BS ISO/IEC 12034-1:2017

BSI Standards Publication

# Information technology — Archive eXchange Format (AXF)

Part 1: Structure and semantics

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

# National foreword

This British Standard is the UK implementation of ISO/IEC 12034-1:2017.

The UK participation in its preparation was entrusted to Technical Committee IST/-/99, Participating IST members.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2018
Published by BSI Standards Limited 2018

ISBN 978 0 580 98179 1

ICS 37.080

**Compliance with a British Standard cannot confer immunity from legal obligations.**

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 January 2018.

**Amendments/corrigenda issued since publication**

| Date | Text affected |
|---|---|

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

INTERNATIONAL STANDARD

# ISO/IEC 12034-1

First edition
2017-12

# Information technology — Archive eXchange Format (AXF) —

## Part 1: Structure and semantics

*Technologies de l'information — Archive eXchange Format (AXF) — Partie 1: Structure et sémantique*

Reference number
ISO/IEC 12034-1:2017(E)

Documents
Document Sharing Hub

© ISO/IEC 2017

Full document details: BS ISO/IEC 12034-1:2017

**COPYRIGHT PROTECTED DOCUMENT**

© ISO/IEC 2017, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

## Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

ISO/IEC 12034-1:2017 was prepared by the Society of Motion Picture and Television Engineers (SMPTE) and was adopted, under the PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by the national bodies of ISO and IEC.

© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

**ISO/IEC 12034-1**
**SMPTE ST 2034-1:2017**

# ISO/IEC 12034-1 STANDARD

# Archive eXchange Format (AXF) — Part 1: Structure & Semantics



## Table of Contents

Page


Foreword ..... 3
Intellectual Property ..... 3
Introduction ..... 3
1 Scope ..... 7
2 Conformance Notation ..... 7
3 Normative References ..... 7
4 Definitions ..... 8
5 Storage Media Types ..... 13
5.1 Media with File Systems ..... 14
5.2 Media without File Systems ..... 14
5.3 File Marks ..... 15
5.4 Relationships Between AXF Structures and Storage Media Types ..... 15
6 Archive eXchange Format (AXF) Structure ..... 16
6.1 Form of Data Expression ..... 16
6.2 Byte Order ..... 17
6.3 General AXF Concepts ..... 17
6.4 AXF Data Structures ..... 18
7 General Usage Considerations ..... 36
7.1 File Naming ..... 36
7.2 Media Preparation ..... 36
7.3 AXF Object Index Structures ..... 37
7.4 Creating, Reading, Writing, Copying, and Transferring AXF Objects ..... 38
7.5 Nesting AXF Objects ..... 39
8 Spanning ..... 39
8.1 Spanning Linkages ..... 39
8.2 Encountering a Spanning Situation ..... 43
8.3 Recovery of Spanned AXF Objects ..... 43
8.4 Spanning Rules ..... 43
9 Collected Sets ..... 44
9.1 Collected Set Linkages ..... 44
9.2 Collected Set Structure ..... 45
9.3 Add/Replace/Delete Processes ..... 45
9.4 Tracking Versions ..... 46


© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

10 AXF Data Model ..... 48
10.1 AXF Medium Identifier ..... 49
10.2 Object Header ..... 52
10.3 Object Fragment Header ..... 58
10.4 File Footer ..... 61
10.5 Object Fragment Footer ..... 63
10.6 Object Footer ..... 68
10.7 AXF Object Index ..... 74
10.8 UUID ..... 78
10.9 PositionInteger ..... 78
10.10 FileFolder ..... 79
10.11 Folder ..... 81
10.12 File ..... 84
10.13 Symlink ..... 87
10.14 FileTree ..... 90
10.15 Application ..... 91
10.16 Entity ..... 93
10.17 Location ..... 95
10.18 Identifiers ..... 97
10.19 Checksums ..... 97
10.20 Identifier ..... 97
10.21 Checksum ..... 98
10.22 ByteOrder ..... 100
10.23 Media Type ..... 101
10.24 Structure Version ..... 101



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

## Foreword

SMPTE (the Society of Motion Picture and Television Engineers) is an internationally-recognized standard developing organization. Headquartered and incorporated in the United States of America, SMPTE has members in over 80 countries on six continents. SMPTE's Engineering Documents, including Standards, Recommended Practices, and Engineering Guidelines, are prepared by SMPTE's Technology Committees. Participation in these Committees is open to all with a bona fide interest in their work. SMPTE cooperates closely with other standards-developing organizations, including ISO, IEC and ITU.

SMPTE Engineering Documents are drafted in accordance with the rules given in its Standards Operations Manual.

SMPTE ST 2034-1 was prepared by Technology Committee 31FS on File Formats and Systems.

## Intellectual Property

At the time of publication, no notice had been received by SMPTE claiming patent rights essential to the implementation of this Engineering Document. However, attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. SMPTE shall not be held responsible for identifying any or all such patent rights.

## Introduction

This section is entirely informative and does not form an integral part of this Engineering Document.

The Archive eXchange Format (AXF) is an open format that supports interoperability among disparate data storage systems and ensures long–term availability of data, no matter how storage or file system technologies evolve. AXF inherently supports interoperability between existing, discrete storage systems, irrespective of the operating and file systems used, and also future–proofs digital storage by abstracting the underlying technology so that content remains available across generations of technology development.

At the most basic level, AXF is a file container that can encapsulate any number, size, and type of files in a fully self–contained and self–describing package. The package contains its own light-weight file system, which establishes independence from underlying operating systems, storage technologies, and file systems and can store any type of data on any type of storage media. Inside its packaging, AXF can contain metadata of any format, applicable to either AXF Objects or to individual files contained within AXF Objects; AXF also carries key preservation information, such as provenance, fixity, and the like — all key to ensuring long-term robustness and recoverability.

Historically, digital archive systems have used media data storage formats that are proprietary to their manufacturers, either intentionally or due to the lack of established standards. There have been neither interchange of media nor interoperability of archive systems between manufacturers and in some cases between different archive systems from the same manufacturer. Archives could be orphaned due to support ending for the systems used to create data archives. End users and manufacturers recognized that the proprietary nature of archive systems and the data stores that they create result in significant costs of operation that are unnecessary. These costs could be avoided if there were standardization of the format used for storage of the data on media and for transfer of the data between systems and locations. AXF permits separating the stored content from the systems that create and recover sets of data, thereby enabling refreshing of storage technology, recovering sets of data that otherwise would have been orphaned, and transferring sets of data between systems and locations.

This standard specifies a structure for data that can be written to any current or future data storage subsystem, regardless of the type of media on which it is stored. The data can include any types of files and associated metadata that are stored and transferred together in a structure called an "AXF Object." A single AXF Object can be spanned across multiple physical media, can be copied from one set of physical media to another, and



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

is agnostic to the Storage Media Type on which it is stored, e.g., spinning disc or linear tape. Regardless of the Storage Media Types on which they are stored, AXF Objects are identically structured and formatted for any given set and relationship of contained files and metadata.

AXF initially arose from the storage needs of the audiovisual production and archiving communities but quickly encompassed any type of file-based data. The transition to file-based workflows led to a new set of requirements throughout pre-production, production, distribution, storage, and preservation processes. Those requirements included long-term archiving of finished and unfinished materials, writing data to any type of storage subsystem using a standard scheme, transporting formatted archives between systems and locations using either media or networks, and allowing extensibility sufficient to accommodate any type of file, of any size, from any source, as well as adoption of any future storage technologies. AXF was created to address these requirements.

Audiovisual content archiving spans a wide range of content and data archiving systems and practices. At the time this standard was written, many different methods and media were commonly used to store file-based audiovisual content and its supporting information. Examples range from individual hard drives, solid state drives, and linear magnetic tape drives in small organizations to large spinning disc arrays in combination with very large robotic systems with multiple robots, each having multiple drives, in very large cultural, scientific, and legal archives. Applications in other industries that could benefit from the methods defined herein include medical imaging, geophysical exploration, scientific research, and similar high-volume producers of data.

The cultural, scientific, and business value of assets stored on these data systems is significant. Methods for storage, interchange, transport, and preservation of such assets, both locally and remotely, over both short and very long retention periods, demands a standardized, well-documented, non-manufacturer-specific method of writing data to any data storage system, from which the data then can be recovered and its contents used, updated, or transferred to another data storage system. All that would be necessary to achieve these objectives is a mechanism for recovering data from the media on which it is stored, plus utilities or applications that implement AXF.

