**ISO/IEC 23090-39:2025**

ISO/IEC JTC 1/SC 29/WG 03

Secretariat: XXXX

**Information technology — Coded representation of immersive media — Part 39: Avatar Representation Format**

CD stage

**Warning for WDs and CDs**

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

© ISO 2025

All rights reserved. Unless otherwise specified, or required in the context of its implementation, 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

CP 401 • Ch. de Blandonnet 8

CH-1214 Vernier, Geneva

Phone: +41 22 749 01 11

Fax: +41 22 749 09 47

Email: copyright@iso.org

Website: www.iso.org

Published in Switzerland

Contents

[Foreword 5](#_Toc197899392)

[Introduction 6](#_Toc197899393)

[1 Scope 1](#_Toc197899394)

[2 Normative references 1](#_Toc197899395)

[3 Terms, definitions, symbols and abbreviated terms 1](#_Toc197899396)

[3.1 Terms and Definitions 1](#_Toc197899397)

[3.2 Abbreviated Terms 1](#_Toc197899398)

[4 Overview 2](#_Toc197899399)

[4.1 System Description 2](#_Toc197899400)

[4.2 Brands 3](#_Toc197899401)

[4.3 Schemes 3](#_Toc197899402)

[5 Data Model of Base Avatar 3](#_Toc197899403)

[6 Avatar Representation Format Document 4](#_Toc197899404)

[6.1 General 4](#_Toc197899405)

[6.2 Preamble 5](#_Toc197899406)

[6.2.1 Overview 5](#_Toc197899407)

[6.2.2 Authentication Features 6](#_Toc197899408)

[6.2.3 Supported Animations 6](#_Toc197899409)

[6.2.4 Proprietary Animation 7](#_Toc197899410)

[6.3 Metadata 8](#_Toc197899411)

[6.4 Structure 8](#_Toc197899412)

[6.4.1 Overview 8](#_Toc197899413)

[6.4.2 Asset 9](#_Toc197899414)

[6.4.3 Level of Detail (LOD) 9](#_Toc197899415)

[6.5 Components 10](#_Toc197899416)

[6.5.1 Overview 10](#_Toc197899417)

[6.5.2 Skeleton 10](#_Toc197899418)

[6.5.3 Skin 11](#_Toc197899419)

[6.5.4 Mesh 12](#_Toc197899420)

[6.5.5 BlendshapeSet 12](#_Toc197899421)

[6.5.6 LandmarkSet 13](#_Toc197899422)

[6.5.7 TextureSet 13](#_Toc197899423)

[6.5.8 AnimationLink 14](#_Toc197899424)

[6.5.9 Node 14](#_Toc197899425)

[6.5.10 Mapping 15](#_Toc197899426)

[6.5.11 LinearAssociation 15](#_Toc197899427)

[6.5.12 TextureTarget 16](#_Toc197899428)

[6.6 Data 17](#_Toc197899429)

[6.7 ProtectionConfiguration 17](#_Toc197899430)

[7 ARF Container Format 17](#_Toc197899431)

[7.1 General 17](#_Toc197899432)

[7.2 ISOBMFF-based container format 18](#_Toc197899433)

[7.2.1 General 18](#_Toc197899434)

[7.2.2 Avatar Component Information 18](#_Toc197899435)

[7.2.3 Avatar Animation Tracks 19](#_Toc197899436)

[7.3 Zip-based container format 21](#_Toc197899437)

[7.3.1 Overview 21](#_Toc197899438)

[7.3.2 Avatar Animation Stream 22](#_Toc197899494)

[8 Animation Stream Format 25](#_Toc197899495)

[8.1 General 25](#_Toc197899496)

[8.2 Facial Animation Sample Format 26](#_Toc197899497)

[8.3 Joint Animation Sample Format 27](#_Toc197899498)

[8.4 Landmark animation sample format 28](#_Toc197899499)

[Annex A (normative) ARF Document JSON Schema 29](#_Toc197899500)

[Annex B (normative) Integration into Scene Description 40](#_Toc197899501)

[Annex C Reference Avatar Client 42](#_Toc197899502)

[Annex D Authentication Procedure (Informative) 43](#_Toc197899503)

[D.1 Introduction 43](#_Toc197899504)

[Annex E Tensor Data Format (Normative) 44](#_Toc197899505)

[E.1 Tensor Data Format 44](#_Toc197899506)

[E.2 MIME Type Registration 44](#_Toc197899507)

[E.3 Registration Form 44](#_Toc197899508)

[Annex F Examples (Informative) 46](#_Toc197899509)

[Bibliography 47](#_Toc197899510)

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.

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](https://www.iso.org/directives-and-policies.html) or [www.iec.ch/members\_experts/refdocs](https://www.iec.ch/members_experts/refdocs)).

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](https://www.iso.org/iso-standards-and-patents.html)) or the IEC list of patent declarations received (see <https://patents.iec.ch>).

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

For an explanation of 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 [www.iso.org/iso/foreword.html](https://www.iso.org/iso/foreword.html). In the IEC, see [www.iec.ch/understanding-standards](https://www.iec.ch/understanding-standards).

This document was prepared by Technical Committee ISO/IEC/JTC 1 *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

A list of all parts in the ISO 23090 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at [www.iso.org/members.html](https://www.iso.org/members.html) and [www.iec.ch/national-committees](https://www.iec.ch/national-committees).

Introduction

This document defines an Avatar Representation Format (ARF). For this purpose, the document defines a data model for the Avatar Representation Format, a data document that describes the components of an ARF base avatar model, several container formats for carriage , animation sample formats for transmission of animation parameters, and a binary format for the streaming of the Avatar Representation Format.

Information technology — Coded representation of immersive media — Part 39: MPEG Avatar Representation Format

# Scope

This document specifies the Avatar Representation Format (ARF) with the goal of offering an interoperable exchange format for the storage, carriage and animation of 3D avatars.

# Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document.

* ISO/IEC 23090-14, Information technology — Coded representation of immersive media Part 14: Scene Description
* IETF RFC 8259, The JavaScript Object Notation (JSON) Data Interchange Format
* ISO/IEC 21320-1, Information technology — Document Container File Part 1: Core

# Terms, definitions, symbols and abbreviated terms

## Terms and Definitions

The following terms and definitions are listed in this document.

|  |  |
| --- | --- |
| Asset | independently accessible element of an avatar |
| Avatar | 3D graphics-based representation of a user. |
| Animation Data | skeletal, blend shape set, and other animation-related information. |
| Animation Streams | timed data used to animate the base avatar |
| Base avatar model | personalized and animatable 3D model of the user |
| Blend shape | displacements and/or variations of the based avatar model to express key-frame animations |
| ARF container | container that includes all components of the base avatar model, its associated digital assets, and the related metadata |
| ARF document | JSON-formatted document that acts as the entry point to an ARF container |
| Joint | specifies a spatial location of a skeletal joint of the avatar model. |
| Skeleton | A hierarchical representation of joints that are connected with bones to form the skeletal structure of the base avatar model. |

## Abbreviated Terms

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at [https://www.iso.org/obp](https://www.iso.org/obp/ui)

— IEC Electropedia: available at <https://www.electropedia.org/>

|  |  |
| --- | --- |
| ARF | Avatar Representation Format |
| ISOBMFF | ISO base media file format |
| HMD | Head-Mounted Display |
| JSON | JavaScript Object Notation |
| LBS | Linear Blend Skinning |
| LoD | Level of Detail |
| ML | Machine Learning |

# Overview

## System Description

The Avatar Representation Format (ARF) defined in this document focuses specifically on two key components of an avatar animation system: (i) the Base Avatar Format and, (ii) the Animation Stream Format. These standardized formats, highlighted in dashed gray boxes in Figure 1, form the core scope of this document, enabling interoperable avatar animation across different implementations.

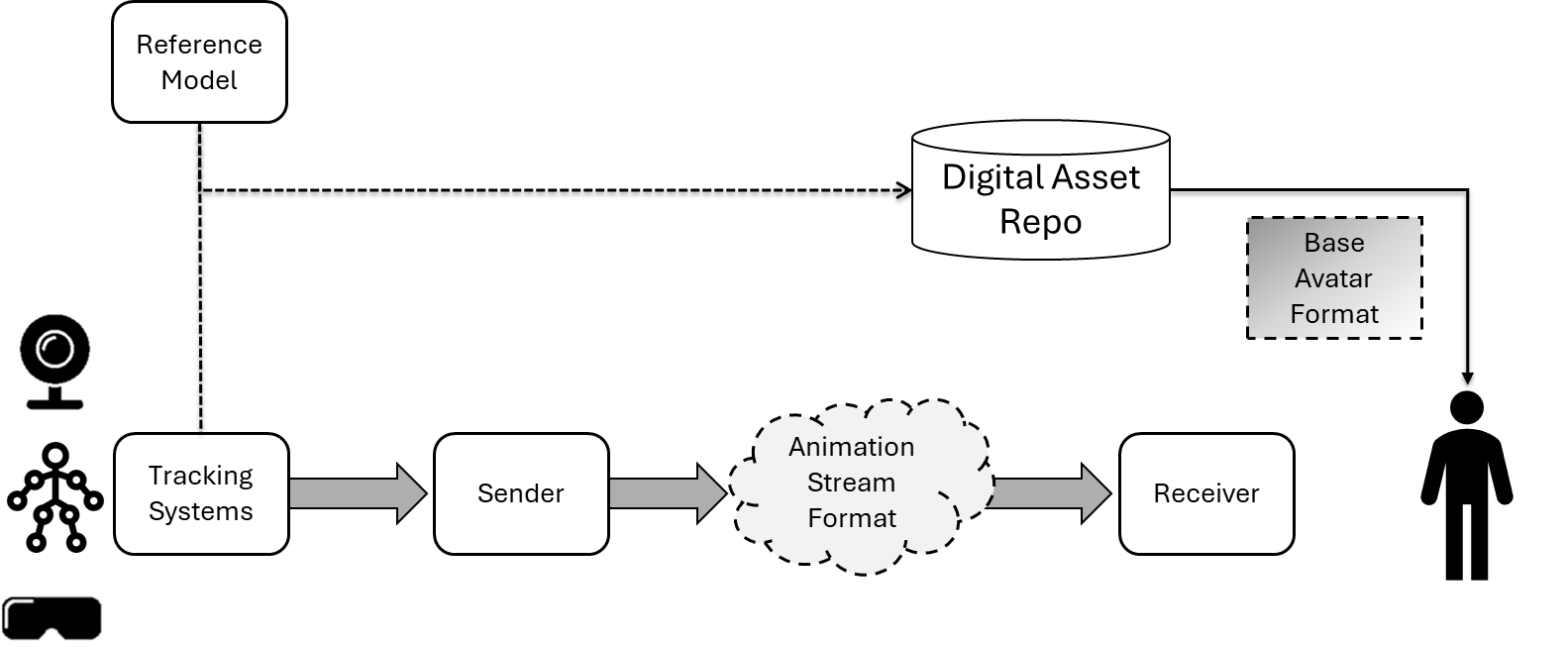


Figure 1 — Avatar reference architecture

The Base Avatar Format establishes the standardized representation for avatar models, which can then be stored in a digital asset repository, ensuring that the fundamental avatar assets can be reliably accessed and animated by the receiving entity. A data model for the base avatar is defined in clause 5. A document describing the Base Avatar is defined in clause 6, referred to as ARF Document.

