ISO/IEC AWI 23090-39(en)

First edition

Date: 2025-08-01

Information technology — Coded representation of immersive media —

Part 39:  
Avatar representation format

© 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

E-mail: copyright@iso.org

Website: [www.iso.org](http://www.iso.org)

Published in Switzerland

Contents

[Foreword](#224)

[Introduction](#225)

[Scope](#226)

[Normative references](#227)

[Terms and definitions](#228)

[Overview](#229)

[System Description](#230)

[Schemes](#231)

[Gerneral Conventions](#232)

[Data Model](#233)

[Avatar Representation Format Document](#234)

[General](#235)

[Preamble](#236)

[Metadata](#237)

[Structure](#238)

[Components](#239)

[Data](#240)

[ProtectionConfiguration](#241)

[ARF Container Format](#242)

[General](#243)

[ISOBMFF-based container format](#244)

[Zip-based container format](#245)

[Animation Stream Format](#246)

[General](#247)

[Facial Animation Sample Format](#248)

[Joint Animation Sample Format](#249)

[Landmark animation sample format](#250)

[Texture animation sample format](#251)

[(normative) ARF Document JSON Schema](#252)

[(normative) Integration into Scene Description](#253)

[(normative) Reference Avatar Client](#254)

[(informative) Authentication Procedure](#255)

[(normative) Tensor Data Format](#256)

[(informative) Examples](#257)

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

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 ISO 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](http://www.iso.org/directives)).

ISO draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO [had/had not] received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at [www.iso.org/patents](http://www.iso.org/patents).ISO shall not be held responsible for identifying any or all such patent rights.

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](http://www.iso.org/iso/foreword.html).

This document was prepared by Technical Committee [or Project Committee] , , [name of committee], Subcommittee SC .

This edition cancels and replaces the edition (ISO :), which has been technically revised.

The main changes are as follows:

A list of all parts in the ISO 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](http://www.iso.org/members.html).

Introduction

The introduction is an optional/conditional element of the text.

For rules on the drafting of the introduction, refer to the [ISO/IEC Directives, Part 2:2021, Clause 13](https://www.iso.org/sites/directives/current/part2/index.xhtml#_idTextAnchor169).

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.

Identification of patent holders: the following text shall be included if patent rights have been identified.

The International Organization for Standardization (ISO) [and/or] International Electrotechnical Commission (IEC) draw[s] attention to the fact that it is claimed that compliance with this document may involve the use of a patent.

ISO [and/or] IEC take[s] no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured ISO [and/or] IEC that he/she is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with ISO [and/or] IEC. Information may be obtained from the patent database available at [www.iso.org/patents](http://www.iso.org/patents).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those in the patent database. ISO [and/or] IEC shall not be held responsible for identifying any or all such patent rights.

Information technology — Coded representation of immersive media —

Part 39:  
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. It defines the structure, components, and metadata necessary to represent and animate avatar models accurately and consistently across various systems and platforms, ensuring interoperability and efficient use in immersive media applications.

Moreover, the document includes the specifications for the ARF container formats, animation stream formats, and procedures for integrating ARF into existing MPEG Scene Descriptions. It further addresses data representation methodologies, including tensor data formats for handling complex avatar-related data, as well as security and authentication mechanisms to verify avatar authenticity and prevent impersonation.

# 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. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

**Packing order**: Tensors SHALL be encoded in row-major (C-contiguous) order. For a tensor with shape [D0, D1, …, DN-1], the last index varies fastest in memory. The linear offset of an element at index tuple (i0, i1, …, iN-1) is i0·(D1⋯DN-1) + i1·(D2⋯DN-1) + … + iN-2·DN-1 + iN-1 elements from the base. Unless explicitly stated, tensors are tightly packed with no padding between contiguous elements.

**Endianness**: All integer and IEEE-754 floating-point components in tensor payloads SHALL be encoded little-endian. This requirement ensures direct interoperability with glTF and common GPU runtimes.

ISO/IEC 14496-12, *Information technology — Coding of audio-visual objects — Part 12: ISO base media file format*

ISO/IEC 21320-1, *Information technology — Document Container File — Part 1: Core*

ISO/IEC 23090-14, *Information technology — Coded representation of immersive media — Part 14: Scene description*

IEEE 754, *IEEE Standard for Floating-Point Arithmetic*

IETF RFC 8259, *The JavaScript Object Notation (JSON) Data Interchange Format*

# Terms and definitions

*The Terms and definitions clause is a mandatory element of the text.*

*For rules on the drafting of the Terms and definitions, refer to the* [*ISO/IEC Directives, Part 2:2021, Clause 16*](https://www.iso.org/sites/directives/current/part2/index.xhtml#_idTextAnchor208)*.*

To insert a new terminological entry, go to the *Structure* tab and click on *Insert Term entry*.

For the purposes of this document, the *following terms and definitions / terms and definitions given in , as well as the following [delete what does not apply]* apply.

Abbreviated Terms

|  |  |
| --- | --- |
| 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 |

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

* IEC Electropedia: available at <http://www.electropedia.org/>
* ISO Online browsing platform: available at <http://www.iso.org/obp>

Animation Data

skeletal, blend shape set, and other animation-related information

Animation Streams

timed data used to animate the base avatar

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

Asset

independently accessible element of an avatar

Avatar

3D graphics-based representation of a user

Base avatar model

personalized and animatable 3D model of the user

Blend shape

displacements and/or variations of the base avatar model to express key-frame animations

