2" type="xs:decimal" minOccurs="0"/>
                  <xs:element name="u3" type="xs:decimal" minOccurs="0"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="u4" type="xs:decimal" minOccurs="0"/>
            <xs:element name="u5" type="xs:decimal" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="c12" type="dateType" minOccurs="0"/>
      <xs:element name="c13" type="xs:string" minOccurs="0"/>
      <xs:element name="c14" type="xs:string" minOccurs="0"/>
      <xs:element name="c15" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <!-- date type between 0001 and 9999 restricted to UTC -->
  <xs:simpleType name="dateType">
    <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>
</xs:schema>
```

D.4 Examples of the table data of a table

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

D.4a table2.xml (simple table)

```
<?xml version="1.0" encoding="UTF-8"?>
<table xmlns="http://www.bar.admin.ch/xmlns/siard/2/table.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bar.admin.ch/xmlns/siard/2/table.xsd table2.xsd" version="2.2">
  <row><c1>100</c1><c2>Steven</c2><c3>King</c3><c4>SKING</c4><c5>515.123.4567</c5>
    <c6>2003-06-16Z</c6><c7>AD_PRES</c7><c8>24000</c8><c11>90</c11></row>
  <row><c1>101</c1><c2>Neena</c2><c3>Kochhar</c3><c4>NKOCHHAR</c4><c5>515.123.4568</c5>
    <c6>2005-09-20Z</c6><c7>AD_VP</c7><c8>17000</c8><c10>100</c10><c11>90</c11></row>
  <row><c1>102</c1><c2>Lex</c2><c3>De Haan</c3><c4>LDEHAAN</c4><c5>515.123.4569</c5>
    <c6>2001-01-12Z</c6><c7>AD_VP</c7><c8>17000</c8><c10>100</c10><c11>90</c11></row>
  <row><c1>103</c1><c2>Alexander</c2><c3>Hunold</c3><c4>AHUNOLD</c4><c5>590.423.4567</c5>
    <c6>2006-01-02Z</c6><c7>IT_PROG</c7><c8>9000</c8><c10>102</c10><c11>60</c11></row>
  <row><c1>104</c1><c2>Bruce</c2><c3>Ernst</c3><c4>BERNST</c4><c5>590.423.4568</c5>
    <c6>2007-05-20Z</c6><c7>IT_PROG</c7><c8>6000</c8><c10>103</c10><c11>60</c11></row>
  <row><c1>105</c1><c2>David</c2><c3>Austin</c3><c4>DAUSTIN</c4><c5>590.423.4569</c5>
    <c6>2005-06-24Z</c6><c7>IT_PROG</c7><c8>4800</c8><c10>103</c10><c11>60</c11></row>
  <row><c1>106</c1><c2>Valli</c2><c3>Pataballa</c3><c4>VPATABAL</c4><c5>590.423.4560</c5>
    <c6>2006-02-04Z</c6><c7>IT_PROG</c7><c8>4800</c8><c10>103</c10><c11>60</c11></row>
  <row><c1>107</c1><c2>Diana</c2><c3>Lorentz</c3><c4>DLORENTZ</c4><c5>590.423.5567</c5>
    <c6>2007-02-06Z</c6><c7>IT_PROG</c7><c8>4200</c8><c10>103</c10><c11>60</c11></row>
  <row><c1>108</c1><c2>Nancy</c2><c3>Greenberg</c3><c4>NGREENBE</c4><c5>515.124.4569</c5>
    <c6>2002-08-16Z</c6><c7>FI_MGR</c7><c8>12008</c8><c10>101</c10><c11>100</c11></row>
  …
  <row><c1>206</c1><c2>William</c2><c3>Gietz</c3><c4>WGIETZ</c4><c5>515.123.8181</c5><c6>2002-06-
06Z</c6><c7>AC_ACCOUNT</c7><c8>8300</c8><c10>205</c10><c11>110</c11></row>
</table>
```

D.4b table7.xml (table with internal large object)