The Animation Stream Format defines how animation data is structured and carried between senders and receivers. This format defines how facial and body animation information is encoded, allowing data captured from input devices like Head-Mounted Displays (HMDs) and sensors to be consistently interpreted across different systems for the animation of associated avatars.

Other components in Figure 1 are considered outside the scope of this document and may be implemented in different ways.

## Brands

The ISO base media file format, ISO/IEC 14496-12, defines the concept of brands; brand values identify specifications or conformance points. This document specifies several brands, as listed in Table 1.

Table 1 — Brands defined in this document

|  |  |  |
| --- | --- | --- |
| **Brand identifier** | **Clause in this document** | **Informative description** |
| 'ARF' | 7.2 | every ISOBMFF-based container shall declare ARF as the major brand. |
| 'maas' | 7.2 | Files that contain stored animation streams shall declare maas among their compatibility brands. |

## Schemes

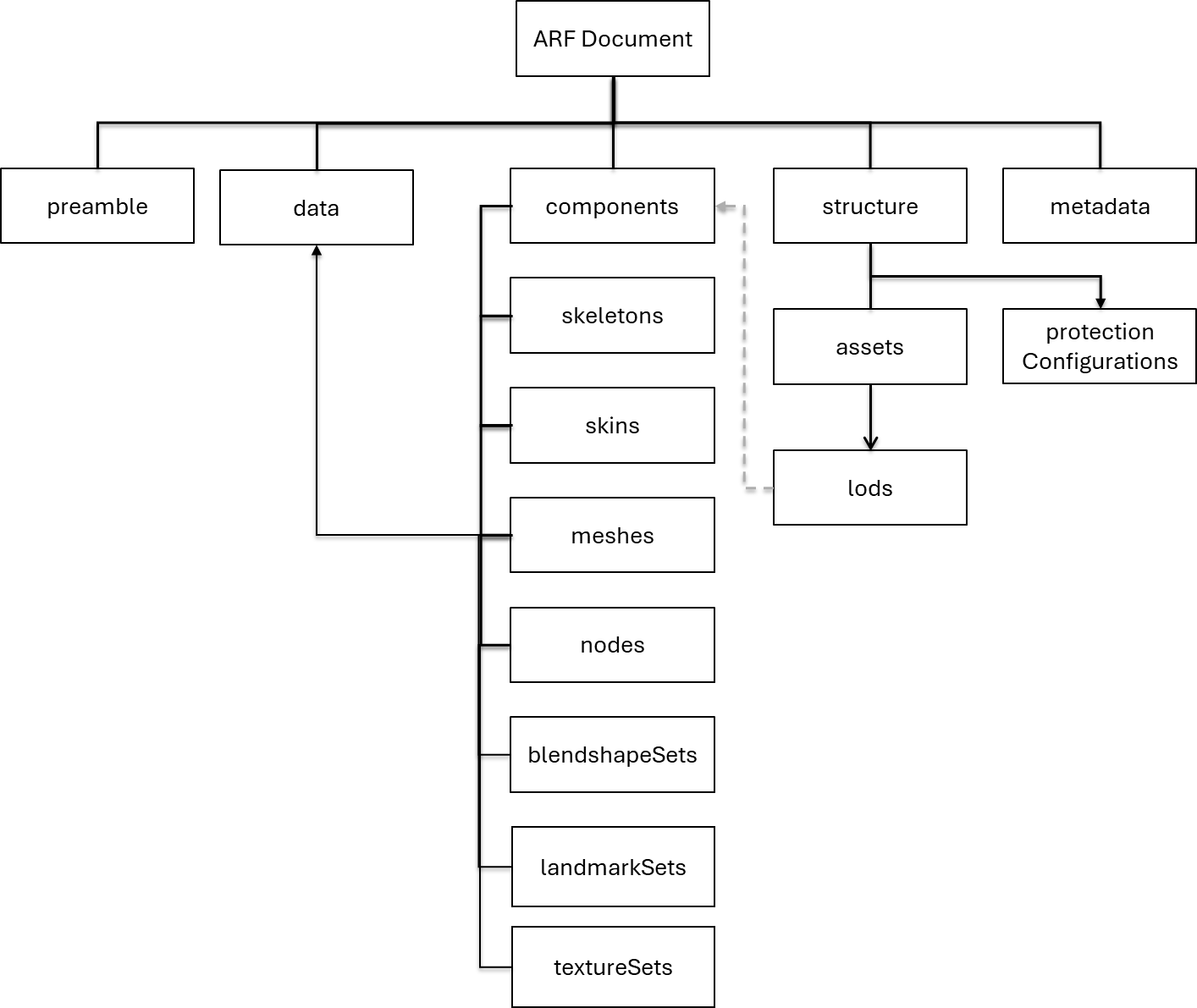
This document specifies several schemes as listed in Table 2.

Table 2 — Schemes defined in this document

| **Scheme identifier** | **Clause in this document** | **Informative description** |
| --- | --- | --- |
| urn:mpeg:avatar:animation | 7.2 | The URI identifying the type of the metadata in the ISO BMMF. |

# Data Model of Base Avatar

This clause defines a data model for the Base Avatar following the illustration in Figure 2.

  
Figure 2 — ARF document structure

The description of each of these components is provided in clauses 6 and 7.

# Avatar Representation Format Document

# General

The Avatar Representation Format (ARF) document describes the user’s base avatar model. The document shall conform to the JavaScript Object Notation (JSON) Data Interchange Format according to IETF RFC 8259 and shall validate against the JSON schema as defined in Annex A.

The document shall contain objects and properties as defined in the remainder of this clause. In particular the data formats defined by the 'Name', 'Type', 'Use' and 'Description' in the tables in the remainder of this clause shall apply.

Table 3 defines the high-level component objects of the ARF document.

Table 3 — High-level component objects of the ARF document

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| preamble | Preamble | M | specifies data that uniquely identifies the format and characteristics of the ARF container.  For details refer to clause 6.2. |
| metadata | Metadata | M | specifies metadata related to the base avatar model.  For details refer to clause 6.3. |
| structure | Structure | M | Contains the data structures of the ARF container.  For details refer to clause 6.4. |
| components | Components | M | Contains the core elements of the base avatar model. It lists the main ARF containers to represent and animate the base avatar.  For details refer to clause 6.5. |
| data | array(Data) | M | Contains the data for each element of the components of the ARF container.  For details refer to clause 6.6. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

# Preamble

### Overview

The **Preamble** is used to uniquely identify the format and characteristics of the Avatar Representation Format. It carries a unique signature as well as information about the compatible animation frameworks that work with this base avatar model.

Table 4 defines the Preamble object.

Table 4 — Preamble object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| signature | string | M | Specifies a unique identifier of this object within ARF document. |
| version | string | M | specifies the version of the MPEG Avatar Representation Format. |
| authenticationFeatures | array(AuthenticationFeatures) | O | specifies a set of features that are used to identify the owner of this base avatar.  For more details refer to clause 6.2.2. |
| supportedAnimations | SupportedAnimations | M | contains information about the supported animation types.  For more details refer to clause 6.2.3. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### Authentication Features

The authentication features are used to uniquely associate a base avatar model in ARF format to its owner.

Table 5 defines the authenticationFeatures object.

Table 5 — Definition of authenticationFeatures object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| publicKey | URI | M | A URL to the public key that is used to decrypt the features. |
| facialFeature | string | O | A base64 encoded feature vector of floats. This can be used to match extracted facial features during a communication session. The facial feature shall be encoded with the user’s private key to preserve authenticity. |
| voiceFeature | string | O | A base64 encoded feature vector of floats. This can be used to match extracted voice features during a communication session. The voice feature shall be encoded with the user’s private key to preserve authenticity. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### Supported Animations

The supported animation identifies the type of animation supported by the avatar format.

Table 6 defines the supportedAnimations object.

Table 6 — Definition of supportedAnimations object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| faceAnimations | array(uri) | O | Lists the supported face animation types. Each item in the array is a string representing a supported face animation type.    Each identifier should be formatted as a URN that includes an identifier of the framework, followed by an identifier of the facial blendshape set. |
| bodyAnimations | array(uri) | O | Lists the supported body animation types. Each item in the array is a string representing a supported body animation type.    Each identifier should be formatted as a URN that includes an identifier of the body animation/tracking framework, followed by an identifier of the body joint set. |
| handAnimations | array(uri) | O | Lists the supported hand animation types. Each item in the array is a string representing a supported hand animation type.      Each identifier should be formatted as a URN that includes an identifier of the body animation/tracking framework, followed by an identifier of the body joint set. |
| landmarkAnimations | array(uri) | O | Lists the supported landmark animation types. Each item in the array is a string representing a supported landmark animation type.  Each identifier should be formatted as a URN that includes an identifier of the landmark animation/tracking framework, followed by an identifier of the landmark set. |
| proprietaryAnimations | array( ProprietaryAnimation) | O | A list of proprietary animation descriptions, which may be used to animate assets in the ARF container.  For details refer to clause 6.2.4. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### Proprietary Animation

The component **proprietaryAnimations** provides information on how to use an external Machine Learning (ML) model to reconstruct or animate assets in the ARF container.

Table 7 defines the proprietaryAnimation object.