The AXF standard creates a common method of writing individual files or related sets of files, and relevant metadata, onto data storage subsystems so that the structure of an AXF Object will remain the same no matter what vendor equipment or Storage Media Type is used. As long as the media remains viable and data can be read from that media, it will be possible to recover an AXF Object and unwrap its contents with a suitable utility or application running on whatever platform is current at the time. The AXF Object also has to be able to be recovered and stored on future data storage systems without requiring any changes to its contents simply to accomplish the act of medium migration, but it also needs to allow changes to its contents, in case updating is needed to data that already has been archived.

AXF addresses these needs through a combination of predefined eXtensible Markup Language (XML) schema fields, defined binary data structures that enable an AXF Object to carry any type of file within its File Payload, internal file system functionality, and key metadata enabling the spanning of AXF Objects across multiple physical media. The XML schema also enables essential information about an AXF Object and its contents to be read without having to process all the information within the AXF Object.

In addition to media interchange, AXF enables the interoperability of disparate systems through networks because it is structured as a streaming data set. Such interconnections enable seamless movement of AXF Objects from systems that create them, to systems that do not recognize the AXF protocol but store the AXF Object files nonetheless (perhaps in “cloud” storage), then to systems that are designed to recover data from AXF Objects.

Functionally, AXF acts like a file wrapper or a repository for all types of data without constraint. Unlike media-centric file formats such as MXF, which are similar in that they wrap essences, AXF can contain any number or types of files of any size encapsulated in an AXF Object. It is applicable across a much broader variety of file storage user groups than any media-specific file wrapper. Types of data can include media essence files, related metadata files, production files (such as word processing documents, hypertext documents, associated essence, applications, spreadsheets, and database copies), or any other type of data that users wish to store together. Unlike other file wrapper definitions, it is payload agnostic and does not require any special mappings or adaptations to accept the data an AXF Object carries.

AXF accommodates very large file sizes and quantities within AXF Objects. In the current version of this standard, 64-bit numbers are used to define the sizes of various parameters applicable to elements of AXF



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

Objects. 64-bit numbers can express values up to 18.44674 x $10^{18}$ (e.g., 18.44674 petabytes). Use of 64-bit numbers thus can define file sizes in bytes, numbers of files, numbers of media in a spanned set, and similar characteristics up to 18.44674 x $10^{18}$ of any particular element. If future requirements exceed the number spaces provided in this document, there is nothing fundamental that limits any particular parameter to expression using a 64-bit number. Future revisions of this standard could adopt larger number spaces (e.g., 96-bit, 128-bit, etc.) for those parameters requiring them. The net result is effectively unlimited storage capability within AXF Objects, in terms of file sizes, numbers of files in an AXF Object, number of AXF Objects on a medium, number of media in a spanned set, and the like.

AXF enables updating AXF Objects when additions of, modifications to, or deletions of files or information that they contain are needed. The functionality to modify AXF Objects is provided by linking “Supplemental” AXF Objects, written into an archive system at a later time to an original (“Anchor”) AXF Object. A Supplemental AXF Object updates contents of previous AXF Objects without requiring the original AXF Object itself to be modified. Since the original content of the AXF Object is retained in its original form, it is possible to restore either the original or the modified version whenever necessary. Additional Supplemental AXF Objects can be added in a chain, with restoration of the current or any earlier version possible at any time. When AXF Objects are refreshed by copying them to new media, it is possible to consolidate an Anchor Object and its Supplemental Object(s) into a single, new AXF Object. In doing so, it is possible to retain all of the constituent Objects of the Collected Set to which they belong, so that all earlier versions still can be reconstituted in the future.

AXF abstracts the storage of data from the applications that create AXF Objects and from the operating systems, file systems, drivers, and drives that store data on media. By this mechanism, any of the surrounding hardware and software components of systems can be replaced without affecting the data and its formatting within AXF Objects. A simplified view of where AXF fits into a basic stack is shown in Figure 1.

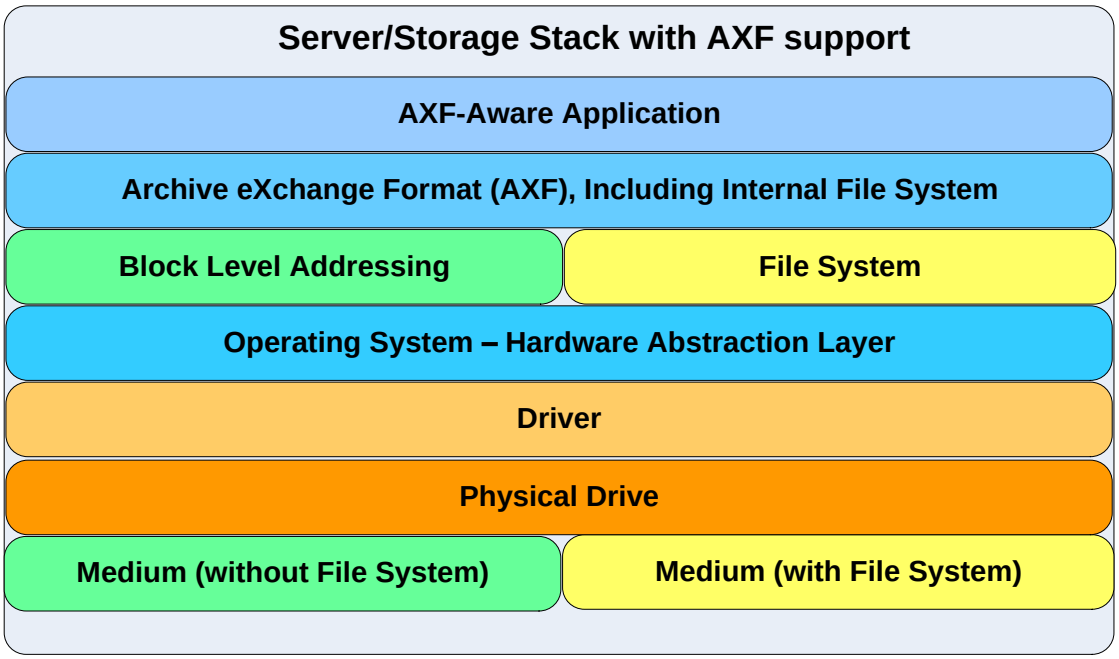


**Figure 1 – Hardware/Software Stack Incorporating AXF Writing to and Reading from Media**

AXF is designed so that each AXF Object comprises four main components, regardless of the technology that is used to store it. These components are:

Object Header – Each AXF Object begins with an Object Header, which contains descriptive XML metadata such as a unique identifier (UUID) for the AXF Object, information regarding its origin, its creation date, and a full index of all the files and folders contained in the Object, including file permissions and the like.



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

Generic Metadata Containers – Following an Object Header can be any number of optional Generic Metadata Containers. Such containers are self–contained, open metadata containers in which applications can place AXF-Object-specific metadata that is not part of the AXF Object File Payload. The metadata can be structured or unstructured, open or vendor-specific, binary, XML, plain text, or any other format.

AXF Object File Payload – Following any Generic Metadata Containers is the AXF Object File Payload. It contains the files encapsulated in the AXF Object. The File Payload consists of any number of triplets: File Data + File Padding + File Footer. File Padding ensures alignment of all AXF Object elements on the boundaries of Chunks into which each AXF Object is divided, thereby enabling addressing, by location within the AXF Object, by its internal file system. File Footer structures contain full information about the preceding file, along with a file-level checksum designed to be processed on-the-fly, with little or no overhead, during restore operations by an application. The information in File Footers enhances the resilience of AXF, as it can be used to recover File Payload data even if Object Header and Footer structures are missing or corrupt.

Object Footer – Completing an AXF Object is an Object Footer. It repeats the information contained in the Object Header and adds information captured during creation of the AXF Object, including per-file checksums, precise file sizes, and file positions within the AXF Object. The Object Footer is important to the interchange of an AXF Object because it allows efficient indexing by foreign systems when the media content is not previously known, thereby enabling media transport between systems that follow the AXF standard. It is one of the key structures that support the self-describing nature of AXF.