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

# 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.

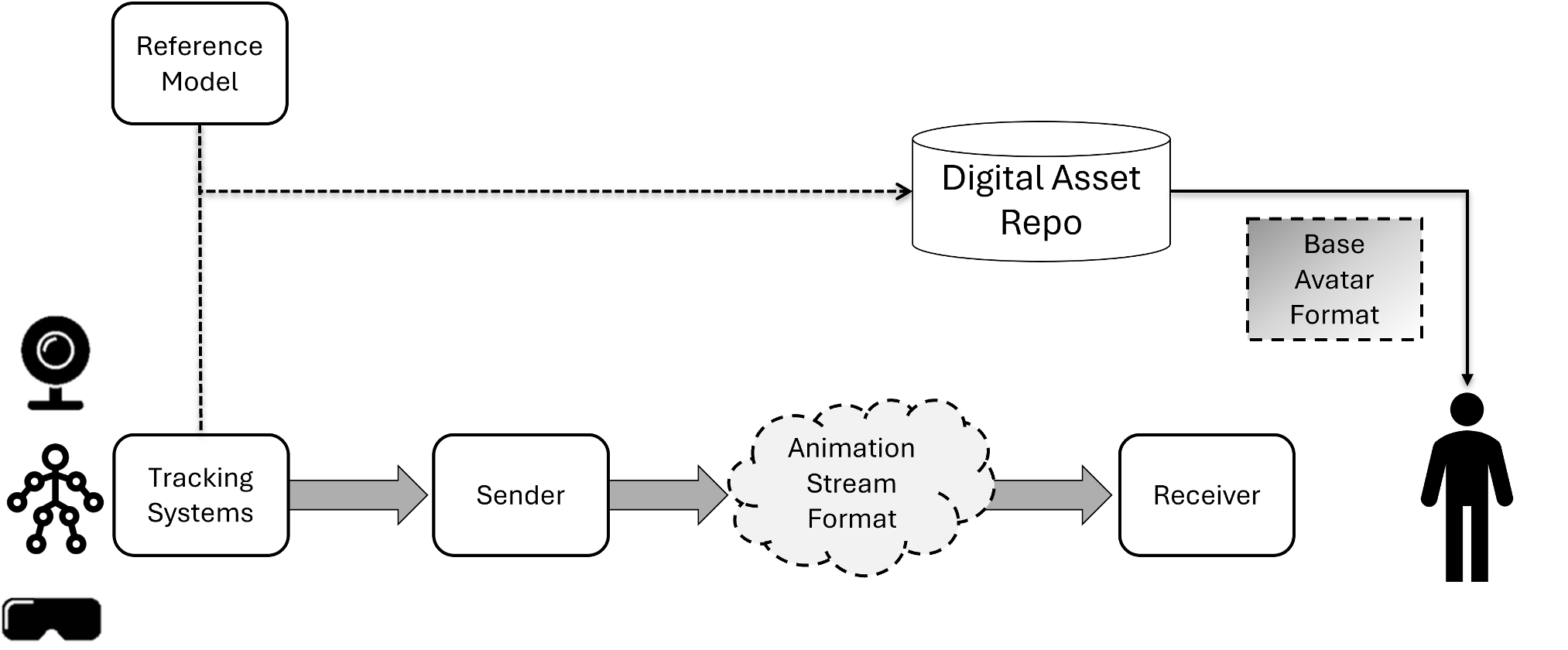


Figure 1 — Avatar reference architecture

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](#Section_id-88838f0e-b45e-422d-aa3b-5149e). A document describing the Base Avatar is defined in [Clause 6](#Section_id-44b16aac-83ba-4a40-eb96-8a825), 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.

## Schemes

This document specifies several schemes as listed in [Table 1](#Table_id-45d16007-c4a2-45e7-c354-16100a2).

Table 1 — 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 for animation streams. |

## General Conventions

ARF uses a right-handed coordinate system with the Y-axis oriented upwards (Y+) as depicted by Figure 2.

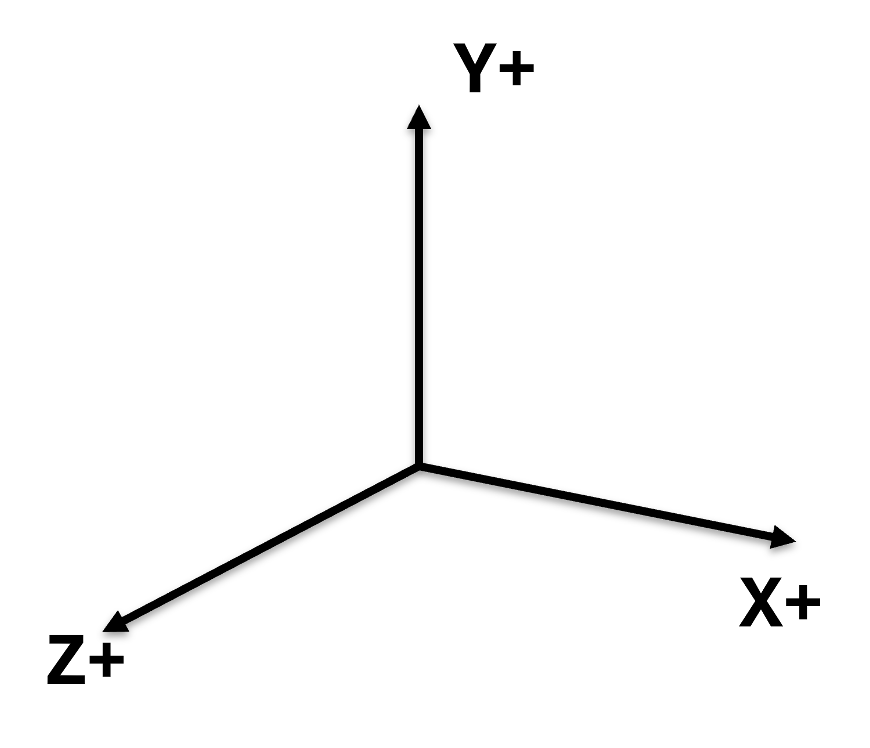


Figure 2 — ARF Coordinate System

ARF adopts the meter as the default unit of measurement. This ensures consistency and accuracy in spatial representation and interoperability across various platforms.

All references used in the ARF document are to the id field of the referred item. Index-based referencing is not used in this specification.

# Data Model

This clause defines a data model for the Base Avatar following the illustration in Figure 3.

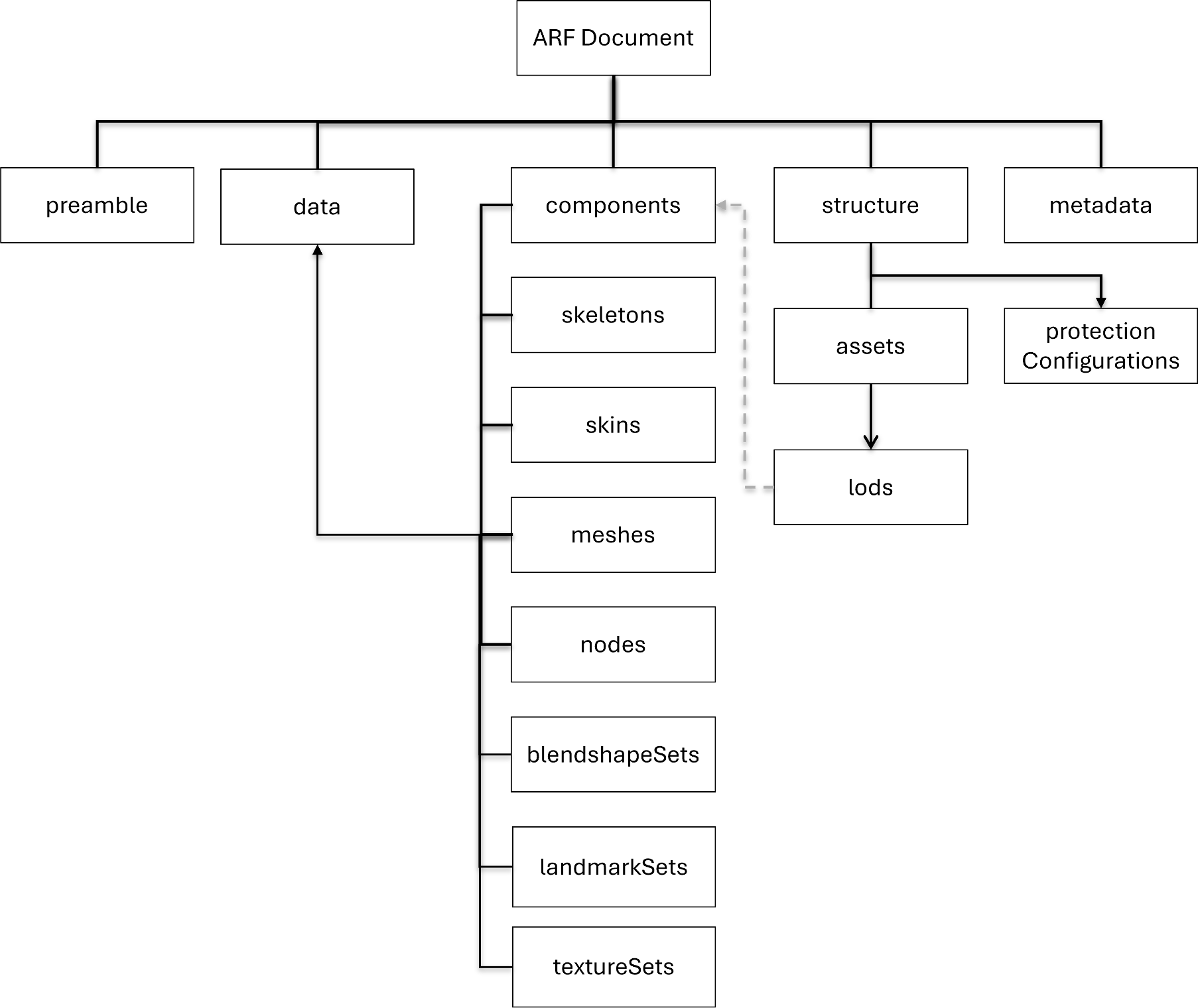


Figure 3 — ARF document structure

The description of each of these components is provided in [Clause 6](#Section_id-44b16aac-83ba-4a40-eb96-8a825) and [Clause 7](#Section_id-db0f826f-675b-40d4-c3bd-6d0ad). At the core, Skeleton objects define the hierarchical joint structures that serve as the foundational framework for avatar animation. Skin objects reference Skeletons and Meshes, enabling mesh deformation driven by joint movements. Mesh objects define geometric shapes and topology, and are referenced by Skins, BlendshapeSets, and LandmarkSets. BlendshapeSets represent facial animation states and are defined as modifications to base Mesh geometries. LandmarkSets similarly represent specific mesh vertices used for precise facial animations and alignments. TextureSets are related to Mesh and Skin objects, providing visual details via textures that enhance the avatar’s appearance. Together, these component types ensure accurate and consistent animation and rendering across multiple platforms.