Table 7 — Definition of proprietaryAnimation object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| id | number | M | A unique identifier of this proprietary animation scheme. |
| scheme | URI | M | A vendor-specific URN to identify the proprietary reconstruction and animation scheme. |
| items | array(number) | M | A list of data item references, e.g. pretrained models or model weights, that are used by this proprietary reconstruction and animation scheme. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

# Metadata

The **Metadata** component contains information about the owner of the base avatar model, some physical characteristics of the base avatar, such as sex, age and height, as well as other metadata related to security and protection of the base avatar model.

Table 8 defines the Metadata object.

Table 8 — Definition of Metadata object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| name | string | M | A string that describes the name of the avatar. |
| id | string | M | A string that uniquely identifies the avatar. |
| age | integer | M | An integer value to define the age of the avatar. |
| gender | string | M | A string that describes the gender of the avatar. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

# Structure

### Overview

The **Structure** component describes the structure of the ARF container. It lists the assets and levels of detail included in this ARF container. It also provides information about the required encryption scheme to decrypt the components of this ARF container that are encrypted.

Table 9 defines the Structure object.

Table 9 — Definition of Structure object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| assets | array(Asset) | M | List the assets included in the ARF container.  For details refer to clause 6.4.2. |
| protectionConfigurations | array(ProtectionConfiguration) | O | A list of protection configuration objects that are used for the protection of components of the ARF container.  For details refer to clause 6.7. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### Asset

The **Asset** constitute the key part of the ARF container. An ARF container can contain multiple assets that define the base avatar model of the user or that are associated with it (e.g. digital assets like garments and wearables). Each asset can be accessed and extracted individually.

Table 10 defines the Asset object.

Table 10 — Definition of Asset object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| name | string | M | The name of the asset. |
| lods | array(LOD) | M | A list of level of details available for this asset in the ARF container.  For details refer to clause 6.4.3. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### Level of Detail (LOD)

The **LOD** object defining the Level of Detail (LOD) provides a link to all components of an asset at a specific level of detail. This facilitates partial access to the ARF container by allowing to extract the desired assets at the desired level of detail.

Table 11 defines the LOD object.

Table 11 — Definition of LOD object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| name | string | M | The name of the LOD. |
| skins | array(number) | O | List of references to all skins that are part of this asset. |
| meshes | array(number) | M | List of non-skinned meshes that are part of this asset. |
| skeletons | array(number) | O | List of references to skeletons in the ARF container. |
| blendshapeSets | array(number) | O | List of references to blend shape sets in the ARF container. |
| landmarkSets | array(number) | O | A list of references to landmark sets in the ARF container. |
| textureSets | array(number) | O | A list of references to texture sets in the ARF container. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

# Components

### Overview

The **Components** component is the core of the ARF document. It lists all the components of the ARF container and provides sufficient information to access and use these components for the reconstruction and animation of the base avatar model.

Table 12 defines the Components object.

Table 12 — Definition of Components object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| skeletons | array(Skeleton) | O | A list of skeletons used to describe the avatar skeletal asset.  For details refer to clause 6.5.2. |
| skins | array(Skin) | O | A list of assets that are stored in this ARF container.  For details refer to clause 6.5.3. |
| meshes | array(Mesh) | M | A list of geometries used to describe the avatar asset.  For details refer to clause 6.5.4. |
| nodes | array(Node) | O | A list of nodes used to organize, merge and describe and transform the avatar components.  For details refer to clause 6.5.9. |
| blendshapeSets | array(BlendshapeSet) | O | A list of blend shape sets used to describe the blend shape-based animations.  For details refer to clause 6.5.5. |
| landmarkSets | array(LandmarkSet) | O | A list of landmark sets used to describe landmark-based animation.  For details refer to clause 6.5.6. |
| textureSets | array(TextureSet) | O | A list of texture sets used to describe parametric textures.  For details refer to clause 6.5.7. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### Skeleton

The **Skeleton** component describes a partial or complete skeleton that is used in the ARF container. The skeleton describes the joints and their relationships.

Table 13 defines the Skeleton object.

Table 13 — Definition of Skeleton object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| name | string | M | The name of the skeleton. |
| root | number | M | Reference to the root joint for the skeleton in the nodes collection. |
| joints | array(number) | M | List of references to the list of joints in node collection of the ARF container. |
| inverseBindMatrix | number | M | References an item in the data collection of the ARF container that contains the inverse bind matrices for the joints in the same order as the joints.  The data should be an Nx16 tensor, where N is the number of joints in the skeleton.  The tensor format is defined in Annex E. |
| animationInfo | array(AnimationLink) | O | Establishes a link to the supported animation and tracking frameworks that this skeleton animation can be used with.  For details refer to clause 6.5.8. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### Skin

The **Skin** component is a skinned mesh representing a part of the Avatar body or an associated digital asset. A skin defines the mapping between a mesh and a skeleton, enabling mesh deformation through a skeletal animation system.

Table 14 defines the Skin object.

Table 14 — Definition of Skin object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| name | string | M | The name of the skin. |
| mapping | string | M | this contains a path indicator that can be used to assign this skinned mesh to a particular node in the scene graph. |
| skeleton | number | M | a reference to the skeleton. |
| mesh | number | M | a reference to the mesh of the skin. |
| weights | number | M | reference to an item in the data collection that contains the weights. These weights correspond to the influence of a set of joint transformations on the mesh vertices positions.  The weights is provided as an NxM-tensor, where N is the number of vertices and M is the number of joints. The tensor format is defined in Annex E. |
| proprietaryAnimations | array(number) | O | An array of references to proprietaryAnimation objects that define a proprietary animation approach that applies to this skin. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### Mesh

The component **Mesh** defines the 3D geometrical primitive of the avatar containing its topology and 3D shape.

Table 15 defines the Mesh object.

Table 15 — Definition of Mesh object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| name | string | M | The name of the mesh. |
| id | number | M | The identifier of the mesh. |
| path | string | M | A string that represents a hierarchical path that can be used to associate the mesh with a node in the external scene graph e.g., “full\_body/upper\_body/head”. |
| data | array(number) | M | A reference into a data item that contains the mesh data. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### BlendshapeSet

The **BlendshapeSet** component defines a set of shapes that deform a given base mesh.

Table 16 defines the BlendshapeSet object.

Table 16 — Definition of BlendshapeSet object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| name | string | M | The name of the blendshape set. |
| id | number | M | A unique identifier of the blendshape set. This id is used in the facial animation to associate the weights with the shapes. |
| animationInfo | array(AnimationLink) | O | Establishes a link to the supported animation and tracking frameworks that this belnd shape set can be used with.  For details refer to clause 6.5.8. |
| shapes | array(number) | M | An array of references to data items that contain each blendshape’s data. |
| baseMesh | number | M | A reference to a data item that contains the base mesh for this blend shape set. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### LandmarkSet

The **LandmarkSet** component defines a set of landmarks that relate to a mesh and can be used to deform that mesh.

Table 17 defines the LandmarkSet object.

Table 17 — Definition of LandmarkSet object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| name | string | M | The name of the landmark set. |
| id | number | M | A unique identifier of the landmark set. This id is used in the facial animation to associate the landmark vertices positions with the landmark vertices. |
| animationInfo | array(AnimationLink) | O | Establishes a link to the supported animation and tracking frameworks that this landmark set can be used with.  For details refer to clause 6.5.8. |
| baseMesh | number | M | The base mesh that is associated with the landmark vertices. |
| vertices | number | M | A reference to the Data object that provides the list of vertex indices that make up the landmark set. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### TextureSet

The **TextureSet** component defines a set of textures related to a material of a data object, and are used to enhance the visual quality of that object by adding to one of the texture of the material a linear combination of texture targets.

Table 18 defines the TextureSet object.

Table 18 — Definition of TextureSet object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| name | string | M | The name of the texture set |
| id | number | M | A unique identifier of the texture set. |
| animationInfo | array(AnimationLink) | M | Establishes a link to the supported parametric texture frameworks that this texture set can be used with.  For details refer to clause 6.5.8. |
| material | number | M | A reference to the Data object that provides the component with a material. |
| materialPath | string | M | Indicates where the texture can be found in the Data object referenced by “material”. |
| targets | array(TextureTarget) | M | The list of texture targets.  For details refer to clause 6.5.12. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### AnimationLink

The **AnimationLink** object establishes a link between an animation component and a list of supported animation frameworks.

Table 19 defines the AnimationLink object.

Table 19 — Definition of AnimationLink object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| type | enumeration | M | The type of the supported animation. The allowed types are:   * ANIMATION\_FACE * ANIMATION\_BODY * ANIMATION\_HAND * ANIMATION\_LANDMARK |
| target | number | M | Provides the index of the target animation framework in the associated supported animations list for which these mappings apply. |
| mappings | array(Mapping) | O | Provides a list of Mapping objects associated with this target animation framework. The Mapping object is defined in clause 6.5.10 |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### Node

The **Node** component defines the skeletal joints hierarchy and structure for the ARF container. Each skeleton in the ARF container makes reference to a set of nodes.

Table 20 defines the Node object.

Table 20 — Definition of Node object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| name | string | M | The name of the node. |
| mapping | string | M | The joint type or semantics e.g., "full\_body/upper\_body/right\_arm". The elements of the path hierarchy should follow the naming convention as defined in table 29 of 23090-14. |
| parent | number | O | If present, the identifier of the parent node of this node. This attribute shall be present for all nodes, except for the root. |
| children | array | O | if present, a list of identifiers of the children nodes of this node. |
| scale | array(number) | O | The node’s non-uniform scale, given as the scaling factors along the x,y and z axes. |
| rotation | array(number) | O | The node’s unit quaternion rotation in the order (x,y,z,w), where w is the scalar. |
| translation | array(number) | O | The node’s translation along the x,y and z axes. |
| transform | array(number) | M | Provides a 4x4 transformation matrix for the node to define its position and orientation. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### Mapping

The **Mapping** object provides a way to signal mappings between source animation frameworks and the parent target animation framework, such as a blendshape set, a skeleton, or a landmark set.