```
<?xml version="1.0" encoding="UTF-8"?>
<table xmlns="http://www.bar.admin.ch/xmlns/siard/2/table.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bar.admin.ch/xmlns/siard/2/table.xsd table7.xsd" version="2.2">
  <row>
    <c1>1</c1>
    <c2 file="content/schema1/table7/lob1/record0.xml" length="270" digestType="MD5" digest="BCA4FB6D6898A2F42C624839B431C386"/>
    <c3>Southlake, Texas</c3>
    <c4>1400</c4>
    <c5><u1>2001</u1><u2>8307</u2><u3><u1>-103.00195</u1><u2>36.500374</u2></u3></c5>
  </row>
  <row>
    <c1>2</c1>
    <c2 file="content/schema1/table7/lob1/record1.xml" length="268" digestType="MD5" digest="7E99F05D8C4D7D3909D3F20987A0DE41"/>
    <c3>San Francisco</c3>
    <c4>1500</c4>
    <c5><u1>2001</u1><u2>8307</u2><u3><u1>-124.21014</u1><u2>41.998016</u2></u3></c5>
  </row>
  <row>
    <c1>3</c1>
    <c2 file="content/schema1/table7/lob1/record2.xml" length="235" digestType="MD5" digest="C495BB25A6EDBFE829DDB9B28C027DC3"/>
    <c3>New Jersey</c3>
    <c4>1600</c4>
    <c5><u1>2001</u1><u2>8307</u2><u3><u1>-74.695305</u1><u2>41.35733</u2></u3></c5>
  </row>
  …
  <row>
    <c1>9</c1><c3>Bombay</c3><c4>2100</c4>
  </row>
</table>
```

D.4c table0.xml (table with udt and array)

```
<?xml version="1.0" encoding="UTF-8"?>
<table xmlns="http://www.bar.admin.ch/xmlns/siard/2/table.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bar.admin.ch/xmlns/siard/2/table.xsd table0.xsd" version="2.2">
  <row>
    <c1>232</c1>
    <c2>Donald</c2>
    <c3>Hunter</c3>
    <c4><u1>5122 Sinclair Ln</u1><u2>21206</u2><u3>Baltimore</u3><u4>MD</u4><u5>US</u5></c4>
    <c5><a1>+1 410 123 4795</a1></c5>
    <c6>us</c6>
    <c7>AMERICA</c7>
    <c8>2400</c8>
    <c9>Donald.Hunter@CHACHALACA.EXAMPLE.COM</c9>
    <c10>145</c10>
    <c11><u1>2001</u1><u2>8307</u2><u3><u1>-76.545732</u1><u2>39.322775</u2></u3></c11>
    <c12>1960-01-19Z</c12>
    <c13>married</c13>
    <c14>M</c14>
    <c15>G: 130,000 - 149,999</c15>
  </row>
  <row>
    <c1>233</c1>
    <c2>Graham</c2>
    <c3>Spielberg</c3>
    <c4><u1>680 Bel Air Rd</u1><u2>21014</u2><u3>Bel Air</u3><u4>MD</u4><u5>US</u5></c4>
    <c5><a1>+1 410 123 4800</a1></c5>
    <c6>us</c6>
    <c7>AMERICA</c7>
    <c8>2400</c8>
    <c9>Graham.Spielberg@CHUKAR.EXAMPLE.COM</c9>
    <c10>145</c10>
    <c11><u1>2001</u1><u2>8307</u2><u3><u1>-76.357073</u1><u2>39.523878</u2></u3></c11>
    <c12>1970-01-28Z</c12>
    <c13>single</c13>
    <c14>M</c14>
    <c15>D: 70,000 - 89,999</c15>
  </row>
  …
  </row>
    <row>
    <c1>235</c1>
    <c2>Edward</c2>
    <c3>Oates</c3>
    <c4><u1>8004 Stansbury Rd</u1><u2>21222</u2><u3>Baltimore</u3><u4>MD</u4><u5>US</u5></c4>
    <c5><a1>+1 410 012 4715</a1><a2>+1 410 083 4715</a2></c5>
    <c6>us</c6>
    <c7>AMERICA</c7>
    <c8>2400</c8>
    <c9>Edward.Oates@OVENBIRD.EXAMPLE.COM</c9>
    <c10>145</c10>