Other significant structures in the AXF protocol are AXF Medium Identifiers and AXF Object Indices. AXF Medium Identifiers are used on media to indicate formatting of the media according to the AXF protocol and to provide unique identification of the media. AXF Object Indices are optional compilations of the information in all Object Footers preceding each AXF Object Index on a medium, providing a single structure from which it is possible to obtain complete information on the contents of the preceding portion of a medium. When an AXF Object Index is the last structure on a medium, complete information about all AXF Objects stored on the medium can be obtained efficiently in one place.

AXF does not require a system to be fully compliant with this standard for it to be able to use and store AXF-generated AXF Objects. The initial adoption of AXF is anticipated to be in applications that create AXF Objects that then are stored on non-AXF-aware storage systems. Because the AXF Objects do not require a storage system to know that the AXF Objects are AXF-formatted, the AXF Objects will be viewed simply as files to be stored and retrieved. All that will be necessary to read AXF Objects will be the software and hardware needed to read the physical storage medium. As adoption grows, files can be moved into and out of AXF-aware systems as necessary, with the full range of features becoming available on systems that are AXF compliant. AXF-compliant applications will be able to read stored AXF Objects from any current operating system without unpacking entire AXF Objects to see critical metadata. Moreover, Archives, or AXF Objects within archives, also can span different types of media, allowing for flexibility within mixed-media archives and for AXF Objects to be identical, regardless of the media on which they are stored.

AXF offers resilience to data corruption and loss. AXF Object Indices, repeated identifier instances, and cryptographic hash checksums on both contents and AXF Objects allow for data corruption to be identified and mitigated. Even in catastrophic events, such as the loss of an external database containing records of the contents of an archive, the content database can be recreated by reading the archive and regenerating the archive-wide database from the records within the AXF Objects. This standard also enables the addition of more powerful data corruption recovery methods in future revisions by including provisions for incorporating forward error correction codes. AXF provides key features to identify and treat data corruption and loss across all types of storage formats. It allows big data and small to be processed in a consistent and standardized way.

Additional information on AXF can be found at OpenAXF.org



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

# 1 Scope

This standard is Part 1 of a series of documents that specify a general-purpose format for the storage and/or communication of information in bulk form. The format is named the Archive eXchange Format (AXF). The format described is intended both for interchange between systems and to serve as a native format within systems.

This standard identifies two major categories of data storage media and specifies the basic structures of data stored on those Storage Media Types. It specifies a number of structural elements for use in constructing the appropriate structures for use on each of the Storage Media Types. It defines the semantics of data contained within fields specified for use in the structural elements. The structural elements themselves are documents coded in the eXtensible Markup Language (XML), and this document defines an XML Schema Description (XSD) file for use in formulating the XML documents to be used for the structural elements of AXF Objects.

# 2 Conformance Notation

Normative text is text that describes elements of the design that are indispensable or text that contains the conformance language keywords: "shall," "should," or "may." Informative text is text that is potentially helpful to the user, but not indispensable, and that can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: The Introduction, any section explicitly labeled as "Informative," or individual paragraphs that start with "Note:".

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword “reserved” indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword “forbidden” indicates “reserved” and, in addition, indicates that the provision never will be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

Unless otherwise specified, the order of precedence of the types of normative information in this document shall be as follows: Normative prose shall be the authoritative definition; tables shall be next, followed by formal languages, then figures, and then any other language forms.

# 3 Normative References

The following standards contain provisions that, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

XML 1.0, World Wide Web Consortium (W3C) (2008, November) Extensible Markup Language 1.0 (Fifth Edition)

XML Namespaces 1.0, World Wide Web Consortium (W3C) (8 December 2009) Namespaces in XML 1.0 (Third Edition)

XML Schema Part 1 — World Wide Web Consortium (W3C) (28 October 2004), XML Schema Part 1: Structures (Second Edition)

XML Schema Part 2 — World Wide Web Consortium (W3C) (28 October 2004), XML Schema Part 2: Datatypes (Second Edition)

ISO/IEC 10646:2012, Information Technology — Universal Coded Character Set (UCS), Third Edition (1 June 2012)

ISO/IEC 19505-1:2012, Object Management Group (OMG) (August 2011), Unified Modeling Language: Infrastructure, v2.4.1

ISO/IEC 19505-2:2012, Object Management Group (OMG) (August 2011), Unified Modeling Language: Superstructure, v2.4.1

IETF RFC 1494, Equivalence Between 1988 X.400 and RFC-822 Message Bodies (August 1993)

IETF RFC 2231, MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations. (November 1997)

IETF RFC 3629, UTF-8, A Transformation Format of ISO 10646 (November 2003)

IETF RFC 3986, Uniform Resource Identifier (URI): Generic Syntax

IETF RFC 4122, A Universally Unique IDentifier (UUID) URN Namespace. (July 2005)

IETF RFC 6657, Update to MIME Regarding “Charset” Parameter Handling in Textual Media Types (July 2012)

IETF RFC 6838, Media Type Specifications and Registration Procedures

ISO/IEC 1001:2012(E) (VOL1), Information Technology — File Structure and Labeling of Magnetic Tapes for Information Interchange (2012-08-01)

ISO/IEC 13239:2012 (CRC), Information Technology — Telecommunications and Information Exchange Between Systems -- High-Level Data Link Control (HDLC) Procedures

FIPS PUB 180-4, Federal Information Processing Standards Publication, Secure Hash Standard *SHS), Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD, March 2012

UTC, Coordinated Universal Time, ITU-R Recommendation TF 460-6, Standard-Frequency and Time-Signal Emissions, International Telecommunications Union, Radio Sector, Geneva, Switzerland, February 2002

## 4 Definitions

The following terms shall have the definitions specified for them below, both throughout this document and in related Parts of this SMPTE standard.



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

### 4.1 Anchor Object

The first AXF Object in a Collected Set of AXF Objects, the **SetSequence** value of which is 1. It is the AXF Object from which all Subsequent Objects in the Collected Set differentiate the contents of the Product Object.

### 4.2 Archive Object

An AXF Object.

### 4.3 Attribute

An XML markup construct consisting of a name/value pair that describes an element.

### 4.4 AXF

Archive eXchange Format.

### 4.5 AXF-Aware

Capable of reading, interpreting, and/or writing AXF Objects and other AXF data structures.

### 4.6 AXF Media

Multiple media prepared to carry AXF Objects.

### 4.7 AXF Medium

A medium prepared to carry AXF Objects.

### 4.8 AXF Medium Identifier

A data structure at the start or root of a medium that indicates that it has been prepared for storage of AXF Objects and that provides information about the identification, formatting, and history of the medium.

### 4.9 AXF Object

A group of data elements that are stored as a unit and that are contained within a single data structure.

### 4.10 AXF Object Index

A data structure, stored on a medium, that is a collection of the complete information about all previously-stored AXF Objects on that medium and the contents of those AXF Objects, as well as the locations of the content files stored within those AXF Objects.

### 4.11 Base Name

The fundamental name of a file; the portion of a file name prior to any file name extension that identifies the type of file or structure.

### 4.12 Big Endian

The order in which the bytes of a multi-byte number are transmitted with the most significant byte first.

### 4.13 Binary Structure Container

A data structure that is formed using easily identifiable binary data patterns of UTF-8 characters and that carries a payload of a specified type for the purpose of providing structural elements for AXF Objects and for the files contained within them.

### 4.14 Binary Structure Container Payload

The data contained within the **Payload** field of a Binary Structure Container.

### 4.15 Block



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

A section of data of uniform size (measured in bytes) as stored on a physical medium that is independently addressable for write and read operations, and which is defined as a “block” on certain Storage Media Types and as a “sector,” a “page” or some other term on other Storage Media Types.

### 4.16 Block-Based

Media that enable direct access to Blocks of data stored thereon and that do not incorporate or depend upon File Systems.

### 4.17 Block Position

The location on a medium of a block of data, measured as the number of blocks from the beginning of the medium, as counted by the drive on which the medium is carried.

### 4.18 Block Size

A parameter that defines the size of Blocks contained on a medium. Some Storage Media Types can accommodate varying block sizes across the storage space of a medium.

### 4.19 Byte Order Mark

A small sequence of bytes carried within a file from which it is possible to determine the byte ordering (Big Endian or Little Endian) of the contents of the file. (Abbreviated BOM.)

### 4.20 Chunk

A section of data of uniform size (measured in bytes) within an AXF Object.

### 4.21 Chunk Size

A parameter that defines the size of all of the Chunks within an AXF Object.

### 4.22 Collected Set

A group of AXF Objects, consisting of an Anchor Object and one or more Subsequent Objects, that, when compiled, produce a Product Object differing from the Anchor Object as a result of additions, replacements, or deletions of files.