The *Mapping* object is defined in table 21:

Table 21 — Definition of a Mapping object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| source | number | M | provides the index of the source animation framework, of which the animation data is mapped to the target animation framework, using the provided mapping table. |
| associations | array(LinearAssociation) | M | an array of linear associations mapping a set of values from the source animation framework to one value of the target animation framework. The LinearAssociation object is defined in clause 6.5.10. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

The source property gives the index of the animation framework in the associated supported animations list (see Section 6.2.3 of the ARF specifications) from which the mapping is done.

### LinearAssociation

The **LinearAssociation** object defines a linear mapping between a set of values from the source animation framework to a value of the target animation framework. For example, blend shape #5 of the target animation framework is a weighted sum of blend shapes #4 and #52 from the source animation framework.

The LinearAssociation object is defined in the following table:

Table 22 — Definition of a LinearAssociation object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| targetIndex | number | M | provides the index of the value to be produced by this linear association. For example, for a blend shape set, this would indicate the index of the blend shape. |
| sourceIndices | array(number) | M | Provides an array of indices of the values from the source animation framework referenced that contribute to the target index value. |
| weights | array(number) | M | The associated weights for the mapping of the contributing source animation value into the target animation value. The weights shall be provided in the same order as the contributing animation ids in sourceIndices.  The animation weight of the target animation value with index targetIndex is calculated as follows for blend shape animations:  The transform matrix of the target animation value with index targetIndex is calculated as follows for joint animations: |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### TextureTarget

The **TextureTarget** component defines one of the texture targets used to improve a texture of the material referenced by the TextureSet.

Table 23 defines the TextureTarget object.

Table 23 — Definition of TextureTargets object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| name | string | M | The name of the texture target. |
| id | integer | M | A unique identifier of the texture target. |
| texture | number | M | References an item in the data root component with a texture. |
| texturePath | string | O | Indicates where the texture can be found in the item referenced by “texture”. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

# Data

The **Data** object contains the low-level content of the ARF container e.g., meshes, tensors, images, or other data. Each data item may be compressed and/or encrypted.

Table 24 defines the Data object.

Table 24 — Definition of Data object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| name | string | M | a string that defines the name of this data. |
| type | string | M | a string that provides the mime type of the data. |
| uri | string | M | a string that defines the data content or reference to the data content depending on type. |
| offset | integer | O | defines the number of bytes used as offset into the data content as pointed to by uri. |
| byteLength | integer | O | defines the number of bytes to use in data content. |
| compression | string | O | an identifier of the compressor used to compress this LoD representation of the mesh. The compressor shall be identified by a URN. |
| protection | number | O | an identifier of the protection configuration that is applied to encrypt this LoD representation of the mesh. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

# ProtectionConfiguration

The **ProtectionConfiguration** object provides the necessary information to describe and access a protection scheme that is needed to decrypt one or more components of the ARF container.

Table 25 defines the ProtectionConfiguration object.

Table 25 — Definition of ProtectionConfiguration object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| schemeIdUri | string | M | identifies a protection or encryption scheme. |
| value | object | O | Provides additional information specific to the protection or encryption scheme. For example, it may provide information such as DRM version, encryption mode, etc. The contents of this object are proprietary to the protection scheme. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

# ARF Container Format

# General

The ARF container is an integral component of the Avatar Representation Format (ARF), which is designed to facilitate efficient and flexible avatar representation and transmission in communication and shared space sessions. It acts as a structured repository for all the elements that constitute the user’s base avatar model, thus enabling seamless integration and animation across platforms and applications.

The ARF document as defined in clause 4 shall be marked as the entry point to the ARF container. The ARF document describes all the components that make up the user’s base avatar model. All components that are described by the ARF document shall be stored in the ARF container and the addressing scheme shall allow for locating these components within the ARF container.

A key feature of the ARF container format is its support for partial access. This means that depending on the specific requirements of the application or on the network conditions, only a subset of the user’s base avatar components need to be downloaded. The selection of the components is based on factors like the desired level of detail (LoD), the target bitrate, the user’s selection (e.g. the skinned meshes that represent garments).

The ARF container format plays a crucial role in enabling real-time avatar-based communication and shared experiences. By providing a standardized and interoperable way to store and transmit avatar data, it streamlines the process of sharing and animating avatars across different platforms and applications. In a typical scenario, a user would first create and upload their base avatar model to a central server. When participating in a communication or shared experience session, the user's avatar information, including the location of the ARF container, is shared with other participants. Based on the received information and the negotiated access level, the other participants can then download the container with only the necessary/authorized components of the user's avatar and animate it in real time using the transmitted animation streams.

In this specification, we define two ARF container formats for the storage of the user’s base avatar model. The first one is ISOBMFF-based and the second is Zip-based.

# ISOBMFF-based container format

### General

ISO/IEC 14496-12 defines the concept of brands, which may be indicated in the FileTypeBox.

When stored in an ISOBMFF-based container, the user’s base model shall be stored as metadata items with the MetaBox being declared at the file level. A PrimaryItemBox shall be present and shall contain the item identifier of the item that contains the ARF document.

The following shall apply:

* The HandlerBox shall have a handler\_type set to 'AVRF'
* The primary item shall declare content\_type of "model/ARF+json"
* It may contain an item protection box that defines the encryption for the components of the base avatar model that are protected.
* each component of the base avatar model, including the different LoD variants,  shall be stored as an independent item.

### Avatar Component Information

Each component item is associated with an AvatarComponentInfoProperty that describes which avatar, asset, and level-of-detail the component is associated with. A corresponding AvatarComponenatInfoProperty instance shall be present in the ItemPropertyContainerBox of the ItemPropertiesBox, defined in ISO/IEC 14496-12, for each component item.

The AvatarComponentInfoProperty is defined as follows.

Table 26 — Syntax of AvatarComponentInfoProperty

|  |
| --- |
| aligned(8) class AvatarComponentInfoProperty  extends ItemProperty(‘avcp’) {  unsigned int(1) static\_association\_flag;  bit(7) reserved = 0;  if (static\_association\_flag) {  unsigned int(8) avatar\_id;  unsigned int(8) asset\_id;  }  unsigned int(4) component\_type;  unsigned int(4) level\_of\_detail;  } |

The semantics of the fields of AvatarComponentInfoProperty are as follows:

static\_association\_flag is a flag indicating if the component is associated with a single avatar. Value 0 indicates that the component may be associated with more than one avatar. Value 1 indicates that the component is associated with a single avatar whose identifier is given by the avatar\_id field.

avatar\_id is the unique identifier for the avatar that this component is associated with. This field is only present if static\_association\_flag is set to 1.

asset\_id is the unique identifier for the avatar asset that this component is associated with. This field is only present if static\_association\_flag is set to 1.

component\_type is an integer indicating the type of the component. Values 0 to 5 designate the component types: skeleton, skin, mesh, node, blend shape set, and landmark set, respectively. Other values are reserved for future use.

level\_of\_detail indicates the level of detail of the asset to which the component is associated.

The association between each component item and its AvatarComponentInfoProperty is done using the ItemPropertyAssociationBox, defined in ISO/IEC 14496-12. The essential bit flag shall be set to 1 for each property entry in the ItemPropertyAssociationBox referring to an AvatarComponentInfoProperty, signalling that it is an essential property of the item.

To identify all the components that relate to a particular avatar model in the container, a SingleItemTypeReferenceBox with reference type ‘avcr’ shall be present in the ItemReferenceBox, where the from\_item\_ID field is set to the item\_ID of the avatar item and list of to\_item\_IDs corresponding to each component item.

### Avatar Animation Tracks

#### General

When animation streams are also stored as part of the ARF container, at least one avatar animation track shall be present in the file and shall carry the avatar animation samples. Avatar animation tracks are timed-metadata tracks whose samples carry avatar animation data. An avatar animation track has a sample entry of type AvatarAnimationSampleEntry as defined in Table 27, where AvatarAnimationConfigurationBox is defined in Table 28.

Table 27 — Syntax of AvatarAnimationSampleEntry

|  |
| --- |
| aligned(8) class AvatarAnimationSampleEntry() extends MetadataSampleEntry('ava1') {  AvatarAnimationConfigurationBox config;  } |

Table 28 — Syntax of AvatarAnimationConfigurationBox

|  |
| --- |
| aligned(8) class AvatarAnimationConfigurationBox extends FullBox('avaC', version=0, flags=0) {  AvatarAnimationConfigurationRecord() ava\_animation\_config;  } |

Table 29 — Syntax of AvatarAnimationConfigurationRecord

|  |
| --- |
| aligned(8) class AvatarAnimationConfigurationRecord {  unsigned int(3) unit\_size\_precision\_bytes\_minus1;  unsigned int(3) weight\_precision;  bit(2) reserved = 0;  unsigned int((unit\_size\_precision\_bytes\_minus1 + 1)\*8) config\_unit\_length;  bit(config\_unit\_length \* 8) config\_unit;  } |

The semantics for the fields defined in AvatarAnimationConfigurationBox are as follows:

unit\_size\_precision\_minus1 indicates the length in bytes of the AAUnitLength field in an animation sample of the associated stream minus one. For example, a size of one byte is indicated with a value of 0. The value of this field shall be one of 0, 1, or 3 corresponding to a length encoded with 1, 2, or 4 bytes, respectively.

weight\_precision is the length in bytes of the weight values within each sample. The value of precision shall be greater than 0 and smaller or equal to 4.

config\_unit\_length indicates the size of the configuration AAU carried in this AvatarAnimationConfigurationBox.

config\_unit is an AAU of type AAU\_CONFIG (i.e., a configuration avatar animation unit), see subclause 7.3.2.2.

The following requirements shall be fulfilled for avatar animation tracks:

* The handler type 'meta' shall be used in the HandlerBox of the MediaBox.
* Independent animation samples shall be marked as sync samples.

#### Avatar Animation Track Sample Format

The samples of an avatar animation track include avatar animation data. Each sample carries avatar animation data associated with a particular timestamp in the presentation timeline. An animation sample may contain one or more AAUs which belong to the same presentation time.

The format of each avatar animation sample of the track is defined as follow.

Table 30 — Syntax of AvatarAnimationSample

|  |
| --- |
| aligned(8) class AvatarAnimationSample {  // sample\_size size of sample from SampleSizeBox  for (int i=0; i < sample\_size; ) {  unsigned int((AvatarAnimationConfigurationRecord.unit\_size\_precision\_bytes\_minus1 + 1)\*8) AAUnitLength;  bit(AAUnitLength \* 8) AAUnit;  i += (AvatarAnimationConfigurationRecord.unit\_size\_precision\_bytes\_minus1 + 1) + AAUnitLength;  }  } |

The semantics of the fields defined in AvatarAnimationSample are as follows:

AAUnitLength is the size of the AAU measured in bytes. The length field includes the size of both the AAU header and the AAU payload but does not include the length field itself.