    <c11><u1>2001</u1><u2>8307</u2><u3><u1>-76.500344</u1><u2>39.25618</u2></u3></c11>
    <c12>1955-03-20Z</c12>
    <c13>married</c13>
    <c14>M</c14>
    <c15>E: 90,000 - 109,999</c15>
  </row>
  …
</table>
```

Appendix E - Example with segmenting internal LOBs

Below is an example of the folder structure for the example database Northwind with internal LOBs stored outside the SIARD file. Northwind includes the following tables, which in our example are ordered this way:

Orders	(table0)
Products	(table1)
Categories	(table2) - has LOBs which exceed 2000 bytes or characters
Shippers	(table3)
Employees	(table4) - has LOBs which exceed 2000 bytes or characters
Territories	(table5)
CustomerDemographics	(table6)
CustomerCostumerDemo	(table7)
Suppliers	(table8)
EmployeeTerritories	(table9)
Customers	(table10)
Sysdiagrams	(table11)
Region	(table12)

Only the table 'Categories' (table2) below is used in this example

CategoryID	CategoryName	Description	Picture
1	Beverages	Soft drinks, coffees, teas, beers, and ales	BLOB (Size: 10151)
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings	BLOB (Size: 12107)
3	Confections	Desserts, candies, and sweet breads	BLOB (Size: 12007)
4	Dairy Products	Cheeses	BLOB (Size: 9756)
5	Grains/Cereals	Breads, crackers, pasta, and cereal	BLOB (Size: 12131)
6	Meat/Poultry	Prepared meats	BLOB (Size: 11280)
7	Produce	Dried fruit and bean curd	BLOB (Size: 12338)
8	Seafood	Seaweed and fish	BLOB (Size: 12069)

Limits for file number and for total file size per folder

The file number limit per folder is in this case set to 4 and the total file size limit per folder is set to 45,000 bytes (unrealistic but useful as an example).

Rows 1, 2, 3 and 4 will have their LOBs from column 4 ('Picture') (`t2_c4_r1.bin`, `t2_c4_r2.bin`, `t2_c4_r3.bin`, `t2_c4_r4.bin`) stored together in a folder named `seg_0`
Here the *file number limit* of 4 is reached, meaning no more files can be stored in this folder, so a new folder is created `seg_1`.

Rows 5, 6, and 7 will have their LOBs from column 4 ('Picture') (`t2_c4_r5.bin, t2_c4_r6.bin, t2_c4_r7.bin`) stored together in a folder named `seg_1`.
Row 8 will *not* have its LOB from column 4 ('Picture') (`t2_c4_r8.bin`) stored together with the ones from rows 5, 6 and 7 in the folder named `seg_1`. Not because the *file number limit* of 4 is reached, but because the *accumulated file size per folder limit* of 45,000 bytes is reached.

The LOBs from rows 5, 6 and 7 have respectively a size of 12,131, 11,280 and 12,338 bytes which accumulates to 35,749 bytes. Adding the LOB from row 8 with a size of 12,069 bytes to the 35,749 bytes of rows 5, 6 and 7 would accumulate to 47,818 bytes and exceed the *accumulated file size per* *folder limit* of 45,000 bytes. Therefore a new folder is created named `seg_2`, and Row 8 will have its LOB from column 4 ('Picture') (`t2_c4_r8.bin`) stored in it.