# 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 2](#Table_id-4aa39633-c806-455d-aea9-13b9fcb) defines the high-level component objects of the ARF document.

Table 2 — 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](#Section_id-bada1936-dca9-440f-fe01-6e45e). |
| metadata | Metadata | M | specifies metadata related to the base avatar model.  For details refer to clause [6.3](#Section_id-77cac336-a88f-4574-9b0d-b030f). |
| structure | Structure | M | Contains the data structures of the ARF container.  For details refer to clause [6.4](#Section_id-289a126e-51c7-4874-80bc-f0b3b). |
| 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](#Section_id-cb63ace8-b338-4b79-da1e-91cdc). |
| data | array(Data) | M | Contains the data for each element of the components of the ARF container.  For details refer to clause [6.6](#Section_id-c4978f02-4a87-4873-c077-513ba). |
| **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 3](#Table_id-6c7bd03c-e9fb-4145-94cf-1df08bd) defines the Preamble object.

Table 3 — 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](#Section_id-b5ef53ac-fa74-4b02-b6dd-bba95). |
| supportedAnimations | SupportedAnimations | M | contains information about the supported animation types.  For more details refer to clause [6.2.3](#Section_id-e878d548-5678-46b0-d9ae-e9475). |
| **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 4](#Table_id-f1f991e0-a8b7-4950-9f0c-ad0c322) defines the AuthenticationFeatures object.

Table 4 — 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 5](#Table_id-13b8b36b-fdb2-4f91-8429-89ffd54) defines the SupportedAnimations object.

Table 5 — 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 hand 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. |
| textureAnimations | array(uri) | O | Lists the supported texture animation types. Each item in the array is a string representing a supported texture animation type.  Each identifier should be formatted as a URN that includes an identifier of the texture animation framework. |
| 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](#Section_id-7ff9d015-8e42-4329-b89e-175fb). |
| **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 a customized animation frameworks, such as Machine Learning (ML) models to reconstruct or animate assets in the ARF container.

[Table 6](#Table_id-7b4548cc-18c3-4454-910d-348cee8) defines the ProprietaryAnimation object.

Table 6 — 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 gender, age and height, as well as other metadata related to security and protection of the base avatar model.

[Table 7](#Table_id-bbc66456-2717-4f8f-ef0c-484a9d6) defines the Metadata object.

Table 7 — 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 8](#Table_id-dcf065c0-bc89-4a2e-c119-09c3bc9) defines the Structure object.

Table 8 — 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](#Section_id-85a9cb63-570c-47fb-b5f1-142e6). |
| protectionConfigurations | 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](#Section_id-47f9b500-1136-4040-d899-787e4). |
| **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 9](#Table_id-af687175-b076-4060-d7c0-db7b176) defines the Asset object.

Table 9 — Definition of Asset object

| **Name** | **Type** | **Use** | **Description** |
| --- | --- | --- | --- |
| name | string | M | The name of the asset. |
| id | number | M | A unique identifier for the asset. |
| lods | array(LOD) | M | A list of levels of detail available for this asset in the ARF container.  For details refer to clause [6.4.3](#Section_id-6fcbe607-8fbf-4278-f39b-3bee3). |
| **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 10](#Table_id-2478c03f-ae1b-4d30-810f-19b6628) defines the LOD object.

Table 10 — Definition of LOD object

| **Name** | **Type** | **Use** | **Description** |
| --- | --- | --- | --- |
| name | string | M | The name of the LOD. |
| id | number | M | A unique identifier of the LOD. |
| skins | array(number) | CM | List of references to all skins that are part of this asset. |
| meshes | array(number) | CM | List of references to 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 used by at least one of the skins of this asset. |
| landmarkSets | array(number) | O | A list of references to landmark sets used by at least one of the skins of this asset in the ARF container. |
| textureSets | array(number) | O | A list of references to texture sets used by at least one of the skins of this asset in the ARF container. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory.  At least one Skin object or one Mesh object should be present in the LoD of the asset. | | | |

## 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 11](#Table_id-9d802c36-463e-4c53-e83e-2046669) defines the Components object.

Table 11 — 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](#Section_id-67fde882-edfa-449a-fbb6-ae397). |
| skins | array(Skin) | O | A list of skinned meshes that are stored in this ARF container.  For details refer to clause [6.5.3](#Section_id-d9ddfa45-aaf5-4c78-962d-adecb). |
| meshes | array(Mesh) | M | A list of mesh objects that are used by skins and other components of avatar assets.  For details refer to clause [6.5.4](#Section_id-4fcd70a4-55fb-4786-aba3-3e315). |
| nodes | array(Node) | O | A list of nodes used to organize, merge, describe, and transform the avatar components.  For details refer to clause [6.5.9](#Section_id-10a82b84-d548-4dd9-9860-bf7ba). |
| 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](#Section_id-87db85c5-2b20-44f8-9aa7-22010). |
| landmarkSets | array(LandmarkSet) | O | A list of landmark sets used to describe landmark-based animation.  For details refer to clause [6.5.6](#Section_id-ee554b80-c0a9-4e47-c169-cd6d6). |
| textureSets | array(TextureSet) | O | A list of texture sets used to describe parametric textures.  For details refer to clause [6.5.7](#Section_id-b299e907-505f-4e5d-b86e-61430). |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory.  NOTE: additional component types may be added in the future. | | | |

### 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 12](#Table_id-7b675423-fe11-4586-820b-fe58120) defines the Skeleton object.

Table 12 — Definition of Skeleton object

| **Name** | **Type** | **Use** | **Description** |
| --- | --- | --- | --- |
| name | string |  | The name of the skeleton. |
| id | number |  | A unique identifier of this skeleton component. |
| root | number |  | Id of the Node that contains the root joint for the skeleton in the nodes collection. |
| joints | array(number) |  | List of ids of nodes that build a subset of child nodes of the root node and that make up the current skeleton together with the root node. |
| inverseBindMatrix | number |  | 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) |  | 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](#Section_id-f72e60a5-0f37-4e24-cd1c-e177a). |
| **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 13](#Table_id-2ff634da-d5f9-4216-8437-88e9e06) defines the Skin object.

Table 13 — Definition of Skin object

| **Name** | **Type** | **Use** | **Description** |
| --- | --- | --- | --- |
| name | string | M | The name of the skin. |
| id | number | M | A unique identifier of this Skin component. |
| mapping | string | O | this contains a path indicator that can be used to assign this skinned mesh to a particular node in the scene graph.  For example, the skin that contains the head of the avatar may provide the following mapping: "full\_body/upper\_body/head" to assign the skin as a child node of the avatar node in the scene graph with the provided hierarchy. |
| skeleton | number | O | a reference to the skeleton. |
| blendshapeSet |  |  | a reference to a blendshapeSet that is defined for this Skin object. If present, the baseMesh of the blendshapeSet shall be the same as the mesh for this Skin. |
| landmarkSet |  |  | a reference to a landmarkSet that is defined for this Skin object. |
| textureSet |  |  | a reference to a textureSet that is defined for this Skin object. |
| mesh | number | M | a reference to the mesh of the skin. |
| weights | number | CM | 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 are provided as an NxM-tensor, where N is the number of vertices and M is the number of joints in the referenced skeleton. 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 14](#Table_id-5d8c1565-dfc4-432c-b335-fbc16d3) defines the Mesh object.