AAUnit contains a single AAU as defined in subclause 7.4, where the payload is based on the sample formats defined in clause 8.

An avatar animation sample may be designated as a sync sample. An avatar animation sync sample shall satisfy all the following conditions:

* It shall be possible to independently process the sample.
* None of the samples that come after the sync sample have any processing dependency on any sample prior to the sync sample.
* All samples that come after the sync sample can be successfully processed.

Samples may be grouped to indicate a sequence of associated animation codes that are stored and ready for playback. The sample group shall be signalled using the group type 'aasq'. Each animation sample group shall have a description about the pre-stored animation sequence, e.g. "smile", "dance".

The sample format for an animation sample is defined in clause 8.

#### Association of Assets with Avatar Animation Tracks

The set of avatar animation tracks associated with an asset shall be grouped with the asset using an AvatarAssetAnimationGroupBox defined as follows.

Table 31 — Syntax of AvatarAssetAnimationGroupBox

|  |
| --- |
| aligned(8) class AvatarAssetAnimationGroupBox()  extends EntityToGroupBox('avag', version=0, flags) {  unsigned int(8) avatar\_id;  unsigned int(8) asset\_id;  unsigned int(4) level\_of\_detail; } |

The semantics of the fields of AvatarAssetAnimationGroupBox are as follows:

avatar\_id is the unique identifier for the avatar associated with this entity group.

asset\_id is the unique identifier for the avatar asset that this entity group is associated with.

level\_of\_detail indicates the level of detail of the asset to which the component is associated.

# Zip-based container format

### Overview

An alternative to the ISOBMFF-based container format is the zip-based container format. A Zip container shall be formatted according to ISO/IEC 21320-1. All components of the base avatar model shall be included in the Zip file. The references to these components shall be relative to the location of the ARF document. The ARF document shall be in the root folder of the Zip container and shall be named arf.json.

If present, animation sequences shall be stored as individual binary files with file extension ".bin" under a folder named "animations". The format of each of the animation streams is described in clause 7.3.2.

### Avatar Animation Stream

#### Introduction

An avatar animation stream is composed of a sequence of avatar animation units (AAUs).

The general syntax structure for an AAU is shown in , where the data types used for the definition of different fields in the syntax structures are as follows.

* **uimsbf**: Unsigned integer with most significant bit first.
* **vlc8**: Variable length character string. Contains string data stored as a character array encoded in UTF-8.
* **boolean**: A single bit that represents a Boolean value.
* **float32**: A 32-bit floating point value represented according to the IEEE 754 specification.

Each avatar animation unit (AAU) contains a header and a payload. An AAU header contains at least a field that indicates the unit type and a field that indicates the AAU payload. The contents of the payload depend on the type of the AAU, where ByteAlignment is a padding with up to seven bits set to 0 for the AAU payload to be byte-aligned.

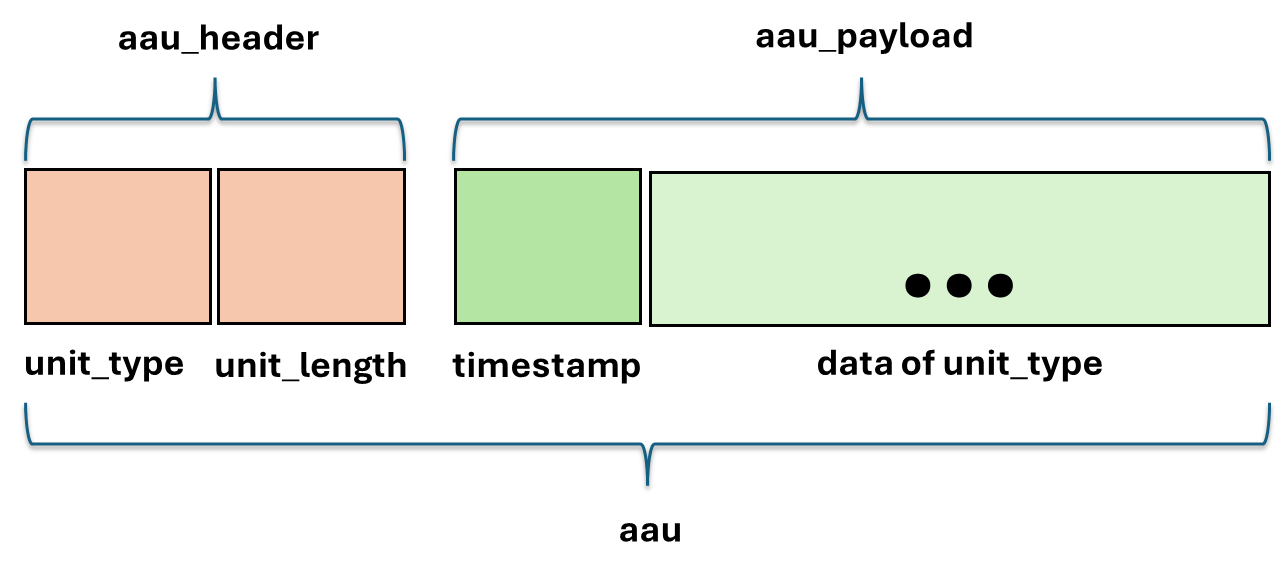


Figure 3 — Illustration of the non-compressed binary structure using AAUs.

Table 32 — Syntax of avatar\_animation\_unit()

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| avatar\_animation\_unit() |  |  |
| { |  |  |
| aau\_header(); |  |  |
| aau\_payload(); |  |  |
| ByteAlignement | 0-7 | uimsbf |
| } |  |  |

The avatar\_animation\_unit() syntax construct contains the following syntax elements:

* ByteAlignment: is a padding with up to seven bits set to 0 for the AAU payload to be byte-aligned.

The syntax structure of the AAU header is as shown in Table 33.

Table 33 — Syntax of aau\_header().

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| aau\_header() |  |  |
| { |  |  |
| aau\_unit\_type; | 7 | uimsbf |
| aau\_unit\_length; | 32 | uimsbf |
| reserved | 1 | uimsbf |
| } |  |  |

The aau\_header() syntax construct contains the following syntax elements:

* aau\_unit\_type: indicates the type of the AAU. The possible values are described in Table 34 .
* aau\_unit\_length: indicates the size of the AAU payload in bytes.

Table 34 — Avatar Animation Unit type codes and corresponding payloads.

|  |  |  |
| --- | --- | --- |
| aau\_unit\_type | Name of AAU type | Content of AAU payload |
| 0 | AAU\_CONFIG | aau\_config\_unit\_payload() |
| 1 | AAU\_BLENDSHAPE | aau\_blendshape\_unit\_payload() |
| 2 | AAU\_JOINT | aau\_joint\_unit\_payload() |
| 3 | AAU\_LANDMARK | aau\_landmark\_unit\_payload() |
| 4..10 | AAU\_RSV\_4 AAU\_RSV\_10 | Reserved AAU types. |
| 11..127 | AAU\_UNSPEC\_11 AAU\_UNSPEC\_127 | Unspecified AAU types. |

The aau\_payload() is defined as shown in Table 35.

Table 35 — Syntax of aau\_payload().

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| aau\_payload() |  |  |
| { |  |  |
| aau\_timestamp; | 32 | uimsbf |
| if (aau\_unit\_type == AAU\_CONFIG) |  |  |
| aau\_config\_unit\_payload(); |  |  |
| else if (aau\_unit\_type == AAU\_BLENDSHAPE) |  |  |
| aau\_blendshape\_unit\_payload(); |  |  |
| else if (aau\_unit\_type == AAU\_JOINT) |  |  |
| aau\_joint\_unit\_payload(); |  |  |
| else if (aau\_unit\_type == AAU\_LANDMARK) |  |  |
| aau\_landmark\_unit\_payload(); |  |  |
| } |  |  |

The aau\_payload() syntax construct contains the following syntax elements:

* aau\_timestamp: is the timestamp of the AAU in ticks. The timestamp in seconds can be calculated as timestamp/timescale, where timescale is signalled in the configuration AAU.

#### Configuration Unit

A configuration unit is an AAU with aau\_unit\_type set to AAU\_CONFIG. The payload of such AAU is defined as shown in Table 36.

Table 36 — Syntax of aau\_config\_unit\_payload().

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| aau\_config\_unit\_payload() |  |  |
| { |  |  |
| acu\_profile\_length; | 8 | uimsbf |
| acu\_animation\_profile; | acu\_profile\_length \* 8 | vlc8 |
| acu\_timescale; | 32 | float32 |
| } |  |  |

The aau\_config\_unit\_payload() syntax construct contains the following syntax elements:

* acu\_profile\_length: is the number of characters in the profile string signalled by acu\_animation\_profile.
* acu\_animation\_profile: is a character string with the name of the profile that generated stream conforms to.
* acu\_timescale: is the number of ticks per second.

#### Blendshape Unit

A blendshape unit is an AAU whose aau\_unit\_type field is set to AAU\_BLENDSHAPE. The payload of such AAU is defined as shown in Table 37.

Table 37 — Syntax of aau\_blendshape\_unit\_payload().

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| aau\_blendshape\_unit\_payload() |  |  |
| { |  |  |
| afa\_facial\_animation\_sample(); |  |  |
| } |  |  |

#### Joint Unit

A joint unit is an AAU whose aau\_unit\_type field is set to AAU\_JOINT. The payload of such AAU is defined as shown in Table 38.

Table 38 — Syntax of aau\_joint\_unit\_payload().

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| aau\_joint\_unit\_payload() |  |  |
| { |  |  |
| aja\_joint\_animation\_sample(); |  |  |
| } |  |  |

#### Landmark Unit

A landmark unit is an AAU whose aau\_unit\_type field is set to AAU\_LANDMARK. The payload of such AAU is defined as shown in Table 39.

Table 39 — Syntax of aau\_landmark\_unit\_payload().