Illustration of the folder structure for the example

```
Northwind.siard <!-- packaged as a ZIP file ->
     content/
     header/
          metadata.xml
          metadata.xsd
          siardversion/
               2.2/

Northwind_lobs/s0_t2_c4/seg_0/t2_c4_r1.bin
Northwind_lobs/s0_t2_c4/seg_0/t2_c4_r2.bin
Northwind_lobs/s0_t2_c4/seg_0/t2_c4_r3.bin
Northwind_lobs/s0_t2_c4/seg_0/t2_c4_r4.bin      <!-- folder file number limit -->
Northwind_lobs/s0_t2_c4/seg_1/t2_c4_r5.bin
Northwind_lobs/s0_t2_c4/seg_1/t2_c4_r6.bin
Northwind_lobs/s0_t2_c4/seg_1/t2_c4_r7.bin      <!-- folder file size limit  -->
Northwind_lobs/s0_t2_c4/seg_2/t2_c4_r8.bin
```

Extract of metadata.xml for the example

```
<siardArchive>...<lobFolder>./Northwind_lobs/</lobFolder>
<dbname>Northwind</dbname>


..
<column>...<lobFolder>s0_t2_c4/</lobFolder>...</column>
```

Extract of table2.xml for the example

```
<row><c1>1</c1><c2>Beverages</c2><c3>Soft drinks, coffees, teas, beers, and ales</c3>
<c4 file="seg_0/t2_c4_r1.bin" length="10151" digestType="md5"
digest="74f24080fc9d234d3ac221b8e743c763"/></row>
<row><c1>2</c1><c2>Condiments</c2><c3>Sweet and savory sauces, relishes, spreads, and seasonings</c3>
<c4 file="seg_0/t2_c4_r2.bin" length="12107" digestType="md5"
digest="22a0cbe8960b78ce48b07a285ce69e3c"/></row>
<row><c1>3</c1><c2>Confections</c2><c3>Desserts, candies, and sweet breads</c3>
<c4 file="seg_0/t2_c4_r3.bin" length="12007" digestType="md5"
digest="3e2f2028a9147c29bdcd36ed4e5f25b3"/></row>
<row><c1>4</c1><c2>Dairy Products</c2><c3>Cheeses</c3>
<c4 file="seg_0/t2_c4_r4.bin" length="9756" digestType="md5"
digest="12f588040e11cc2021ea37d46aa10c51"/></row>
<row><c1>5</c1><c2>Grains/Cereals</c2><c3>Breads, crackers, pasta, and cereal</c3>
<c4 file="seg_1/t2_c4_r5.bin" length="12131" digestType="md5"
digest="e2d8ef03e1b24edd946820dbbf44fdfd"/></row>
<row><c1>6</c1><c2>Meat/Poultry</c2><c3>Prepared meats</c3>
<c4 file="seg_1/t2_c4_r6.bin" length="11280" digestType="md5"
digest="814a3eb95253c08137f70bcfc279e00f"/></row>
<row><c1>7</c1><c2>Produce</c2><c3>Dried fruit and bean curd</c3>
<c4 file="seg_1/t2_c4_r7.bin " length="12338" digestType="md5"
digest="ee114cd7700f566b1f7c7e8e0f68ca0f"/></row>
```

```
<row><c1>8</c1><c2>Seafood</c2><c3>Seaweed and fish</c3>
<c4 file="seg_2/t2_c4_r8.bin" length="12069" digestType="md5"
digest="2de1ac4c4e8ebb853e17db01af3fb7c3"/></row>
```

Resolving the URL for the example

According to SIARD based on the RFC 8089 the base URI and the relative-path reference in the example for row 4 will resolve to this:

Base URI:	`./Northwind_lobs/`
Column URI:	`s0_t2_c4/`
File URI:	`seg_0/t2_c4_r4.bin`
Resolves into:	`./Northwind_lobs/s0_t2_c4/seg_0/t2_c4_r4.bin`

where the `./` from the base URI `./Northwind_lobs/` relates to the position of the `metadata.xml` file in the folder `header` inside the `Northwind.siard` zip file, i.e the folder `Northwind_lobs/` is at the same level as the Northwind.siard file.