Table 14 — Definition of Mesh object

| **Name** | **Type** | **Use** | **Description** |
| --- | --- | --- | --- |
| name | string | M | The name of the mesh. |
| id | number | M | A unique identifier of the Mesh object. |
| path | string | O | 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 list of references to data items that contain the mesh data. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### BlendshapeSet

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

[Table 15](#Table_id-b7ac6015-baea-42c8-f0d4-bb59f46) defines the BlendshapeSet object.

Table 15 — 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 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 blend shape set can be used with.  For details refer to clause [6.5.8](#Section_id-f72e60a5-0f37-4e24-cd1c-e177a). |
| shapes | array(number) | M | An array of ids of data items that contain each blendshape’s data. The shape keys are by default GLB files that only have geometry information (vertices and faces). They may optionally have other attributes such as normals. No materials or textures shall be included. Alternative representations may be possible and need to be identified through the MIME type. |
| baseMesh | number | M | A reference to a Mesh object that contains the base mesh for this blend shape set. Note that the topology of the baseMesh and the associated shapes shall be identical (i.e. same number of vertices and faces). |
| **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 16](#Table_id-6d2fbf7b-dc60-45eb-bc8f-7ff9f88) defines the LandmarkSet object.

Table 16 — 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](#Section_id-f72e60a5-0f37-4e24-cd1c-e177a). |
| baseMesh | number | M | The base mesh that is associated with the landmark vertices. |
| vertices | number | O | A reference to the Data object that provides the list of vertex indices that make up the landmark set. |
| faces | number | O | A reference to the Data object that provides the list of mesh face indices on which the landmarks in the landmark set are located. |
| weights | number | O | A reference to the Data object that provides a triplet of barycentric coordinate weights for each landmark in 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 textures of the material a linear combination of texture targets.

[Table 17](#Table_id-eea457ca-946b-4025-cd9c-8001274) defines the TextureSet object.

Table 17 — 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](#Section_id-f72e60a5-0f37-4e24-cd1c-e177a). |
| 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](#Section_id-7de1158d-73bf-4ce4-8b40-2a9fe). |
| **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 18](#Table_id-71e380b4-aba3-42e1-ed9c-5e46714) defines the AnimationLink object.

Table 18 — Definition of AnimationLink object

| **Name** | **Type** | **Use** | **Description** |
| --- | --- | --- | --- |
| type | enumeration | M | The type of supported animation. The allowed types are:   * ANIMATION\_FACE * ANIMATION\_BODY * ANIMATION\_HAND * ANIMATION\_LANDMARK * ANIMATION\_TEXTURE |
| 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](#Section_id-990b40e1-174c-4849-e75d-323b0). |
| **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 19](#Table_id-77c5262a-5509-456d-bd67-97100fd) defines the Node object.

Table 19 — Definition of Node object

| **Name** | **Type** | **Use** | **Description** |
| --- | --- | --- | --- |
| name | string | M | The name of the node. |
| id |  | M | A unique identifier of the Node object. |
| 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 ISO/IEC 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(number) | O | if present, a list of identifiers of the children’s 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) |  | The node’s translation along the x,y and z axes. |
| transform | array(number) |  | Provides a 4x4 transformation matrix for the node to define its position and orientation. This is an alternative representation of the TRS and should be used mutually. |
| **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 20](#Table_id-770331f6-6c08-4743-90bf-e68e68e).

Table 20 — 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. |
| linearAssociations | array(LinearAssociation) | CM | 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](#Section_id-990b40e1-174c-4849-e75d-323b0). |
| nonLinearAssociations | array(NonLinearAssociation) | CM | An array of non-linear associations mapping a set of values form the source animation framework to one value of the target animation framework. The NonLinearAssociation object is defined in clause 6.5.12. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory.  At least a linearAssociations or nonLinearAssociations must be present. | | | |

### 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 [Table 21](#Table_id-14023da1-b507-4b1e-e2e7-2204a38).

Table 21 — 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:  (1)  The transform matrix of the target animation value with index targetIndex is calculated as follows for joint animations:  (2) |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### NonLinearAssociation

The **NonLinearAssociation** object defines a non-linear mapping between a set of values from the source animation framework to a value of the target animation framework.

The NonLinearAssociation object is defined in the [Table 22](#id-7de1158d-73bf-4ce4-8b40-2a9fe).

Table 22 — Definition of an NonLinearAssociation object

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| targetIndex | number | M | Provides the index of the value to be produced by this association. For example, for a blend shape set, this would indicate the index of the blend shape. |
| channels | array(Channel) | M | Defines an array of channels, each channel transforming the weights from a source animation framework. |
| combination | enum | M | Defines the method to combine output of each channel. For example, COMBINATION\_SUM and COMBINATION\_MUL. |
| **Legend**:  For Use: M=mandatory, O=optional, OD=optional with default value, CM=conditionally mandatory. | | | |

### Channel

The **Channel** component defines functions to compute the outputs using the weigths in the target animation framework modified by the **NonLinearAssociation** object.

The **Channel** object is defined in the [Table 23](#tab_channel).

Table 23 — Properties of a Channel

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Use | Description |
| sourceIndex | number | M | The index of the value from the source animation framework referenced that contributes to the target index value. |
| range | number[2] | O | Minimum and maximum values for the source weight. |
| path | string | O | Defines the element of the target to update. For example, CHANNEL\_BLENDSHAPE, CHANNEL\_JOINT, CHANNEL\_JOINT\_ROTATION, CHANNEL\_JOINT\_SCALE or CHANNEL\_JOINT\_TRANSLATION. |
| input | integer | M | The identifier of a data component with the input values. |
| interpolation | enum | O | The interpolation method, for example, INTERPOLATION\_LINEAR, INTERPOLATION\_STEP or INTERPOLATION\_CUBICSPLINE. The default is INTERPOLATION\_LINEAR. |
| output | integer | M | The identifier of a data component with the output values |

The path property values must be consistent with the samples defined by ***output***. The definition of each value are as follows:

* CHANNEL\_BLENDSHAPE: weight of a blendshape (single float).
* CHANNEL\_JOINT: transform of a joint (16 floats).
* CHANNEL\_TRANSLATION: translation of a joint (3 floats).
* CHANNEL\_ROTATION: rotation of a joint (quaternion, 4 floats).
* CHANNEL\_SCALE: scale of a joint (3 floats).

### TextureTarget

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

[Table 24](#id-1182ed7b-ca39-4e71-bdc0-84e603d)defines the TextureTarget object.

Table 24 — Definition of TextureTarget 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 25](#id-3e217e04-f175-4287-ef8d-ad1ce62) defines the Data object.

Table 25 — Definition of Data object

| **Name** | **Type** | **Use** | **Description** |
| --- | --- | --- | --- |
| name | string | M | a string that defines the name of this data. |
| id | number | M | a unique identifier of this Data item. |
| 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 26](#id-5ec10676-012b-477f-ba2d-5a647e0) defines the ProtectionConfiguration object.

Table 26 — 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](#Section_id-45e1f41a-9bc5-4c12-9af5-254ce) 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.