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| aau\_landmark\_unit\_payload() |  |  |
| { |  |  |
| ala\_landmark\_animation\_sample(); |  |  |
| } |  |  |

# Animation Stream Format

## General

This version of the specification supports face, body, and hand animation. Facial animation is supported through weighted blend shapes. Body and hand animations are performed through Linear Blend Skinning (LBS).

LBS is a technique that is used in 3D animation to deform a mesh, usually a humanoid character, based on the positions of its joints. Each vertex in the mesh is assigned weights associated with a subset of the body joints. When a joint moves, the skin vertices associated with it move with it, each proportionally to the assigned weight for that joint. This creates a smooth and realistic-looking animation of the character. For every vertex, the weights assigned to the joints that impact its position should add up to 1.0 or a value very close to it, to avoid artifacts in the animation.

The position of a vertex *i* is determined using the set of bone transformations and their associated weights as described by the following equation:

where M is the global transformation matrix for bone *j*, which is the cumulative product of the transformation matrices of all parent joints as well as the inverse bind matrix of bone *j*.

Facial blend shapes are a method to animate a character’s face, where facial expressions and deformations need to be captured with precision. A set of versions of the 3D mesh of the face/head is used, where each version represents a different facial expression (blend shape). By adjusting the weights that control the influence of each blend shape, the desired facial expression can be achieved.

The following figure depicts an example of applying a “smile” facial expression at different weights:

A collage of a person's face

AI-generated content may be incorrect.  
Figure 4 Blend shape weight animation

Different facial expressions can be combined to render a mixed expression according to the following formula:

In this equation, *v*0 represents the position of the vertex in the base mesh, which is the mesh at the neutral expression.

The following sections define the formats for the blend shape and joint animation stream sample formats. A stream is a timed sequence of animation samples, which are formatted according to these specified formats.

## Facial Animation Sample Format

The facial animation sample shall follow the format specified in the following table:

Table 40 — Syntax of afa\_facial\_animation\_sample

|  |  |  |
| --- | --- | --- |
| afa\_facial\_animation\_sample() { | No. of bits | Mnemonic |
| afa\_blendshape\_set\_id | 16 | uimsbf |
| afa\_confidence\_present | 1 | boolean |
| reserved | 7 | uimsbf |
| afa\_blendshape\_count\_minus1 | 16 | uimsbf |
| for(i=0;i<=afa\_blendshape\_count\_minus1;i++) { |  |  |
| afa\_blendshape\_index[i] | 16 | uimsbf |
| afa\_weight[i] | 32 | float32 |
| if (afa\_confidence\_present) { |  |  |
| afa\_confidence[i] | 32 | float32 |
| } |  |  |
| } |  |  |
| } |  |  |

 The semantics of the fields defined in the sample are as follows:

* afa\_blendshape\_set\_id is the identifier of the blendshape set to which the animation samples apply.
* afa\_confidence\_present is a flag indicating whether confidence information is present for each signalled weight in the sample.
* afa\_blendshape\_count\_minus1 plus 1 indicates the number of blendshapes whose weights are signalled in the sample.
* afa\_blendshape\_index[i] is the index of the i-th blendshape whose weight is signalled in the sample.
* afa\_ weight[i] is the weight of the i-th blendshape whose index is signalled by the field afa\_blendshape\_id[i].
* afa\_confidence[i] is the confidence value associated with the weight signalled for the i-th animation target.

## Joint Animation Sample Format

The joint animation sample shall follow the format specified in the following table:

Table 41 — Syntax of aja\_joint\_animation\_sample

|  |  |  |
| --- | --- | --- |
| aja\_joint\_animation\_sample() { | No. of bits | Mnemonic |
| aja\_joint\_set\_id | 16 | uimsbf |
| aja\_velocity\_present | int(1) | boolean |
| reserved | int(7) |  |
| aja\_oint\_count\_minus1 | 16 | uimsbf |
| for(i=0;i<=aja\_joint\_count\_minus1;i++) { |  |  |
| aja\_target\_joint\_index[i] | 16 | uimsbf |
| aja\_joint\_transform[i] | 16 \*32 | float32 |
| if (aja\_velocity\_present) { |  |  |
| aja\_joint\_velocity[i] | 16\*32 | float32 |
| } |  |  |
| } |  |  |
| } |  |  |

 The semantics of the fields defined in the sample are as follows:

* aja\_joint\_set\_index indicates the target joint set index.
* aja\_velocity\_present is a flag indicating whether velocity information is present for each singnalled joint transform in the sample.
* aja\_joint\_count\_minus1 plus 1 indicates the number of joint transformations signalled in the sample.
* aja\_target\_joint\_index[i] indicates the target joint index for the i-th joint signalled in the sample.
* aja\_joint\_transform[i] is the transformation matrix for the target whose index is signalled by the field aju\_target\_joint\_index[i].
* aja\_joint\_velocity[i] is the velocity associated with the joint transformation.

## Landmark animation sample format

The landmark animation sample shall follow the format specified in the following table.

**Table 42** — **Landmark animation sample format.**

|  |  |  |
| --- | --- | --- |
| ala\_landmark\_animation\_sample() { | No. of bits | Mnemonic |
| ala\_landmark\_set\_id; | 16 | uimsbf |
| ala\_velocity\_present; | 1 | boolean |
| ala\_confidence\_present; | 1 | boolean |
| ala\_is\_3d\_flag; | 1 | boolean |
| reserved; | 5 | uimsbf |
| ala\_landmark\_count\_minus1; | 16 | uimsbf |
| for (i=0;i<=ala\_landmark\_count\_minus1;i++) { |  |  |
| ala\_target\_landmark\_index[i] | 16 | uimsbf |
| if (ala\_is\_3d\_flag) { |  |  |
| ala\_landmark\_coordinates; | 3\*32 | float32 |
| } else { |  |  |
| ala\_landmark\_coordinates; | 2\*32 | float32 |
| } |  |  |
| if (ala\_velocity\_present) { |  |  |
| ala\_velocity[i] | 32 | float32 |
| } |  |  |
| if (ala\_confidence\_present) { |  |  |
| ala\_confidence[i] | 32 | float32 |
| } |  |  |
| } |  |  |
| } |  |  |

The semantics of the fields defined in the sample are as follows:

* ala\_landmark\_set\_index indicates the target landmark set index.
* ala\_confidence\_present is a flag indicating whether confidence information is signalled for each landmark transform in the sample.
* ala\_velocity\_present is a flag indicating whether velocity information is signalled for each landmark transform in the sample.
* ala\_landmark\_count\_minus1 plus 1 indicate the number of landmark transformations signalled in the sample.
* ala\_target\_landmark\_index[i] indicates the target landmark index for the i-th landmark signalled in the sample.
* ala\_landmark\_coordinates[i] is a vector of 2D or 3D coordinates that provides the tracked coordinates of the target landmark vertex/point with index is aja\_target\_ landmark\_index[i].
* ala\_confidence[i] is the confidence value associated with the i-th landmark transform signalled in the sample.
* ala\_velocity[i] is the velocity associated with the i-th landmark transform signalled in the sample.

1. (normative)  
     
   ARF Document JSON Schema

The following table contains the JSON Schema for the ARF document.

|  |
| --- |
| {    "$schema": "http://json-schema.org/draft-07/schema#",    "type": "object",    "title": "ARF Container Schema",    "required": [      "preamble",      "metadata",      "structure",      "components",      "data"    ],    "properties": {      "preamble": {        "$ref": "arf-preamble.schema.json",        "description": "Contains data that uniquely the format and characteristics of the ARF container"      },      "metadata": {        "$ref": "arf-metadata.schema.json",        "description": "Contains metadata related to the base avatar model"      },      "structure": {        "$ref": "arf-structure.schema.json",        "description": "Contains the data structures of the ARF container"      },      "components": {        "$ref": "arf-components.schema.json",        "description": "Contains the core elements of the base avatar model. It lists the main ARF containers to represent and animate the base avatar"      },      "data": {        "$ref": "arf-data.schema.json",        "description": "Contains the data for each element of the 'components' ARF container"      }    }  } |

The schema for Preamble is provided in the following table:

|  |
| --- |
| {      "$schema": "http://json-schema.org/draft-07/schema#",      "type": "object",      "title": "Preamble Schema",      "required": ["signature", "version", "supportedAnimations"],      "properties": {        "signature": {          "type": "string",          "description": "Uniquely identifies the ARF"        },        "version": {          "type": "string",          "description": "Specifies the version of the MPEG Avatar Representation Format"        },        "authenticationFeatures": {          "type": "array",          "items": {            "$ref": "#/components/schemas/AuthenticationFeatures"          },          "description": "An array of features that are used to identify the owner of this base avatar"        },        "supportedAnimations": {          "$ref": "#/components/schemas/SupportedAnimations"        }      },      "components": {        "schemas": {          "AuthenticationFeatures": {            "type": "object",            "required": ["publicKey"],            "properties": {              "publicKey": {                "type": "string",                "format": "uri",                "description": "A URL to the public key that is used to decrypt the features"              },              "facialFeature": {                "type": "string",                "description": "A base64 encoded feature vector of floats. This can be used to match extracted facial features during a communication session. The facial feature shall be encoded with the user's private key to preserve authenticity"              },              "voiceFeature": {                "type": "string",                "description": "A base64 encoded feature vector of floats. This can be used to match extracted voice features during a communication session. The voice feature shall be encoded with the user's private key to preserve authenticity"              }            }          },          "SupportedAnimations": {            "type": "object",            "properties": {              "faceAnimations": {                "type": "array",                "items": {                  "type": "string",                  "format": "uri"                },                "description": "Lists the supported face animation types. Each item in the array is a string representing a supported face animation type. Each identifier should be formatted as a URN that includes an identifier of the framework, followed by an identifier of the facial blendshape set"              },              "bodyAnimations": {                "type": "array",                "items": {                  "type": "string",                  "format": "uri"                },                "description": "Lists the supported body animation types. Each item in the array is a string representing a supported body animation type. Each identifier should be formatted as a URN that includes an identifier of the body animation/tracking framework, followed by an identifier of the body joint set"              },              "handAnimations": {                "type": "array",                "items": {                  "type": "string",                  "format": "uri"                },                "description": "Lists the supported hand animation types. Each item in the array is a string representing a supported hand animation type. Each identifier should be formatted as a URN that includes an identifier of the body animation/tracking framework, followed by an identifier of the body joint set"              },              "landmarkAnimations": {                "type": "array",                "items": {                  "type": "string",                  "format": "uri"                },                "description": "Lists the supported landmark animation types. Each item in the array is a string representing a supported landmark animation type. Each identifier should be formatted as a URN that includes an identifier of the landmark animation/tracking framework, followed by an identifier of the landmark set"              },              "proprietaryAnimations": {                "type": "array",                "items": {                  "$ref": "#/components/schemas/ProprietaryAnimation"                },                "description": "A list of proprietary animation descriptions, which may be used to animate assets in the ARF container"              }            }          }        }      }    } |