### 4.23 Content File

A File of any type that is stored as part of the File Payload within an AXF Object. Also a Payload File.

### 4.24 Data Element

A file or data structure that forms part of the overall data stored within an AXF Object or on a medium.

### 4.25 Element

An XML logical document component bounded by matching start and end tags and including any content between the tags plus any associated attributes.

### 4.26 Embedded File System

A light weight File System having its database contained within an AXF Object.

### 4.27 Endianness

The ordering of bytes within a multi-byte number. Byte orders typically are Little Endian or Big Endian, although other, less frequently used byte orders exist.

### 4.28 File Mark

A data structure written or read by a Linear Media drive to provide separation between data elements stored on a medium using the media technology for which the drive is designed.



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

### 4.29 File Footer

A Binary Structure Container that carries a payload with information to identify the file that it follows and to provide detailed information about that file.

### 4.30 File Padding

Data inserted at the end of a file to completely fill the last Chunk occupied by the file.

### 4.31 File Payload

The portion of an AXF Object that carries the Payload Files and their associated File Footer Binary Structure Containers.

### 4.32 File-System-Based

Media that incorporate and depend upon File Systems and provide no direct access to Blocks of data.

### 4.33 File System

A type of data store that can be used to store, retrieve, and update a set of files.

### 4.34 File Tree

A hierarchical structure of folders and files, expressed in XML, that starts at a root point and establishes relationships between folders and the folders and files that they contain.

### 4.35 Fixity

Fixity is the property of being constant, steady, and stable. Fixity checking is the process of verifying that a digital object has not been altered or corrupted. In practice, this is most often accomplished by computing and comparing cryptographic hash checksums.

### 4.36 Fragment

A data structure that contains a portion of an AXF Object that is linked to preceding or following Fragments, or both, through shared UUID values.

### 4.37 Generic Metadata Container

A Binary Structure Container the payload of which is metadata.

### 4.38 Linear Media

Media requiring writing and reading sequentially along the length of the medium (typically tape).

### 4.39 Little Endian

The order in which the bytes of a multi-byte number are transmitted with the least significant byte first. This might or might not match the order in which numbers are normally stored in memory for a particular processor.

### 4.40 Media

Plural of medium.

### 4.41 Media Class

The general category of Media, that is, Block-Based Media (i.e., both Linear and Non-Linear Media without File System support) and File-System-Based Media (i.e., both Linear and Non-Linear Media with File System support).

### 4.42 Media Family

A group of related Storage Media Types sharing a common set of media technology but allowing for different implementation characteristics, such as with respect to storage density.

### 4.43 Media Technology



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

A set of techniques used to store data on a particular form of media, including the choice of medium itself and the methods for such functions as recording bits on a medium, error correction, indicating a separation between portions of data stored, and the like.

### 4.44 Media Type

A registered name that describes the format of the content of a file.

### 4.45 Medium

A physical carrier on which data is stored, which comprises an integer number of Blocks and which can be either Block-Based or File-System-Based. (Equivalent to **Storage Medium**.)

### 4.46 Non-Linear Media

Media that can be written, read, and accessed in a non-sequential manner, enabling data to be accessed randomly on such a medium.

### 4.47 Object Footer

A data structure at the end of an AXF Object that provides complete information about the contents of the AXF Object, the positions within the AXF Object of the files that it contains, and other information about the AXF Object and its contents.

### 4.48 Object Fragment

The portion of an AXF Object stored on a single medium when the AXF Object is spanned across multiple media.

### 4.49 Object Header

A data structure at the start of an AXF Object that provides some or all of the information about the contents of the AXF Object, the positions within the AXF Object of the files that it contains, and other information related to the AXF Object and its contents.

### 4.50 Padding

Data having a value of 0x00 used to fill data space to maintain the format of a data structure.

### 4.51 Padding Chunk

A Chunk completely filled with bytes having a value of 0x00.

### 4.52 Payload File

A file contained within an AXF Object following the Payload Start Binary Structure Container and preceding the Payload Stop Binary Structure Container, if present.

### 4.53 Product Object

The compilation of files, directory structure, and other information that results from the combination of the Anchor Object and all subsequent AXF Objects in a Collected Set.

### 4.54 Spanned Set

A group of media containing Fragments of a single AXF Object linked together by a shared UUID identifying the AXF Object and by shared UUID values matching the Span Points of the AXF Object.

### 4.55 Spanning

A process for fragmenting a single AXF Object and storing its fragments on multiple media.

### 4.56 Span Point



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

The location within an AXF Object at which the AXF Object contents are fragmented for storage in separate Fragments.

### 4.57 Storage Media Type

A specific combination of media technology with parameters for the various data storage characteristics and data processing methods.

### 4.58 Storage Medium

A Medium used to store data. (Equivalent to **Medium**.)

### 4.59 Subsequent Object

An AXF Object in a Collected Set of AXF Objects other than the Anchor Object, the **SetSequence** value of which is greater than 1. It is an AXF Object that differentiates the contents of the Product Object from the Anchor Object of the Collected Set by indicating additions, replacements, or deletions of files and that carries any files necessary to effectuate the additions or replacements of files that it indicates.

### 4.60 Symbolic Link

A pointer contained within a directory structure to a file or folder that may be internal or external to the file hierarchy described by the directory structure. Symbolic Links are used with certain types of File Systems, while other types of File Systems use “shortcut” files referenced by the directory structure to perform the pointer function.

### 4.61 Symlink

Symbolic Link

### 4.62 URI

Uniform Resource Identifier – A compact sequence of characters that identifies an abstract or physical resource.

### 4.63 UUID

Universally Unique Identifier – An identifier that is unique for all practical purposes and is created according to an algorithm specified to lead to different values for each such identifier generated.

### 4.64 VOL1

A label applied at the beginning of data tape media according to the provisions of ISO/IEC 1001 “Information technology – File structure and labeling of magnetic tapes for information interchange”

## 5 Storage Media Types

Media that carry AXF Objects in conformance with this standard fall into one of two fundamental categories: Block-Based and File-System-Based. The formatting of each of these Storage Media Types is according to specific arrangements described generally in this section and in greater detail below in other sections. As also described below, AXF Objects stored on the respective Storage Media Types do not differ in their basic formats, but certain indexes and pointers do differ somewhat in the values they carry depending upon the Storage Media Types upon which their AXF Objects are stored. Mechanisms are provided herein and in other Parts of this document suite for the storage, transfer and processing of AXF Objects between Storage Media Types in a transparent manner and using minimal processing. Block-Based and File-System-Based Media are either Linear or Non-Linear.

Linear Media are media that only can be written and read sequentially, from end to end. In the drives that write and read them, Linear Storage Media Types include mechanisms for writing and reading File Marks or similar identifiers of points of separation between sequential sections of data stored on the media. Linear Storage Media Types include in the drives that write and read them at least the capability to search for File Marks, to count File Marks over the length of a medium, and to advance in either direction by a specified number of File



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

Marks. Typical Linear Storage Media Types are based upon the use of electromagnetic tape for the storage of data. Linear Media carry File Systems (e.g., LTFS) or do not carry File Systems.

Non-Linear Media do not use File Marks and instead are addressed using logical block addresses that point to specific locations where data are stored. Non-Linear Media provide the ability to move directly from any place on the medium to any other place on the medium. Typical Non-Linear Storage Media Types are based upon the use of magnetic or optical disks or flash memory devices for the storage of data. Non-Linear Media generally are expected to be File-System-Based, but the AXF protocol will support Block-Based variants as well.

Within AXF Objects, pointers are provided in several data structures to enable retrieval of files stored within the AXF Objects. The pointers indicate the locations of files within AXF Objects using either relative or absolute addresses based on Chunk counts within the AXF Objects. Relative addressing is used when pointing to all structures contained within an AXF Object. Absolute addressing is used when pointing to the starting location of an AXF Object. Combining relative and absolute addresses enables determining absolute locations of all structures within an AXF Object on the medium on which it is stored.

## 5.1 Media with File Systems

In the AXF context, File-System-Based Media shall be media that store the necessary indexes for the files that they contain and operate with file system software that manages those indexes and the data storage and recovery processes on the media. File systems provide services that include at least indexing mechanisms for files and their storage locations on media, management of the locations of files when writing to the media, and recovery of files with their data presented in the order in which they were stored.