Directions from the XML standard, escaping spaces

The SIARD 2.2 format uses the XML datatype xs:anyURI for representing URIs[1]. According to the W3C Recommendation[2] the datatype xs:anyURI can have values that can be absolute or relative, and may have optional fragment identifiers. The W3C recommendation refers to RFC 2396[3] and RFC 2732[4] for definitions. RFC 2396 and RFC 2732 are the standards that lie between RFC 1738 and the most current RFC 8089, all of which are already presented in this recommendation.
The W3C recommendation also includes a note that the lexical space, i.e. the set of valid literals for the datatype, in principle allows spaces, however, their use is highly discouraged unless they are encoded as %20. For spaces in URIs the SIARD 2.2 format follows the W3C recommendation, but is recommended not to have space characters in the content of the *lobFolder* elements, as shown in this example: `Northwind_lobs/s0_t2_c4/seg_0/t2_c4_r4.bin`

Algorithm prefix to hash value required in SIARD 2.0, changed in 2.1

Note that according to the SIARD 2.0 format specification the messageDigest value is prefixed with the algorithm followed by the hash value. After SIARD 2.1 the value and algorithm are separated.

Lower case for messageDigest recommended

Note that the messageDigest indicates hexadecimal values and it is therefore not of strict importance whether they are set in upper or lower case. However, lower case is mostly used and enforced, see e.g. RFC 2831 https://www.ietf.org/rfc/rfc2831.txt

Appendix F - Handling large objects in relational databases

Large object (LOB) is the common name for Binary Large Object (BLOB) and Character Large Object (CLOB). BLOB is content such as video, sound, images, word processing documents etc., and CLOB is text content. These LOBs can be stored inside a relational database as BLOBs or CLOBs or outside as DATALINKs (SQL:2008 SQL/MED).

Binary data in regard to relational databases is defined as data for which no simple datatype (such as integer or character) exists. In addition, the size of binary data is also important due to the need for efficient handling in databases. Binary data is mostly referred to as a binary large object (BLOB). Similarly, large amounts of character data are named CLOB, they pose a problem due to size more than the lack of a proper data type. In this specification CLOBs will be treated as BLOBs, and both are generally named LOBs (large objects).

Databases and their handling of large objects (LOBs) has always been a challenge, regardless of whether the handling was based on:

1. Internal LOBs - contained in the records in the RDB.
2. External LOBs - stored as files outside the RDB, and referred to by path (URL)

LOB handling in SQL Standards

The first method using internal LOBs has been available for many versions of the SQL standard. It is supported by all current relational database management systems.

The other method using external files has been available since SQL:2003 as the Management of External Data (SQL/MED). It is partially supported by current relational database management systems, and perhaps due to lack of details in the SQL standard those RDBMS's that do support it partially do so differently.

LOB handling in the SIARD 2.2 format

The SIARD 2.2 format is based on the SQL:2008 standard.

Support for internal LOBs (ISO/IEC 9075-2:2008 - BLOBS) in SIARD 2.2

The SIARD 2.2 format specification supports the SQL:2008 method for using internal LOBS (ISO/IEC 9075-2:2008), as did SIARD 1.0 (SQL:1999).

The SIARD 2.2 format supports LOBs stored as files inside the SIARD file and describes this in detail in the SIARD 2.2 format specification (similar to SIARD 1.0).

The SIARD 2.2 format supports LOBs stored as files outside the SIARD file (a new feature in SIARD 2.0) and specifies the details in this specification.

Support for external files (ISO/IEC 9075-9:2008 – SQL/MED) in SIARD 2.2

The SIARD 2.2 format supports the SQL:2008 method for using external files (ISO/IEC 9075-9:2008 – SQL/MED).

Appendix G - SIARD and ZIP

The SIARD specification G_3.2-1 states that *“A relational database is archived in a single SIARD file”*. But this does not necessarily mean that this file cannot be segmented. SIARD is using ZIP (32 bit and 64 bit format) and ZIP supports splitting a ZIP file into several ZIP segments of a given size[20] (max 4 GiB per segment for ZIP64).