The schema for the Metadata object is provided in the following table:

|  |
| --- |
| {      "$schema": "http://json-schema.org/draft-07/schema#",      "type": "object",      "title": "Metadata Schema",      "required": ["name", "id", "age", "gender"],      "properties": {        "name": {          "type": "string",          "description": "A string that describes the name of the avatar"        },        "id": {          "type": "string",          "description": "A string that uniquely identifies the avatar"        },        "age": {          "type": "integer",          "description": "An integer value to define the age of the avatar"        },        "gender": {          "type": "string",          "description": "A string that describes the gender of the avatar"        }      }  } |

The schema for Structure is provided in the following table:

|  |
| --- |
| {      "$schema": "http://json-schema.org/draft-07/schema#",      "type": "object",      "title": "Structure Schema",      "required": ["assets"],      "properties": {        "assets": {          "type": "array",          "items": {            "$ref": "#/components/schemas/Asset"          },          "description": "List the assets included in the ARF container"        },        "protectionConfigurations": {          "type": "array",          "items": {            "$ref": "#/components/schemas/ProtectionConfiguration"          },          "description": "A list of protection configuration objects that are used for the protection of components of the ARF container"        }      },      "components": {        "schemas": {          "Asset": {            "type": "object",            "required": ["name", "lods"],            "properties": {              "name": {                "type": "string",                "description": "The name of the asset"              },              "lods": {                "type": "array",                "items": {                  "$ref": "#/components/schemas/LOD"                },                "description": "A list of level of details available for this asset in the ARF container"              }            }          },          "LOD": {            "type": "object",            "required": ["name", "meshes"],            "properties": {              "name": {                "type": "string",                "description": "The name of the LOD"              },              "skins": {                "type": "array",                "items": {                  "type": "number"                },                "description": "List of references to all skins that are part of this asset"              },              "meshes": {                "type": "array",                "items": {                  "type": "number"                },                "description": "List of non-skinned meshes that are part of this asset"              },              "skeletons": {                "type": "array",                "items": {                  "type": "number"                },                "description": "List of references to skeletons in the ARF container"              },              "blendshapeSets": {                "type": "array",                "items": {                  "type": "number"                },                "description": "List of references to blend shape sets in the ARF container"              },              "landmarkSets": {                "type": "array",                "items": {                  "type": "number"                },                "description": "A list of references to landmark sets in the ARF container"              }            }          },          "ProtectionConfiguration": {            "type": "object",            "required": ["schemeIdUri"],            "properties": {              "schemeIdUri": {                "type": "string",                "description": "identifies a protection or encryption scheme"              },              "value": {                "type": "object",                "description": "Provides additional information specific to the protection or encryption scheme. For example, it may provide information such as DRM version, encryption mode, etc. The contents of this object are proprietary to the protection scheme"              }            }          }        }      }    } |

The schema for Components is provided in the following table:

|  |
| --- |
| {      "$schema": "http://json-schema.org/draft-07/schema#",      "type": "object",      "title": "Components Schema",      "required": ["meshes"],      "properties": {        "skeletons": {          "type": "array",          "items": {            "$ref": "#/components/schemas/Skeleton"          },          "description": "A list of skeletons used to describe the avatar skeletal asset"        },        "skins": {          "type": "array",          "items": {            "$ref": "#/components/schemas/Skin"          },          "description": "A list of assets that are stored in this ARF container"        },        "meshes": {          "type": "array",          "items": {            "$ref": "#/components/schemas/Mesh"          },          "description": "A list of geometries used to describe the avatar asset"        },        "nodes": {          "type": "array",          "items": {            "$ref": "#/components/schemas/Node"          },          "description": "A list of nodes used to organize, merge and describe and transform the avatar components"        },        "blendshapeSets": {          "type": "array",          "items": {            "$ref": "#/components/schemas/BlendshapeSet"          },          "description": "A list of blend shape sets used to describe the blend shape-based animations"        },        "landmarkSets": {          "type": "array",          "items": {            "$ref": "#/components/schemas/LandmarkSet"          },          "description": "A list of landmark sets used to describe landmark-based animation"        },        "textureSets": {          "type": "array",          "items": {            "$ref": "#/components/schemas/TextureSet"          },          "description": "A list of texture sets used to describe the avatar textures"        }      },      "components": {        "schemas": {          "Skeleton": {            "type": "object",            "required": ["name", "root", "joints", "inverseBindMatrix"],            "properties": {              "name": {                "type": "string",                "description": "The name of the skeleton"              },              "root": {                "type": "number",                "description": "Reference to the root joint for the skeleton in the nodes collection"              },              "joints": {                "type": "array",                "items": {                  "type": "number"                },                "description": "List of references to the list of joints in node collection of the ARF container"              },              "inverseBindMatrix": {                "type": "number",                "description": "References an item in the data collection of the ARF container that contains the inverse bind matrices for the joints in the same order as the joints. The data should be an Nx16 tensor, where N is the number of joints in the skeleton. The tensor format is defined in Annex E"              },              "animationInfo": {                "type": "array",                "items": {                  "$ref": "#/components/schemas/AnimationLink"                },                "description": "Establishes a link to the supported animation and tracking frameworks that this skeleton animation can be used with"              }            }          },          "AnimationLink": {            "type": "object",            "required": ["type", "target"],            "properties": {              "type": {                "type": "string",                "description": "The type of the supported animation",                "enum": ["ANIMATION\_FACE", "ANIMATION\_BODY", "ANIMATION\_HAND", "ANIMATION\_LANDMARK"]              },              "target": {                "type": "number",                "description": "The index of the target animation framework in the associated supported animations list for which these mappings apply"              },              "mappings": {                "type": "array",                "description": "A list of Mapping objects associated with this target animation framework",                "items": {                  "$ref": "#/components/schemas/Mapping"                }              }            }          },          "Mapping": {            "type": "object",            "description": "Mapping object that defines how animation data is mapped from source to target frameworks",            "required": ["source", "associations"],            "properties": {              "source": {                "type": "number",                "description": "The index of the source animation framework, of which the animation data is mapped to the target animation framework, using the provided mapping table"              },              "associations": {                "type": "array",                "description": "An array of linear associations mapping a set of values from the source animation framework to one value of the target animation framework",                "items": {                  "$ref": "#/components/schemas/LinearAssociation"                }              }            }          },          "LinearAssociation": {            "type": "object",            "description": "Defines a linear association between source animation values and a target animation value",            "required": ["targetIndex", "sourceIndices", "weights"],            "properties": {              "targetIndex": {                "type": "number",                "description": "The index of the value to be produced by this linear association. For example, for a blend shape set, this would indicate the index of the blend shape"              },              "sourceIndices": {                "type": "array",                "description": "An array of indices of the values from the source animation framework referenced that contribute to the target index value",                "items": {                  "type": "number"                }              },              "weights": {                "type": "array",                "description": "The associated weights for the mapping of the contributing source animation value into the target animation value. The weights shall be provided in the same order as the contributing animation ids in sourceIndices",                "items": {                  "type": "number"                }              }            }          },          "TextureTarget": {            "type": "object",            "description": "Represents a texture target",            "required": ["name", "id", "texture"],            "properties": {              "name": {                "type": "string",                "description": "The name of the texture target"              },              "id": {                "type": "integer",                "description": "A unique identifier of the texture target"              },              "texture": {                "type": "number",                "description": "References an item in the data root component with a texture"              },              "texturePath": {                "type": "string",                "description": "Indicates where the texture can be found in the item referenced by 'texture'"              }            }          },          "Skin": {            "type": "object",            "required": ["name", "mapping", "skeleton", "mesh", "weights"],            "properties": {              "name": {                "type": "string",                "description": "The name of the skin"              },              "mapping": {                "type": "string",                "description": "this contains a path indicator that can be used to assign this skinned mesh to a particular node in the scene graph"              },              "skeleton": {                "type": "number",                "description": "a reference to the skeleton"              },              "mesh": {                "type": "number",                "description": "a reference to the mesh of the skin"              },              "weights": {                "type": "number",                "description": "reference to an item in the data collection that contains the weights. These weights correspond to the influence of a set of joint transformations on the mesh vertices positions. The weights is provided as an NxM-tensor, where N is the number of vertices and M is the number of joints. The tensor format is defined in Annex E"              },              "proprietaryAnimations": {                "type": "array",                "items": {                  "type": "number"                },                "description": "An array of references to proprietaryAnimation objects that define a proprietary animation approach that applies to this skin"              }            }          },          "Mesh": {            "type": "object",            "required": ["name", "id", "path", "data"],            "properties": {              "name": {                "type": "string",                "description": "The name of the mesh"              },              "id": {                "type": "number",                "description": "The identifier of the mesh"              },              "path": {                "type": "string",                "description": "A string that represents a hierarchical path that can be used to associate the mesh with a node in the external scene graph e.g., \"full\_body/upper\_body/head\""              },              "data": {                "type": "array",                "items": {                  "type": "number"                },                "description": "A reference into a data item that contains the mesh data"              }            }          },          "BlendshapeSet": {            "type": "object",            "required": ["name", "id", "shapes", "baseMesh"],            "properties": {              "name": {                "type": "string",                "description": "The name of the blendshape set"              },              "id": {                "type": "number",                "description": "A unique identifier of the blendshape set. This id is used in the facial animation to associate the weights with the shapes"              },              "animationInfo": {                "type": "array",                "items": {                  "$ref": "#/components/schemas/AnimationLink"                },                "description": "Establishes a link to the supported animation and tracking frameworks that this belnd shape set can be used with"              },              "shapes": {                "type": "array",                "items": {                  "type": "number"                },                "description": "An array of references to data items that contain each blendshape's data"              },              "baseMesh": {                "type": "number",                "description": "A reference to a data item that contains the base mesh for this blend shape set"              }            }          },          "LandmarkSet": {              "type": "object",              "required": ["name", "id", "baseMesh", "vertices"],              "properties": {                "name": {                  "type": "string",                  "description": "The name of the landmark set"                },                "id": {                  "type": "number",                  "description": "A unique identifier of the landmark set. This id is used in the facial animation to associate the landmark vertices positions with the landmark vertices"                },                "animationInfo": {                  "type": "array",                  "items": {                    "$ref": "#/components/schemas/AnimationLink"                  },                  "description": "Establishes a link to the supported animation and tracking frameworks that this landmark set can be used with"                },                "baseMesh": {                  "type": "number",                  "description": "The base mesh that is associated with the landmark vertices"                },                "vertices": {                  "type": "number",                  "description": "A reference to the Data object that provides the list of vertex indices that make up the landmark set"                }              }          },          "TextureSet": {            "type": "object",            "description": "Represents a texture set component",            "required": ["name", "id", "animationInfo", "material", "materialPath", "targets"],            "properties": {              "name": {                "type": "string",                "description": "The name of the texture set"              },              "id": {                "type": "number",                "description": "A unique identifier of the texture set"              },              "animationInfo": {                "type": "array",                "description": "Establishes a link to the supported parametric texture frameworks that this texture set can be used with. For details refer to clause 6.5.8",                "items": {                  "$ref": "#/components/schemas/AnimationLink"                }              },              "material": {                "type": "number",                "description": "A reference to the Data object that provides the component with a material"              },              "materialPath": {                "type": "string",                "description": "Indicates where the texture can be found in the Data object referenced by 'material'"              },              "targets": {                "type": "array",                "description": "The list of texture targets. For details refer to clause 6.5.12",                "items": {                  "$ref": "#/components/schemas/TextureTarget"                }              }            }          },          "Node": {            "type": "object",            "required": ["name", "mapping", "transform"],            "properties": {              "name": {                "type": "string",                "description": "The name of the node"              },              "mapping": {                "type": "string",                "description": "The joint type or semantics e.g., \"full\_body/upper\_body/right\_arm\". The elements of the path hierarchy should follow the naming convention as defined in table 29 of 23090-14"              },              "parent": {                "type": "number",                "description": "If present, the identifier of the parent node of this node. This attribute shall be present for all nodes, except for the root"              },              "children": {                "type": "array",                "items": {                  "type": "number"                },                "description": "if present, a list of identifiers of the children nodes of this node"              },              "scale": {                "type": "array",                "items": {                  "type": "number"                },                "description": "The node's non-uniform scale, given as the scaling factors along the x,y and z axes"              },              "rotation": {                "type": "array",                "items": {                  "type": "number"                },                "description": "The node's unit quaternion rotation in the order (x,y,z,w), where w is the scalar"              },              "translation": {                "type": "array",                "items": {                  "type": "number"                },                "description": "The node's translation along the x,y and z axes"              },              "transform": {                "type": "array",                "items": {                  "type": "number"                },                "description": "Provides a 4x4 transformation matrix for the node to define its position and orientation"              }            }          }        }      }    } |