Example: File Systems that can be used to store AXF Objects include the File Allocation Table (FAT), the New Technology File System (NTFS), Universal Disk Format (UDF), Distributed File System (DFS), Network File System (NFS), Hierarchical File System Plus (HFS+), Linear Tape File System (LTFS), and many others. On any File System, AXF Objects are stored as files, providing a constant AXF Object format, thus isolating the contents of the AXF Objects from the File Systems.

On File-System-Based Media, the locations of AXF Objects are managed by the File Systems; thus, it is unnecessary to include absolute addresses of the starting locations of AXF Objects within those AXF Objects. Consequently, in the storage of AXF Objects on Media having File Systems, the fields that otherwise would carry the absolute addresses of the AXF Object starting locations should be set to a value of -1. When transferring AXF Objects between Storage Media Types, the values in such fields normally will have to be modified to meet this requirement. Such transfers are discussed in more detail in Section 7.4.

## 5.2 Media without File Systems

Media without File Systems do not provide File-Based references to the content they carry and are referred to as Block-Based Media. They are based on storage of data in blocks of specific sizes, but many of these Storage Media Types lack means for determining the numbering of blocks on a medium. These media either can be Linear or Non-Linear in nature. On Linear Media, File Marks or similar mechanisms typically are used to enable systems writing to them to place specific reference points along media. Writing of File Marks can be a time-consuming process, however, on many Storage Media Types without File Systems. For this reason, the AXF protocol eliminates use of File Marks in all but two instances at the start of each linear AXF Medium.

The fundamental requirement for inclusion of File Systems on Linear Media is the ability of the media to be partitioned. Versions of Linear Storage Media Types that do have the capability for partitioning of the media nevertheless can be used without File Systems when applications can benefit from such utilization. Early generations of many Linear tape families are incapable of being partitioned and consequently cannot support File Systems, while later generations of those families of Linear tape types are capable of such support.

On Block-Based AXF Media, the locations of AXF Objects are not managed by file systems. Consequently, AXF Objects include fields for the absolute addresses of Object Headers on such AXF Media. The absolute addressing used requires the system writing to or reading from a medium to keep track of the block count along



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

the medium as it writes to it or reads from it. In the storage of AXF Objects on Block-Based Media, the fields carrying the absolute addresses of the AXF Object starting locations are set to the positions of the blocks containing the first bytes of the Object Header binary structure containers, the locations of which are being identified. When transferring AXF Objects between Storage Media Types, the values in the fields containing the Object Header absolute addresses should be modified to point to the correct locations. Such transfers are discussed in more detail in Section 7.4.

### 5.3 File Marks

File Marks are data patterns that can be placed on magnetic Linear Media (e.g., data tape) by drives, upon instructions to do so from controlling applications. The data patterns are predetermined for each particular type of medium and are intended to be easily recognized by drive subsystems when reading or scanning a medium at high speed.

The AXF protocol avoids the use of File Marks to the extent possible to improve the efficiency of tape operations. Only two are used: One following the VOL1 identifier and the other following the AXF Medium Identifier on a Linear Block-Based AXF Medium. See Section 5.4 for the designation of Storage Media Types on which File Marks shall be applied.

### 5.4 Relationships Between AXF Structures and Storage Media Types

Fundamentally the same AXF Object data structures are stored on all Storage Media Types, but certain media formatting and identification data structures shall be applied to particular Storage Media Types according to the relationships shown in Table 1. There are four categories of media to which the formatting or identification structures are applied: Block-Based and File-System-Based Linear Media and Block-Based and File-System-Based Non-Linear Media, as represented by the four columns in the right-hand portion of the table. The various structures listed in the leftmost column are described in the sections of this standard enumerated in the next-to-leftmost column of the table.

In Table 1, structures are indicated as being required, not applicable, or optional. Structures indicated as “Required” shall be employed on media in the specified media categories, except that File Marks shall be required only in instances in which the hardware media drives support them. Structures indicated as “Not Applicable” shall not be employed on media in the specified media categories. Structures indicated as “Optional” may be employed on media in the specified media categories.

Note: The VOL1 and File Mark structures are shown in Table 1 as “Not Applicable” in certain instances. They are not applied according to the AXF protocol, but they nevertheless can be present on a medium carrying AXF structures if applied by an underlying file system or other media management protocol.



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

**Table 1 — Relationships between AXF Structures and Storage Media Types**

| AXF Structure | | Linear Media | | Non-Linear Media | |
|---|---|---|---|---|---|
| | Section | Block-Based | File-System-Based | Block-Based | File-System-Based |
| **VOL1 Label** | 6.4.2.1 | √+ | – | √ | – |
| **File Mark** | 5.3 | √* | – | – | – |
| **AXF Medium Identifier** | 0 | √+ | O | √ | O |
| **AXF Object** | 6.4.3.1 | √ | √ | √ | √ |
| **AXF Object Index** | 6.4.2.3 | O | O | O | O |
| **Example** | | Block-Based Data Tape | LTFS-Formatted Data Tape | Block-Based Disk or Flash Drive | NTFS Disk or FAT32 Flash Drive |

**√ = Required, – = Not Applicable, O = Optional, + = Followed by File Mark, * = When Supported**

# 6 Archive eXchange Format (AXF) Structure

The Archive eXchange Format (AXF) describes a protocol that includes an Archive Object (hereinafter "AXF Object") data structure in which related content and information about that content are encapsulated and stored when interchanged between storage systems and in which data can be stored internally within systems that make use of the protocol. AXF also describes a storage arrangement for AXF Objects and other data that is applied to media used for the carriage of AXF Objects. Associated with the AXF Objects and storage arrangement are a number of identifiers, metadata, and data structures that are used to construct specific instances of AXF Objects stored on AXF Media. Included among the identifiers, metadata, and data structures are those that permit the spanning of single AXF Objects across multiple AXF Media and others that allow for indexing of AXF Media by systems.

All of the AXF Object data structures and File Payload data are Chunk-aligned internally. This enhances resiliency and assists in the recoverability of packages in the event of corruption of the embedded indexing structures. The Chunk size may be one byte in length; however, this practice normally should be avoided, as it increases index and recovery complexity. Nevertheless, it can be applied when it is necessary for systems to avoid certain restrictions of Chunk-aligned structures. As a result of the internal Chunk alignment and indexing structures, AXF Objects are said to include Embedded File Systems that assist in the abstraction of the underlying storage technology, medium, File System, and operating system.

AXF Objects are constructed so that they can be transferred sequentially when necessary (e.g., to a Linear medium or from an archive system to a remote storage sub-system over a data circuit or communications link). Sequential transfers allow AXF Objects to be streamed to media in write-once fashion, supporting, for example, linear and write-once media as well as processing during transfers across data circuits prior to completion of the transfers of complete AXF Objects.

## 6.1 Form of Data Expression

The Payloads carried within Binary Structure Containers (see Section 6.4.1.2) to form all of the AXF data structures contained on AXF Media and/or within AXF Objects formatted according to this standard shall be expressed in the form of documents in the eXtensible Markup Language (XML), as specified in XML 1.0 and XML Namespaces 1.0. XML documents created according to this standard shall be generated using the XML Schema Description (XSD) files described in Section 10. The XML Schema Description (XSD) method is specified in XML Schema Part 1, and XML Schema Part 2.



© ISO/IEC 2017 – All rights reserved

Documents Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

## 6.2 Byte Order

The byte order of the Binary Structure Container (see Section 6.4.1.2) other than the binary Structure Container Payload and those fields that are of the UTF-8 data type shall be little endian. The byte order of a Binary Structure Container Payload shall be indicated by the **Payload Format** field as specified in Table 2. The byte order of a Payload File (see Section 6.4.3.7) may be signaled in the associated File metadata (in the XML File data structure); it may be signaled by a Byte Order Mark contained within the Payload File; or it may be unknown. When a Byte Order Mark is known to be contained within a Payload File, that information also may be signaled in the associated File metadata. In cases in which the byte order or the presence of a Byte Order Mark is unknown, no entry with respect to the byte order shall be made in the File metadata.