### 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 27](#id-aa4f3d08-98bf-4b1e-fda6-737744f).

Table 27 — Brands defined in this document

| **Brand identifier** | **Clause in this document** | **Informative description** |
| --- | --- | --- |
| 'ARF' | [7.2](#Section_id-a6e6bbc4-bb31-48c1-ab6e-d8ad4) | every ISOBMFF-based container shall declare ARF as the major brand. |
| 'maas' | [7.2](#Section_id-a6e6bbc4-bb31-48c1-ab6e-d8ad4) | Files that contain stored animation streams shall declare maas among their compatibility brands. |

### 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 in [Table 28](#id-f2c19335-8bd5-47c0-9af1-1629419).

Table 28 — 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, signaling 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 29](#id-9d0ed10d-76ab-4828-b2dc-84384a9), where AvatarAnimationConfigurationBox is defined in [Table 30](#id-306236d0-2104-4f1c-9a5d-92afb7e).

Table 29 — Syntax of AvatarAnimationSampleEntry

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

Table 30 — Syntax of AvatarAnimationConfigurationBox

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

The Avatar AnimationConfigurationRecord is defined in [Table 31](#id-45e1835c-a146-4fec-e583-13eb28c).

Table 31 — 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](#Section_id-c9388910-37e8-4984-c6f8-f8524).

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 in [Table 32](#id-900ada4a-7de6-45fd-ee4c-8f5a047).

Table 32 — 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 [7.3.2.1](#Section_id-ca2b7416-e7df-4c0e-cad1-7a34b), where the payload is based on the sample formats defined in [Clause 8](#Section_id-ca0f789a-f916-49ea-9de1-0fed4).

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 signaled 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 [7.3.2.1](#Section_id-ca2b7416-e7df-4c0e-cad1-7a34b).

#### 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 in [Table 33](#id-84256144-c4d6-4929-c030-dbdf4e0).

Table 33 — 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 7.4.

The ".arfz" file extension is defined for this container format. The MIME type for this container format shall be "model/vnd.mpeg.arf+zip".

## 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 [Table 34](#id-4ef25c84-65ac-4f1a-cfe5-150b9f0), 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.

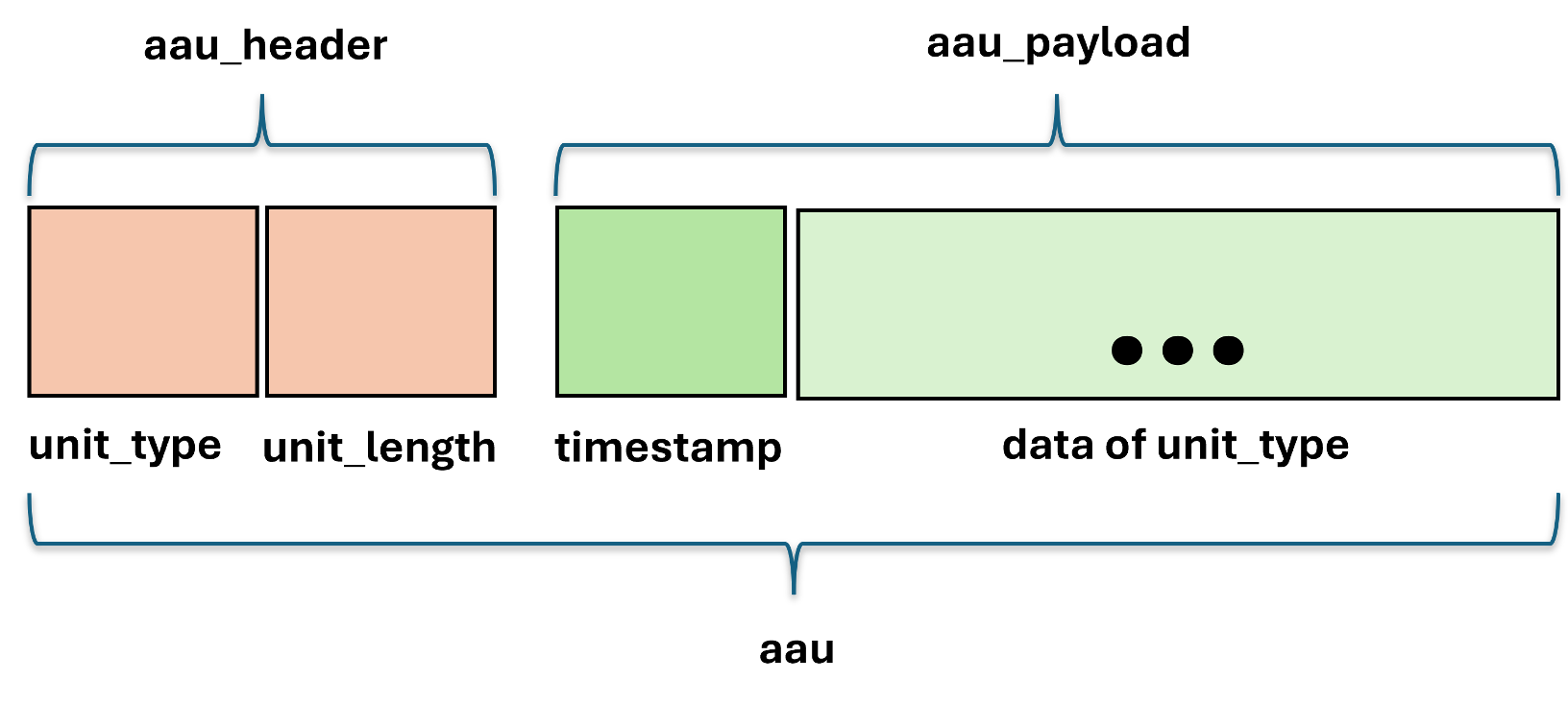


Figure 4 — Illustration of the non-compressed binary structure using AAUs.

Table 34 — 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 35](#id-41b795cf-eb83-4f7f-f332-ffc171e).

Table 35 — 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 36](#id-e78ef756-f8a2-4087-e6dd-c9be89e).
* ***aau\_unit\_length***: indicates the size of the AAU payload in bytes.

Table 36 — 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 | AAU\_TEXTURE | aau\_texture\_unit\_payload() |
| 5..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 37](#id-61d9d9ae-7f87-4a84-b68d-9be134f).

Table 37 — 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(); |  |  |
|  | else if (aau\_unit\_type == AAU\_TEXTURE) | |  |  |
|  |  | aau\_texture\_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 38](#id-15300a9d-ead0-4a45-e6e2-8ee9154).

Table 38 — 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 39](#id-5c855ec1-aad6-4f14-f75e-32c98e2).

Table 39 — 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 40](#id-d8790e08-0e1e-4d24-eff9-e76e035).

Table 40 — 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 41](#id-16bba28f-c477-418b-be90-d6cfef7).

Table 41 — Syntax of aau\_landmark\_unit\_payload()

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

#### Texture Unit

A texture unit is an AAU whose aau\_unit\_type field is set to AAU\_TEXTURE. The payload of such AAU is defined as shown in [Table 42](#id-bab6d6df-909d-4b73-b583-1f27a32).

Table 42 — Syntax of aau\_texture\_unit\_payload()

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

# Animation Stream Format

## General

This version of the specification supports face, body, hand, and texture 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 each 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:

(3)

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*. and are the new vertex position and its position in rest pose (T-pose) for vertex *i*, respectively*.*

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.

Figure 5 depicts an example of applying a “smile” facial expression at different weights:



Figure 5 — Blend shape weight animation

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

(4)

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 as described in clause [7.3.2](#Section_id-d89135c1-06cf-406f-9d3f-05707).

## Facial Animation Sample Format

The facial animation sample shall follow the format specified in [Table 43](#id-0615d40f-97c8-41eb-94d5-55219f4).

Table 43 — Syntax of afa\_facial\_animation\_sample

| **Syntax** | | | | **No. of bits** | **Mnemonic** |
| --- | --- | --- | --- | --- | --- |
| afa\_facial\_animation\_sample() { | | | |  |  |
|  | 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 [Table 44](#id-afd018bb-19ad-4518-8cc8-d42a970).