The Data object is defined in the following JSON schema:

|  |
| --- |
| {      "$schema": "http://json-schema.org/draft-07/schema#",      "type": "object",      "title": "Data Schema",      "required": ["name", "type", "uri"],      "properties": {        "name": {          "type": "string",          "description": "a string that defines the name of this data"        },        "type": {          "type": "string",          "description": "a string that provides the mime type of the data"        },        "uri": {          "type": "string",          "description": "a string that defines the data content or reference to the data content depending on type"        },        "offset": {          "type": "integer",          "minimum": 0,          "description": "defines the number of bytes used as offset into the data content as pointed to by uri"        },        "byteLength": {          "type": "integer",          "minimum": 0,          "description": "defines the number of bytes to use in data content"        },        "compression": {          "type": "string",          "description": "an identifier of the compressor used to compress this LoD representation of the mesh. The compressor shall be identified by a URN"        },        "protection": {          "type": "number",          "description": "an identifier of the protection configuration that is applied to encrypt this LoD representation of the mesh"        }      }    } |

The ProprietaryAnimation object has the following JSON schema:

|  |
| --- |
| {      "$schema": "http://json-schema.org/draft-07/schema#",      "type": "object",      "title": "Animation Schema",      "properties": {          "proprietary\_animation": {              "type": "object",              "description": "This object may provide information about an ML-based proprietary model for reconstruction and animation of the user's avatar",              "required": [                  "id",                  "scheme",                  "items"              ],              "properties": {                  "id": {                      "type": "number",                      "description": "A unique identifier of this proprietary animation scheme"                  },                  "scheme": {                      "type": "string",                      "format": "uri",                      "description": "A vendor-specific URN to identify the proprietary reconstruction and animation scheme"                  },                  "items": {                      "type": "array",                      "description": "A list of data item references, e.g. pretrained models or model weights, that are used by this proprietary reconstruction and animation scheme",                      "items": {                          "type": "number"                      }                  }              }          }      }  } |

1. (normative)  
   Integration into Scene Description

The Avatar Representation Format (ARF) is designed to work with the MPEG Scene Description solution based on glTF as defined in ISO/IEC 23090-14. However, ARF is not limited to MPEG SD but can theoretically be integrated into any scene description solution.

MPEG SD defines an MPEG\_node\_avatar extension that facilitates the integration of Avatars into the scene description. The MPEG\_node\_avatar is extended to provide for a more proper ARF integration.

The description of the MPEG\_Node\_avatar extension is modified as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Usage** | **Default** | **Description** |
| type | string | M |  | The type of the avatar representation is provided as a URN that uniquely identified the avatar representation scheme. The avatar representation scheme defines the format of all components that are used to reconstruct and animate the avatar. The reference MPEG avatar URN is defined in section 8.3.3.  The ARF avatar format shall set this field to “mpeg:avatar:arf:2025”. |
| mappings | array(Mapping) | M |  | The mapping between child nodes and their associated avatar path. Note that the corresponding path for a parent node shall be a prefix of the path of its child nodes. |
| extras | object | O |  | Contains format-specific parameters that are used to initialize the Avatar pipeline.  In this specification, the extras object shall contain the ARF-specific information as given below. |
| ARFContainer | URI | M | N/A | The URL to the ARF container that stores the base avatar model. |
| animationStreams | array(Object) | M | N/A | An array of objects that each describes an animation stream associated with the base avatar model in the ARF container. |
| type | enumeration | M | N/A | The type of the animation stream. In this version of the specification, it shall be either   * “ANIMATION\_BLENDSHAPES” or * “ANIMATION\_JOINTS”. |
| source | number | M | N/A | A pointer to the accessor that contains the animation data. |

1. Reference Avatar Client

The reference avatar client is depicted in Figure 5. The reference client architecture is based on the concepts defined in 23090-14, where an Avatar pipeline is part of a Media Access Function (MAF) and performs the Avatar reception and reconstruction. The Avatar pipeline fetches the ARF container and accesses the animation streams. It uses both to animate and reconstruct the Avatar. The reconstructed Avatar is then made available to the Presentation Engine for rendering through a set of buffers that contain the components of the Avatar’s reconstructed 3D mesh.



Figure 5 — Reference Avatar Client Model

1. Authentication Procedure (Informative)

## D.1 Introduction

This document outlines a procedure for an identity verification system, designed to mitigate the threat of deepfake impersonation in avatar-based communication platforms. The system aims to ensure that the individual offering an avatar is the legitimate owner of the associated base avatar model. This is achieved by analyzing and comparing facial features and potentially other biometric markers extracted from the user's live audio-visual input against those stored within a secure avatar container format.

The system comprises three core components as depicted by the following figure.

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 6 — Avatar feature verification

The Feature Extractor analyzes the user's 2D video and/or audio stream in real-time to extract distinctive facial and/or vocal features. The Identity Matching component then compares these extracted biometric features with the corresponding features stored within the user's avatar container. The comparison process utilizes algorithms designed to tolerate natural variations in appearance due to lighting, expression, and aging.

Finally, the Alert Receiver triggers an alert to the receiver in the event of a significant mismatch between the live and stored features, indicating a potential impersonation attempt.

The avatar container format serves as a secure repository for the user's biometric data. The user’s biometric features are encrypted using the user’s private key to ensure authenticity and allow all receivers to decode and extract these features using the user’s public key.

1. Tensor Data Format (Normative)
   1. Tensor Data Format

This section specifies the data type for dense tensors. Dense tensors are used extensively in the ARF format to describe different data elements, such as weights or inverse bind matrices for joints.

The dense data type represents a regular multi-dimensional array, where each component is of a specific data type.

The following table defines the syntax of the data item.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| num\_of\_dims | int32 | M | Provides the number of dimensions for the data tensor. |
| dims | int32 [num\_of\_dims] | M | A list of integers that define the dimension sizes of the tensor e.g., dimension of [2, 7, 4] refers to a tensor with 2 x 7 x 4 = 56 values, where the first element of the tensor has dimension 2, the second element has dimension 7 and the last element has dimension 4. |
| dtype | enum | M | A number that describes the exact data type of the data. The allowed data types correspond to the glTF 2.0 component types, as specified in glTF 2.0 [x] clause 5.1.3. |

* 1. MIME Type Registration

The MIME type for the tensor data as defined in this Annex shall be "application/mpeg.arf.dense".

* 1. Registration Form

Type name: application

Subtype name: mpeg.arf.dense

Required parameters:

Optional parameters:

Encoding considerations:

Security considerations:

Interoperability considerations:

Published specification: ISO/IEC 23090-39

Applications that use this media type: Avatar Communications

Fragment identifier considerations:

Additional information:

Deprecated alias names for this type:

Magic number(s):

File extension(s):

Macintosh file type code(s):

Person & email address to contact for further information:

Intended usage:

(One of COMMON, LIMITED USE, or OBSOLETE.)

Restrictions on usage:

(Any restrictions on where the media type can be used go here.)

Author:

Change controller:

Provisional registration? (standards tree only):

(Any other information that the author deems interesting may be

added below this line.)

1. Examples (Informative)

NOTE: Examples will be added in the next revision of the document.

Bibliography

1. ISO/IEC 12113:2022, Information technology – Runtime 3D asset delivery format – Khronos glTF 2.0