## 6.3 General AXF Concepts

AXF deals with the limitations of legacy storage formats through application of multiple solutions, employed in parallel with one another: AXF employs a variable length FileTree structure that can list an unlimited number of files. The sizes of files stored within an AXF Object are expressed as XML integers, and XML integers have no fundamental limit in their size. AXF establishes hierarchical relationships between files and folders through inclusion of a FileTree structure and can include an unlimited number of entries on an unlimited number of levels within a hierarchy. AXF expresses file names as XML strings of unlimited size in UTF-8, and UTF-8 can express all Unicode characters in single bytes or multiple-byte groups. AXF provides a mechanism for exceeding the physical limits of the media on which AXF Objects are stored through provisions for Spanning to an unlimited number of media, thereby avoiding any potential limitation on the aggregated size of an AXF Object. AXF provides a mechanism for altering previously-stored AXF Objects, even on write-once media, through use of Collected Sets, which enable additions to, deletions from, and replacements of files in the File Payloads of AXF Objects, without the need to reconstruct or rewrite any previously written AXF Objects.

These concepts are discussed in general terms in this Section (6.3) and in detailed terms in Section 6.4.

### 6.3.1 Embedded File System / File Tree / File Payload

The combination of a File Tree structure and the Payload Files of an AXF Object form an Embedded File System, wholly contained within the AXF Object. AXF establishes relationships between the files in AXF Object File Payloads and folders in which they are contained, through a File Tree structure expressed in XML. Folders in AXF Objects exist solely as listed entries within the File Tree XML structure. The File Tree structure of an AXF Object lists all of the folders and the files within the folders, numbering them in such a way that their relationships can be determined by reading the FileTree data type. Multiple copies of the FileTree data type are stored within an AXF Object and, optionally, elsewhere on the medium on which the AXF Object is stored, to provide redundancy and to support resilience of Payload File recovery. Within an AXF Object, the FileTree data type is stored in the Object Header and Object Footer; external to the AXF Object, the FileTree data type can be stored in one or more Object Indices stored on the medium on which the AXF Object resides, as appropriate for the medium. In the File Payload, files are followed by File Footer XML structures that both demark the separation between one file and the next and provide all of the detailed information about the files that they follow.

### 6.3.2 Spanning

When an AXF Object contains too much content to fit in the remaining space on a medium, it can be divided into two or more fragments that can be stored on multiple media. The process is called Spanning and results in a Spanned Set of AXF Objects, each Object containing one fragment of the overall AXF Object. The Spanned Set is structured in such a way that the overall AXF Object can be reconstructed easily from its fragments. The reconstruction is aided by a series of linkages between the fragments using specialized headers and footers that carry identifiers for their respective fragments and the links between them.



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

### 6.3.3 Collected Sets

When it is desired to modify an AXF Object, through addition, substitution, or deletion of Payload Files, it is possible to do so through creation of a Subsequent Object that carries instructions on the modifications to be made and any additional Payload Files that can be necessary when carrying out the instructions. In such a case, the original AXF Object becomes known as an Anchor Object, and one or more Subsequent Objects can be added to what becomes a Collected Set of AXF Objects. Through compilation of the AXF Objects in a Collected Set in sequence, it is possible to produce a Product Object at any level of revision of the Collected Set that is needed. This mechanism permits modification of AXF Objects written on media that are not rewritable and also preserves all versions of the Collected Sets without loss of any content.

## 6.4 AXF Data Structures

Data structures generally utilized within the AXF protocol, and described in the following sections, are divided into these categories:

- General Data Structures (see Section 6.4.1)
    - Chunk (see Section 6.4.1.1)
    - Binary Structure Container (see Section 6.4.1.2)
- Media Data Structures (see Section 6.4.2)
    - VOL1 Label (see Section 6.4.2.1)
    - AXF Medium Identifier (see Section 6.4.2.2)
    - AXF Object Index (see Section 6.4.2.3)
- AXF Object Data Structures (see Section 6.4.3)
    - File Tree (see Section 6.4.3.3)
    - Object Header (see Section 6.4.3.4)
    - Object Footer (see Section 6.4.3.10)
    - Generic Metadata Container (see Section 6.4.3.5)
    - File Payload Start (see Section 6.4.3.6)
    - File Payload (see Section 6.4.3.7)
    - File Footer (see Section 6.4.3.8)
    - File Payload Stop (see Section 6.4.3.9)
    - Fragment Footer (see Section 6.4.3.11)
    - Fragment Header (see Section 6.4.3.12)



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

### 6.4.1 General Structures

General structures are part of the AXF protocol that transcend AXF Media- and AXF Object-specific structures, as defined in subsequent sections of this standard. These general structures have a wide applicability and are intended to ensure uniformity, regardless of the particulars of the specific implementation, Storage Media Type, or AXF Object to which they apply.

#### 6.4.1.1 Chunks

Requirements for the sizes and alignment of Chunks on media are different for Block-Based and File-Based systems.

##### 6.4.1.1.1 Block-Based Systems

In Block-Based systems, Chunk boundaries shall be aligned to Block boundaries. The first byte of an AXF Object shall align with the first byte of a Block. All AXF Objects on a given Medium shall have the same Chunk size, matching the Block size of the Medium.

Note: When reading or writing AXF Objects on Block-Based systems, chunk or block realignment only can be performed by AXF–aware applications.

##### 6.4.1.1.2 File-System-Based Systems

In File-System-Based systems, Chunk Boundaries need not be aligned with Media internal structures. Different AXF Objects on the same medium may have different Chunk sizes. Applications that read or write AXF Objects on File-System-Based systems need not be AXF aware with respect to AXF Chunk or Block realignment.

Note: On File-System-Based Media, it is possible for an AXF-aware application to tune Chunk sizes of AXF Objects for best efficiency based upon minimum, maximum, and average file sizes.

#### 6.4.1.2 Binary Structure Container

A Binary Structure Container is a mechanism, expressed in binary form, for encapsulating payloads for the multiple data structures that comprise the overall AXF protocol. It is used to create both AXF Media data structures and AXF Object data structures, providing a universal approach to creating, storing, interpreting, and parsing them. A Binary Structure Container can be seen as a well-defined and predictable encapsulation, or wrapper, for each structure (payload) that can be contained within it. Binary Structure Containers are applied to all AXF Media and AXF Object data structures except Payload File data itself.

Binary Structure Containers are used to avoid the complexities associated with raw storage of XML and other data types on various types of media and to ensure bit fixity and the ability to validate these structures during processing. Binary Structure Containers add resiliency to the overall AXF protocol, allowing applications to detect, and potentially recover, key structures independent of other index and metadata structures that are part of an AXF Object or AXF Medium. A Binary Structure Container is a generic container that can contain any type of data in its Payload. A Binary Structure Container also serves as the foundation for a Generic Metadata Container, as described in Section 6.4.3.5.

Binary Structure Containers shall be structured as defined in Table 2 and illustrated in Figure 2. Unless otherwise stated, all values in Table 2 shall be mandatory.