Table 44 — Syntax of aja\_joint\_animation\_sample

| **Syntax** | | | | **No. of bits** | **Mnemonic** |
| --- | --- | --- | --- | --- | --- |
| aja\_joint\_animation\_sample() { | | | |  |  |
|  | aja\_joint\_set\_id | | | 16 | uimsbf |
|  | aja\_velocity\_present | | | 1 | boolean |
|  | reserved | | |  |  |
|  | aja\_oint\_count\_minus1 | | | 7 | 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 singnaled 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 [Table 45](#id-14038ce0-db01-4ef9-c8f9-2397297).

Table 45 — Landmark animation sample format

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Syntax** | | | | **No. of bits** | **Mnemonic** |
| ala\_landmark\_animation\_sample() { | | | |  |  |
|  | 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; | 3\*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.

## Texture animation sample format

The texture animation sample shall follow the format specified in [Table 46](#id-f6c09a6a-1d9f-4cb0-d7d0-813ba21).

Table 46 — **Texture animation sample format**

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

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

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

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 hand 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"

                    },

                    "textureAnimations": {

                        "type": "array",

                        "items": {

                            "type": "string",

                            "format": "uri"

                        },

                        "description": "Lists the supported texture animation types. Each item in the array is a string representing a supported texture animation type. Each identifier should be formatted as a URN that includes an identifier of the texture animation framework"

                    },

                    "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"

                    }

                }

            },

            "ProprietaryAnimation": {

                "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"

                        }

                    }

                }

            }

        }

    }

}

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",

                    "id",

                    "lods"

                ],

                "properties": {

                    "name": {

                        "type": "string",

                        "description": "The name of the asset"

                    },

                    "id": {

                        "type": "number",

                        "description": "A unique identifier for 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",

                    "id"

                ],

                "oneOf": [

                    {

                        "required": [

                            "name",

                            "id",

                            "skins"

                        ]

                    },

                    {

                        "required": [

                            "name",

                            "id",

                            "meshes"

                        ]

                    }

                ],

                "properties": {

                    "name": {

                        "type": "string",

                        "description": "The name of the LOD"

                    },

                    "id": {

                        "type": "number",

                        "description": "A unique identifier 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 references to 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 used by at least one of the skins of this asset"

                    },

                    "landmarkSets": {

                        "type": "array",

                        "items": {

                            "type": "number"

                        },

                        "description": "A list of references to landmark sets used by at least one of the skins of this asset in the ARF container"

                    },

                    "textureSets": {

                        "type": "array",

                        "items": {

                            "type": "number"

                        },

                        "description": "A list of references to texture sets used by at least one of the skins of this asset 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 skinned meshes that are stored in this ARF container"

        },

        "meshes": {

            "type": "array",

            "items": {

                "$ref": "#/components/schemas/Mesh"

            },

            "description": "A list of mesh objects that are used by skins and other components of avatar assets"

        },

        "nodes": {

            "type": "array",

            "items": {

                "$ref": "#/components/schemas/Node"

            },

            "description": "A list of nodes used to organize, merge, 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 parametric textures"

        }

    },

    "components": {

        "schemas": {

            "Skeleton": {

                "type": "object",

                "required": [

                    "name",

                    "id",

                    "root",

                    "joints",

                    "inverseBindMatrix"

                ],

                "properties": {

                    "name": {

                        "type": "string",

                        "description": "The name of the skeleton"

                    },

                    "id": {

                        "type": "number",

                        "description": "A unique identifier of this skeleton component"

                    },

                    "root": {

                        "type": "number",

                        "description": "Id of the Node that contains the root joint for the skeleton in the nodes collection"

                    },

                    "joints": {

                        "type": "array",

                        "items": {

                            "type": "number"

                        },

                        "description": "List of ids of nodes that build a subset of child nodes of the root node and that make up the current skeleton together with the root node"

                    },

                    "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",

                            "ANIMATION\_TEXTURE"

                        ]

                    },

                    "target": {

                        "type": "number",

                        "description": "Provides the index of the target animation framework in the associated supported animations list for which these mappings apply"

                    },

                    "mappings": {

                        "type": "array",

                        "description": "Provides a list of Mapping objects associated with this target animation framework",

                        "items": {

                            "$ref": "#/components/schemas/Mapping"

                        }

                    }

                }

            },

            "Mapping": {

                "type": "object",

                "description": "Provides a way to signal mappings between source animation frameworks and the parent target animation framework",

                "required": [

                    "source"

                ],

                "properties": {

                    "source": {

                        "type": "number",

                        "description": "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"

                    },

                    "linearAssociations": {

                        "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"

                        }

                    },

                    "nonLinearAssociations": {

                        "type": "array",

                        "description": "An array of non-linear associations mapping a set of values from the source animation framework to one value of the target animation framework",

                        "items": {

                            "$ref": "#/components/schemas/NonLinearAssociation"

                        }

                    }

                }

            },

            "LinearAssociation": {

                "type": "object",

                "description": "Defines a linear mapping between a set of values from the source animation framework to a value of the target animation framework",

                "required": [

                    "targetIndex",

                    "sourceIndices",

                    "weights"

                ],

                "properties": {

                    "targetIndex": {

                        "type": "number",

                        "description": "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": {

                        "type": "array",

                        "description": "Provides 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"

                        }

                    }

                }

            },

            "NonLinearAssociation": {

                "type": "object",

                "description": "Defines a non-linear mapping between a set of values from the source animation framework to a value of the target animation framework",

                "required": [

                    "targetIndex",

                    "channels",

                    "combination"

                ],

                "properties": {

                    "targetIndex": {

                        "type": "number",

                        "description": "Provides the index of the value to be produced by this association. For example, for a blend shape set, this would indicate the index of the blend shape"

                    },

                    "channels": {

                        "type": "array",

                        "description": "Defines an array of channels, each channel transforming the weights from a source animation framework",

                        "items": {

                            "$ref": "#/components/schemas/Channel"

                        }

                    },

                    "combination": {

                        "type": "string",

                        "description": "Defines the method to combine output of each channel",

                        "enum": [

                            "COMBINATION\_SUM",

                            "COMBINATION\_MUL"

                        ]

                    }

                }

            },

            "Channel": {

                "type": "object",

                "description": "Defines functions to compute the outputs using the weights in the target animation framework",

                "required": [

                    "sourceIndex",

                    "input",

                    "output"

                ],

                "properties": {

                    "sourceIndex": {

                        "type": "number",

                        "description": "The index of the value from the source animation framework referenced that contributes to the target index value"

                    },

                    "range": {

                        "type": "array",

                        "items": {

                            "type": "number"

                        },

                        "minItems": 2,

                        "maxItems": 2,

                        "description": "Minimum and maximum values for the source weight"

                    },

                    "path": {

                        "type": "string",

                        "description": "Defines the element of the target to update",

                        "enum": [

                            "CHANNEL\_BLENDSHAPE",

                            "CHANNEL\_JOINT",

                            "CHANNEL\_JOINT\_ROTATION",

                            "CHANNEL\_JOINT\_SCALE",

                            "CHANNEL\_JOINT\_TRANSLATION"

                        ]

                    },

                    "input": {

                        "type": "integer",

                        "description": "The identifier of a data component with the input values"

                    },

                    "interpolation": {

                        "type": "string",

                        "description": "The interpolation method",

                        "enum": [

                            "INTERPOLATION\_LINEAR",

                            "INTERPOLATION\_STEP",

                            "INTERPOLATION\_CUBICSPLINE"

                        ],

                        "default": "INTERPOLATION\_LINEAR"

                    },

                    "output": {

                        "type": "integer",

                        "description": "The identifier of a data component with the output values"

                    }

                }

            },

            "TextureTarget": {

                "type": "object",

                "description": "Defines one of the texture targets used to improve a texture of the material referenced by the TextureSet",