As such the ZIP64 format should be suitable for the task of splitting a huge SIARD file into manageable segments:

```
myDatabase.siard.z1
myDatabase.siard.z2
…
myDatabase.siard.z(n-1)
myDatabase.siard.zip
```

The final ZIP segment file contains the directory of the ZIP archive. ZIP files can also be streamed.

All together the ZIP format feature for splitting a zip file into segments together with the ZIP64 format feature for higher limits, the central directory and the streaming possibility should make the ZIP64 format a sufficient solution for handling large SIARD files.
Hopefully in the very near future, just using these built-in format features should be sufficient, but currently it is not deemed to be so.
The reasons are among others:

Lack of widespread support in applications (including programming tools) for properly splitting of a ZIP archive into segments according to the ZIP64 specification - such as a limitation in the maximum number of segments possible, or the maximum segment size (which are sometimes way below the limits of the format specification – currently only 4 GiB, which is another issue.).

Likewise ISO/IEC 21320-1:2015(E) "Document Container File" supports ZIP64, but not Appnote section 8.0 (splitting and spanning Zip files).

Lack of widespread support in applications in efficient handling of segments, such as the need for creating the complete ZIP file before being able to split it; and the need to assemble the whole zip file before addressing or extracting parts of it.

As soon as application support is deemed sufficient, a new version of this specification based on automatic segmentation will be issued, but until then manual segmentation is used as specified in this document.

[20] https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT section 8.5.1

Maximum number of segments = 4,294,967,295 - 1; Maximum .ZIP segment size = 4,294,967,295 bytes

Appendix H – Changes from version 1.0

The following changes have been introduced from version 1.0 to version 2.2[21].

Chapter / ID / Document	Change	RFC
passim	Optional requirements: it is specified whether a field must be filled in or can be left empty.	2013-23
passim, 5.3, 5.4, 5.7, 6.2 & 6.3	Upgrade of SQL:1999 support to SQL:2008 support	2014-110
Title page, chapters 3-6, Appendix D	All examples refer to the newly enclosed example file ech-0165_oe.siard	
Summary	Updated, explains usage of the previous version.	
1, 2	Numbering of chapters according to the new template.	
2.2, passim	ID requirement G instead of A	
G_3.2-1, G_3.4-3, P_4.2-3, passim, metadata.xsd	Adaptations regarding large objects incl. attributes *digest* und *digestType*. Support for storing large objects outside of the SIARD file.	2015-29
G_3.2-2	Deleted because it was no format definition but an organisational requirement	
G_4.1-2	Support for “deflate” as a compression mechanism	Addendum
P_4.2-4, P_4.2-5	Format identification. To complement existing identification methods an empty folder header/siardversion/2.1/ exists in order to improve understanding and identification.	2015-12
P_4.3-3, passim, metadata.xsd	Upgrade of SQL:1999 support to SQL:2008 support. The SQL:2008 abbreviations have been integrated.	2014-110
5.4, metadata.xsd	Nullable-Element of the attribute introduced	
5.6, 5.7	Folder replaced by lobfolder in the columns and fields areas	
T_6.1-2, T_6.1-4	Start of consecutive numbering has been stated more precisely	
T_6.1-3, T_6.3-2, passim, metadata.xsd	Adaptations in the date, time, and timestamp areas	

[21] Changes between versions 2.1 and 2.1.1 are limited to precisions in the wording of the summary and of G_3.3-4.

5.13, metadata.xsd	actionTime of the triggers augmented with INSTEAD OF
metadata.xsd	metadata.xsd has been adapted in different places, for the following reasons: adjustment to SQL:2008 standard (identifiers and type elements), adaptation of regular expressions for the pre-defined data types, implementation of the changes mentioned above, adaptation to the specification, aggregation of primaryKeyType and candidateKeyType to uniqueKeyType. Addition of DATALINK from SQL:2008 part 9. Addition of clobType and blobType.