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

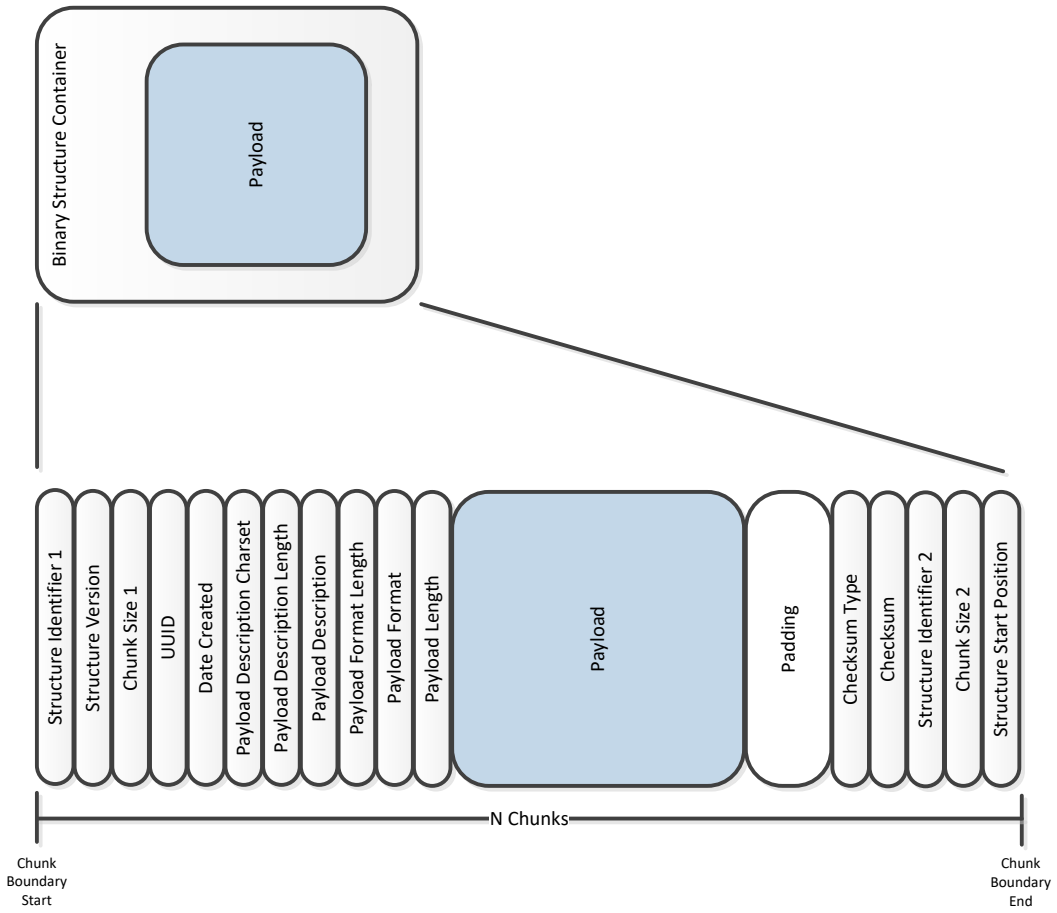


**Figure 2 – Binary Structure Container Overview**



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

**Table 2 – Structural Elements of Binary Structure Containers**

| Field Name | Byte Offset | Length (Bytes) | Data Type | Example | Description and Requirements |
|---|---|---|---|---|---|
| Structure Identifier 1 | 0 | 32 | UTF-8 | AXF_OBJECT_HEADER | The **Structure Identifier 1** value shall be a string of UTF-8 characters identifying the type of data structure within the **Payload** of the structure. Specified character strings shall be used by applications to identify contained data structures. The string shall be padded with UTF-8 NUL characters to the field boundary. UTF-8 NUL characters shall not be used within structure identifier values. A duplicate of this information is contained in the **Structure Identifier 2** field.<br><br>The first bit of each **Structure Identifier 1** shall coincide with the first bit of the Chunk in which it is contained.<br><br>NOTE: The following values identify the AXF data structures defined in this standard:<br><br>AXF_OBJECT_HEADER<br>AXF_OBJECT_FOOTER<br>AXF_OBJECT_METADATA<br>AXF_OBJECT_FILE_PAYLOAD_START<br>AXF_OBJECT_FILE_PAYLOAD_STOP<br>AXF_OBJECT_INDEX<br>AXF_OBJECT_FRAGMENT_HEADER<br>AXF_OBJECT_FRAGMENT_FOOTER<br>AXF_MEDIUM_IDENTIFIER<br>AXF_FILE_FOOTER |
| Structure Version | 32 | 4 | Unsigned 32-bit integer | 1 | The **Structure Version** shall be the version identifier for the container. This particular version information is specific only to Binary Structure Containers. All Binary Structure Containers within an AXF Object shall be of the same **Structure Version**. Binary Structure Containers constructed according to this version of this standard shall have a **Structure Version** value of 1). |



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

| Field Name | Byte Offset | Length (Bytes) | Data Type | Example | Description and Requirements |
|---|---|---|---|---|---|
| Chunk Size 1 | 36 | 8 | Unsigned 64-bit integer | 100000h | The **Chunk Size 1** shall indicate in bytes the Chunk size used for the current AXF Object. The same value is duplicated in the **Chunk Size 2** field.<br><br>Example: A value of 100000h indicates all Chunks within the associated AXF Object have a size of 1,048,576 (decimal) bytes. |
| UUID | 44 | 16 | Unsigned 128-bit integer | 90997230098211E2892E0800200C9A66(h) | The **UUID** value shall be equal to the **AXF Object UUID**, as specified in the associated XML payload carried within the Binary Structure Container, when the Binary Structure Container applies to AXF Object-specific structures such as Object Header, Object Footer, File Footer, etc. All Binary Structure Containers associated with the same AXF Object shall contain the same **AXF Object UUID** value.<br><br>The **UUID** value shall be equal to the **AXF Medium UUID**, as specified in the associated XML payload carried within the Binary Structure Container, when the Binary Structure Container applies to AXF Media-specific structures, i.e., AXF Medium Identifiers and AXF Object Indices. All Binary Structure Containers associated with the same AXF Medium shall contain the same **AXF Medium UUID** value. |
| Date Created | 60 | 8 | Signed 64-bit integer | 1348846975 representing "Fri Sep 28 11:42:55 EDT 2012" | The **Date Created** value shall indicate the time at which the Binary Structure Container in which it is found was created. Its value shall be the elapsed time, since 00:00:00 UTC on January 1, 1970, expressed as a number of seconds. |
| Payload Description Encoding Form | 68 | 40 | UTF-8 | UTF-8 | The **Payload Description Encoding Form** field shall provide the designation of the encoding form used in the **Payload Description** field to encode the scalar values of the characters contained in the payload description. Allowed values shall be: UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF-32, UTF-32LE, and UTF-32BE. The encoding form shall be as defined in ISO/IEC 10646, as shall be the scalar values associated with the characters and ideographs. |



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

| Field Name | Byte Offset | Length (Bytes) | Data Type | Example | Description and Requirements |
|---|---|---|---|---|---|
| Payload Description Length | 108 | 2 | Unsigned 16-bit integer | 128 | The **Payload Description Length** field shall indicate the length of the **Payload Description** field expressed in bytes. |
| Payload Description | 110 | Variable length, defined by the **Payload Description Length** field | Binary data encoded as specified in **Payload Description Encoding Form** | Migration Metadata, Proxy, Vendor Metadata, JPEG Cover Art, etc. | The **Payload Description** field shall contain user-provided information about the contents of the **Payload** field. This field can be left blank, but it can be useful when a user includes Generic Metadata payloads and wishes to help classify and identify such packages to systems into which AXF Objects are recovered. The examples given for this field apply to such Generic Metadata applications.<br><br>The encoding of the scalar values representing characters and ideographs shall be as specified in the **Payload Description Encoding Form** field. The correspondence between scalar values and the characters and ideographs that they represent shall be as defined in ISO/IEC 10646. |
| Payload Format Length | 110 + Payload Description Length | 2 | Unsigned 16-bit integer | 128 | The **Payload Format Length** field shall indicate the length of the **Payload Format** field expressed in bytes.<br>When the **Structure Identifier** value is AXF_OBJECT_FILE_PAYLOAD_START or AXF_OBJECT_FILE_PAYLOAD_STOP, the **Payload Format Length** shall have a value of zero. |



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

| Field Name | Byte Offset | Length (Bytes) | Data Type | Example | Description and Requirements |
|---|---|---|---|---|---|
| Payload Format | 112 + Payload Description Length | Variable length, defined by the **Payload Format Length** field | UTF-8 | application/xml | The **Payload Format** field shall indicate the format of the payload contained within the **Payload** field. The purpose of the **Payload Format** value is to assist in processing of the Binary Structure Container's payload.<br><br>The **Payload Format** field shall carry a comma (",")-separated list of one or more Media Types, as defined by RFC 6838, to specify a nested set of containers carrying the payload content. If only one Media Type is specified, it shall be that of the payload data. If more than one Media Type is specified, the Media Types shall specify the format of the payload containers, with the first Media Type specifying the outermost container and with each successive Media Type describing the format of that container's contents. The last Media Type in the list shall indicate the Media Type of the payload data.<br><br>The Media Type values used to define the **Payload Format** often include an implicit or explicit byte ordering specification, which, when present, shall specify the byte ordering of the payload contained within the **Payload** field. To the extent possible, Media Types that indicate byte ordering of the associated payloads should be used. If a Media Type value that indicates the payload format and the byte ordering of the payload is not available for a particular payload type or the byte ordering is unknown, then the Media Type value "application/octet stream," according to RFC 1494, should be used.<br><br>When the **Structure Identifier** value is AXF_OBJECT_FILE_PAYLOAD_START or AXF_OBJECT_FILE_PAYLOAD_STOP, the **Payload Format** field shall be omitted (i.e., have a length of zero). |