                "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",

                    "id",

                    "mesh"

                ],

                "properties": {

                    "name": {

                        "type": "string",

                        "description": "The name of the skin"

                    },

                    "id": {

                        "type": "number",

                        "description": "A unique identifier of this Skin component"

                    },

                    "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. For example, the skin that contains the head of the avatar may provide the following mapping: \"full\_body/upper\_body/head\" to assign the skin as a child node of the avatar node in the scene graph with the provided hierarchy"

                    },

                    "skeleton": {

                        "type": "number",

                        "description": "A reference to the skeleton"

                    },

                    "blendshapeSet": {

                        "type": "number",

                        "description": "A reference to a blendshapeSet that is defined for this Skin object. If present, the baseMesh of the blendshapeSet shall be the same as the mesh for this Skin"

                    },

                    "landmarkSet": {

                        "type": "number",

                        "description": "A reference to a landmarkSet that is defined for this Skin object"

                    },

                    "textureSet": {

                        "type": "number",

                        "description": "A reference to a textureSet that is defined for this Skin object"

                    },

                    "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 are provided as an NxM-tensor, where N is the number of vertices and M is the number of joints in the referenced skeleton. 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",

                    "data"

                ],

                "properties": {

                    "name": {

                        "type": "string",

                        "description": "The name of the mesh"

                    },

                    "id": {

                        "type": "number",

                        "description": "A unique identifier of the Mesh object"

                    },

                    "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 list of references to data items that contain 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 blend shape set can be used with"

                    },

                    "shapes": {

                        "type": "array",

                        "items": {

                            "type": "number"

                        },

                        "description": "An array of ids of data items that contain each blendshape's data. The shape keys are by default GLB files that only have geometry information (vertices and faces). They may optionally have other attributes such as normals. No materials or textures shall be included. Alternative representations may be possible and need to be identified through the MIME type"

                    },

                    "baseMesh": {

                        "type": "number",

                        "description": "A reference to a Mesh object that contains the base mesh for this blend shape set. Note that the topology of the baseMesh and the associated shapes shall be identical (i.e. same number of vertices and faces)"

                    }

                }

            },

            "LandmarkSet": {

                "type": "object",

                "required": [

                    "name",

                    "id",

                    "baseMesh"

                ],

                "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"

                    },

                    "faces": {

                        "type": "number",

                        "description": "A reference to the Data object that provides the list of mesh face indices on which the landmarks in the landmark set are located"

                    },

                    "weights": {

                        "type": "number",

                        "description": "A reference to the Data object that provides a triplet of barycentric coordinate weights for each landmark in the landmark set"

                    }

                }

            },

            "TextureSet": {

                "type": "object",

                "description": "Defines a set of textures related to a material of a data object",

                "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",

                        "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",

                        "items": {

                            "$ref": "#/components/schemas/TextureTarget"

                        }

                    }

                }

            },

            "Node": {

                "type": "object",

                "required": [

                    "name",

                    "id",

                    "mapping"

                ],

                "properties": {

                    "name": {

                        "type": "string",

                        "description": "The name of the node"

                    },

                    "id": {

                        "type": "number",

                        "description": "A unique identifier of the Node object"

                    },

                    "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 ISO/IEC 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. This is an alternative representation to the TRS and should be used mutually"

                    }

                }

            }

        }

    }

}

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", "id", "type", "uri"],  
 "properties": {  
 "name": {  
 "type": "string",  
 "description": "A string that defines the name of this data"  
 },  
 "id": {  
 "type": "number",  
 "description": "A unique identifier of this Data item"  
 },  
 "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",  
 "description": "Defines the number of bytes used as offset into the data content as pointed to by uri"  
 },  
 "byteLength": {  
 "type": "integer",  
 "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"  
 }  
 }  
 }

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:

Table B.1 — MPEG\_node\_avatar glTF extension

| **Name** | **Type** | **Usage** | **Default** | **Description** |
| --- | --- | --- | --- | --- |
| type | string | M | N/A | The type of the avatar representation is provided as a URN that uniquely identifies 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 of ISO/IEC 23090-14 .  The ARF avatar format shall set this field to the URN defined in clause [4.2](#Section_id-20cdf8f7-f71e-4368-aa8c-44b88). |
| mappings | array(Mapping) | M | N/A | 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(AnimationStream) | M | N/A | An array of objects that each describes an animation stream associated with the base avatar model in the ARF container. |

The AnimationStream object is defined as follows:

Table B.2 — AnimationStream type definition

| **Name** | **Type** | **Usage** | **Default** | **Description** |
| --- | --- | --- | --- | --- |
| type | enumeration | O | ANIMATION\_MUXED | The type of the animation stream. In this version of the specification, it shall be either:   * ANIMATION\_STREAM\_BLENDSHAPES * ANIMATION\_STREAM \_JOINTS * ANIMATION\_STREAM \_LANDMARKS   ANIMATION\_STREAM \_MUXEDNote that the ANIMATION\_STREAM\_MUXED type indicates that the stream may have animation samples from several animation types. The CONFIG unit describes each of these. |
| source | number | M | N/A | A pointer to the accessor that contains the animation data. |

1. (normative)  
     
   Reference Avatar Client

The reference avatar client is depicted in Figure C.1. The reference client architecture is based on the concepts defined in ISO/IEC 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.