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

| Field Name | Byte Offset | Length (Bytes) | Data Type | Example | Description and Requirements |
|---|---|---|---|---|---|
| Payload Length | 112 + Payload Description Length + Payload Format Length | 8 | Unsigned 64-bit integer | 1a2b3c4d5e6f7a8b(h) | The **Payload Length** field shall indicate the size of the (possibly compressed) **Payload** expressed in bytes.<br><br>When the **Structure Identifier** value is AXF_OBJECT_FILE_PAYLOAD_START or AXF_OBJECT_PAYLOAD_STOP, the **Payload Length** field shall have a value of zero. |
| Payload | 120 + Payload Description Length + Payload Format Length | Variable Length Defined by the **Payload Length** field | As indicated by **Payload Format** | N/A | The **Payload** field shall contain the payload encapsulated by the Binary Structure Container.<br><br>When the **Structure Identifier** value is AXF_OBJECT_FILE_PAYLOAD_START or AXF_OBJECT_PAYLOAD_STOP, the **Payload** shall be empty and may be ignored by receiving applications. |
| Padding | 120 + Payload Description Length + Payload Format Length + Payload Length | Chunk Size – ((696 + Payload Description Length + Payload Format Length + Payload Length) Mod Chunk Size) | UTF-8 NUL characters | 0x00 | Padding added to align the entire data structure on Chunk-size boundaries defined for the particular AXF Object shall consist of 0x00 values.<br><br>Padding shall extend the Binary Structure Container by the minimum amount necessary to cause the last byte of the final field of the Binary Structure Container to appear in the byte position immediately preceding a Chunk boundary.<br><br>When the **Structure Identifier** value is AXF_OBJECT_FILE_PAYLOAD_START or AXF_OBJECT_FILE_PAYLOAD_STOP, the **Payload** will be empty, and Padding shall be applied to correctly position the last byte of the final field of the Binary Structure Container.<br><br>The byte length of the Padding is referred to as **Padding Length** and is used for Byte Offset Calculations of subsequent fields in the Binary Structure Container. |



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

| Field Name | Byte Offset | Length (Bytes) | Data Type | Example | Description and Requirements |
|---|---|---|---|---|---|
| Checksum Type | 120 +<br>Payload Description Length +<br>Payload Format Length +<br>Payload Length +<br>Padding Length | 16 | UTF-8 | SHA-1 | The **Checksum Type** field shall define the checksum algorithm used for structure verification. The **Checksum Type** value shall be drawn from the checksum algorithms specified in the list of examples in Section 10.21.1.3.<br><br>The following values shall be permitted:<br><br>CRC64<br>MD5<br>SHA-1<br>SHA-224<br>SHA-256<br>SHA-384<br>SHA-512 |
| Checksum | 136 +<br>Payload Description Length +<br>Payload Format Length +<br>Payload Length +<br>Padding Length | 512 | Unsigned 4096-bit integer | 12fe456d89c76562<br>5df9df9812fe456d<br>89c765625df9df98<br>3345678964ab500<br>000000 .... 0000(h) | The **Checksum** field shall carry the value of the checksum of the **Payload** field contained within the Binary Structure Container, determined using the algorithm specified in the **Checksum Type** field. It shall carry the value in the number of left-most bytes specified for the Checksum Type used and shall be padded in its right-most bytes with NUL values to a length of 512 bytes. |
| Structure Identifier 2 | 648 +<br>Payload Description Length +<br>Payload Format Length +<br>Payload Length +<br>Padding Length | 32 | UTF-8 | AXF_OBJECT_HEADER | The **Structure Identifier 2** field shall carry a duplicate copy of the value contained in the **Structure Identifier 1** field. |
| Chunk Size 2 | 680 +<br>Payload Description Length +<br>Payload Format Length +<br>Payload Length +<br>Padding Length | 8 | Unsigned 64-bit integer | 100000(h) | The **Chunk Size 2** shall indicate in bytes the Chunk size used for the current AXF Object. The same value is duplicated in the **Chunk Size 1** field. |



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

| Field Name | Byte Offset | Length (Bytes) | Data Type | Example | Description and Requirements |
|---|---|---|---|---|---|
| Structure Start Position | 688 +<br>Payload Description Length +<br>Payload Format Length +<br>Payload Length +<br>Padding Length | 8 | Signed 64-bit integer | 1a2b3c4d5e6f7a8b(h) | The **Structure Start Position** field shall specify the relative number of Chunks from the location of the **Structure Start Position** field to the first Chunk of the current Binary Structure Container. (Note: If the Binary Structure Container is contained in one Chunk, the value of **Structure Start Position** will be zero. Otherwise, the value of Structure Start Position always will be negative.)<br><br>Implementation note: SMPTE ST 2034-1:2014 specified the Data Type of **Structure Start Position** as an Unsigned 64-bit integer. AXF Objects created using that version will have positive values for **Structure Start Position**. On encountering positive values for **Structure Start Position**, AXF-aware applications compliant with this version should interpret them as negative values or zero to conform with the current definition of **Structure Start Position**. At the time that such AXF Objects are refreshed in storage, adjustment of the values to comply with the definition above is recommended. |



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017

### 6.4.2 Media Data Structures

Each medium formatted according to this standard shall include specific data structures that define it as a valid AXF Medium. Depending upon the Storage Media Type with respect to Block-Based or File-System-Based media, certain structures are either required or optional (see Section 5.4 and Table 1).

One or more AXF Object may be stored on a single medium, and a single AXF Object may be stored on one or more media through use of the Spanning process (see Section 8). The layout of the various data structures on a medium is specific to the type of medium as described above. The general layout of AXF Objects on an AXF Medium is shown in Figure 3, which is representative of Block-Based Media – the most complex example.

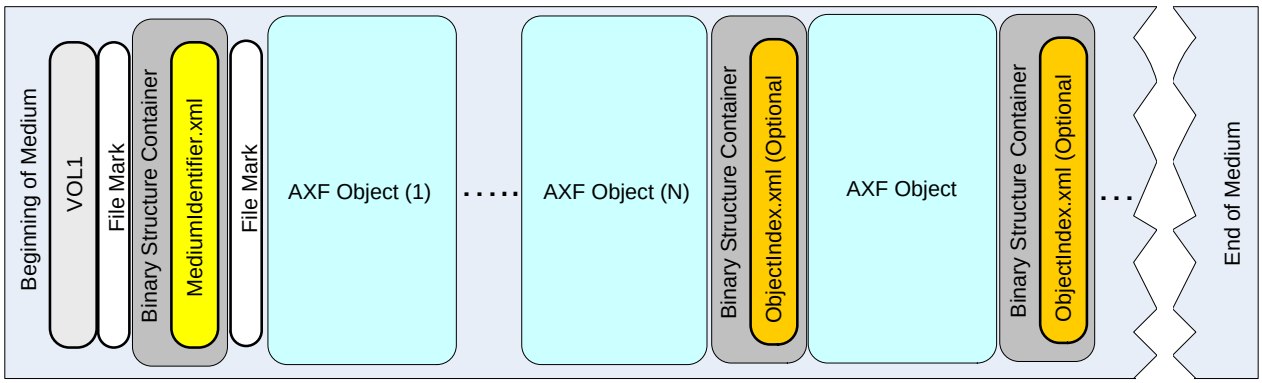


**Figure 3 – AXF Structure Layout on Block-Based Media**

The following sections detail AXF Media-specific structures. Section 5 defines which media data structures are applicable to each particular Storage Media Type.

#### 6.4.2.1 VOL1 Label

The VOL1 Label is an ISO/ANSI standard volume label ("VOL1") structure as specified in ISO/IEC 1001:2012(E). This label is included to maintain compatibility with legacy applications for Storage Media Types that benefit from its use. The VOL1 Implementation Identifier indicates to applications with AXF compatibility that the medium has been properly prepared for AXF read/write activities. The VOL1 Label shall not be encapsulated inside a Binary Structure Container. The VOL1 Label shall be stored on a medium so that the first byte of the label is carried in the first byte of the first available block on the medium. Applications shall write the VOL1 Label structure when preparing a medium for AXF use and shall read the VOL1 Label**,** in combination with the AXF Medium Identifier (see Section 6.4.2.2), to detect whether the medium has been prepared for AXF Objects before any are stored.

To maintain compatibility with non-AXF applications, the VOL1 Label shall be followed by a device-specific File Mark on applicable media. The fields of the VOL1 Label are structured as defined in Table 3. See Table 1 for designations of the Storage Media Types on which the VOL1 Label shall be applied.



© ISO/IEC 2017 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 12034-1:2017