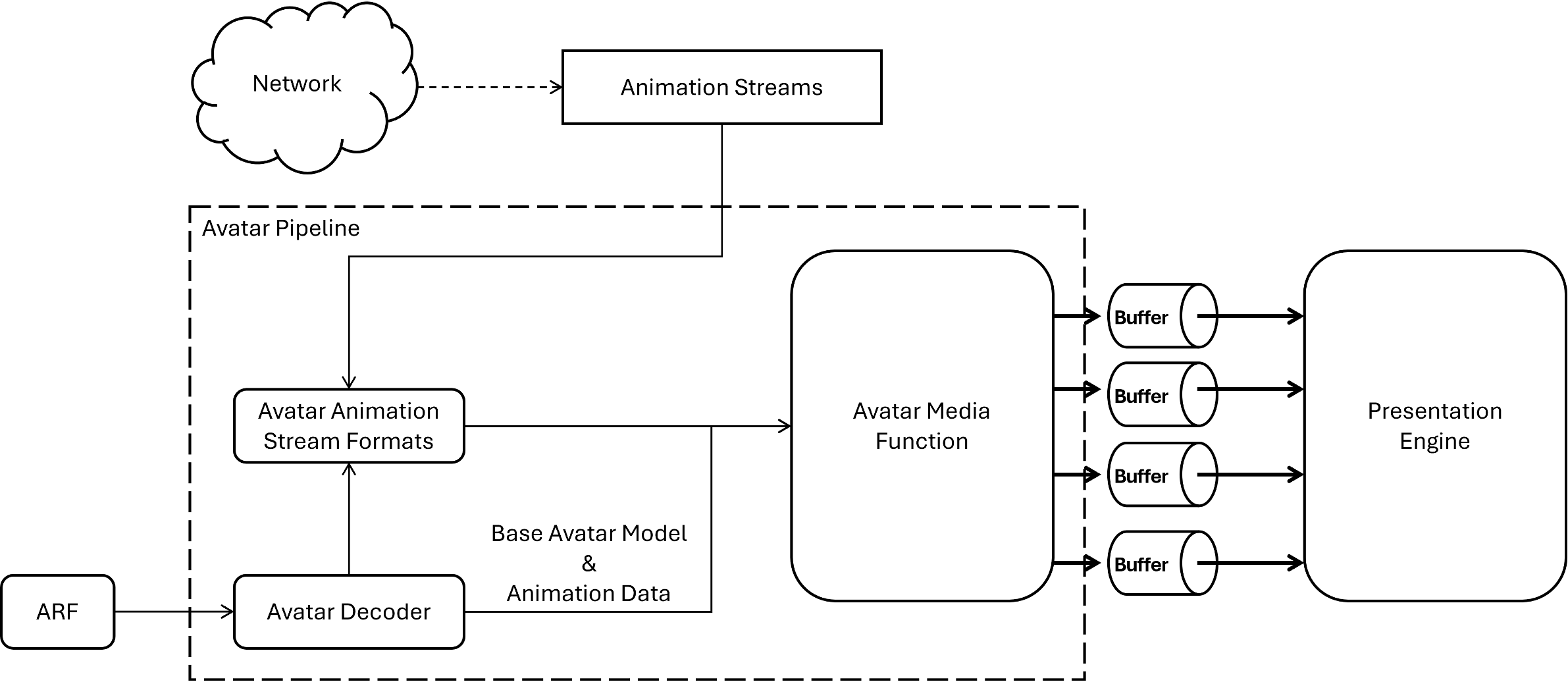


Figure C.1 — Reference Avatar Client Model

The reference client includes essential components that facilitate avatar retrieval, reconstruction, and animation, operating within an integrated workflow. It begins with fetching the ARF container from a remote or local repository, employing protocols suitable for various network conditions. Once the ARF container is acquired, the reference client parses and extracts the relevant avatar components based on the ARF document specifications.

The Avatar pipeline component of the reference client manages the assembly and reconstruction of avatar models from the individual elements specified in the ARF container. This pipeline systematically integrates Meshes, Skins, and Skeletons to form complete avatar structures, preparing them for real-time rendering. It also interprets BlendshapeSets and LandmarkSets to accurately animate detailed facial expressions and nuanced animations through precise vertex manipulation.

Animation streams are integrated within the reference client using the specified Animation Stream Formats. These streams deliver synchronized avatar movements and expressions to the Avatar pipeline, enabling real-time animations that accurately reflect captured input data from tracking devices or other animation sources. The reference client ensures animation integrity by synchronizing animation data with the avatar's reconstruction components, such as Skeletons and Skins, to provide fluid and realistic movement.

The reconstructed avatar is then delivered to the Presentation Engine, which is responsible for rendering the avatar within immersive media environments. This rendering component handles visual optimizations, level-of-detail (LOD) management, and graphical enhancements.

1. (informative)  
     
   Authentication Procedure
   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 Figure D.1.

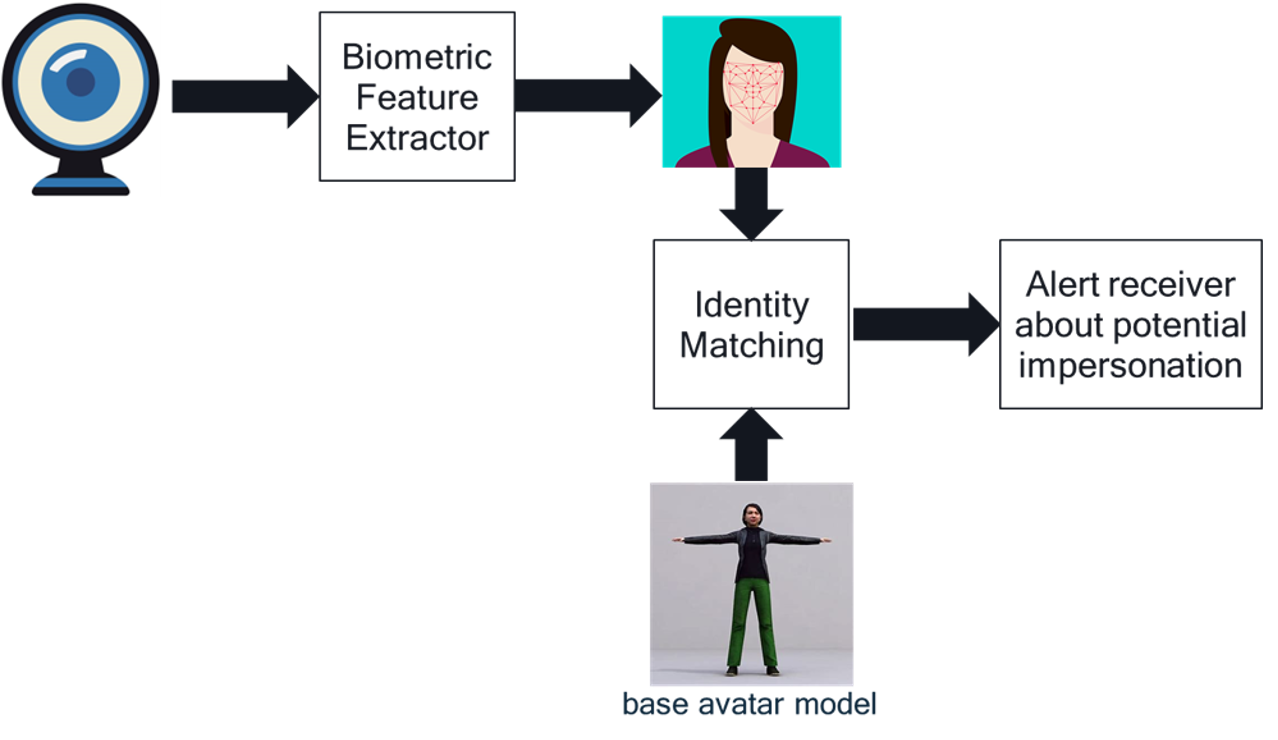


Figure D.1 — Avatar feature verification

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. (normative)  
     
   Tensor Data Format
   1. Dense Tensor Data Format
      1. Introduction

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.

* + 1. Syntax and Semantics

[Table E.1](#Table_id-aae93a15-7ade-4bb5-d802-6bff198) defines the syntax of the dense tensor data item.

Table E.1 — Dense Tensor Data Format

| **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 | int32 | 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 ISO/IEC 12113 clause 5.1.3. |
| data | byte | M | The actual data of the tensor of size size(dtype)\*∏dims[i]. |

* + 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: applicationSubtype

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. Sparse Tensor Data Format
     1. Introduction

The sparse data type represents in general a multi-dimensional tensor. Its dimensions are defined by the “dims” property and its number of dimensions by the “num\_of\_dims” property.

The content of the sparse tensor is encoded as a list of “valueCount” entries representing the non-zero values in the tensor. Each entry consists of an (index, value) pair. “index” is a scalar that specifies the position of the entry in a flattened version of the multi-dimensional sparse tensor. “value” refers to the value of the tensor entry at this position. The flattening of the data tensor is performed in row-major order.

For illustration, assume the sparse tensor has dimensions [4,7,2]. For this tensor, num\_of\_dims=3 and dims=[4,7,2]. Flattening the sparse tensor yields a 1D array of 4x7x2=56 elements. Since flattening is performed in row-major order, assuming zero-based numbering, the first 5 elements of the flattened tensor have indexes (0,0,0), (0,0,1), (0,1,0), (0,1,1), (0,2,0) in the original 3-dimensional tensor.

* + 1. Syntax and Semantics

[Table E.2](#Table_id-b40621ac-2933-46a3-c787-4cd01b8) defines the syntax of the sparse tensor data item.

Table E.2 — Sparse Tensor Data Format

| **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. |
| valueCount | number |  | Number of non-zero entries in the sparse tensor. The tensor entries are encoded as a list of (index, value) pairs where indexes are scalars representing the position of each entry in a 1D-flattened version of the tensor. |
| itype | enum |  | A number that describes the data type of the index element of the (index, value) pairs representing non-zero tensor entries. The allowed data types correspond to the integer glTF 2.0 component types, as specified in glTF 2.0 [3] clause 5.1.3. |
| dtype | int32 | 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 ISO/IEC 12113 clause 5.1.3. |
| indices | byte | M | The actual data that represents the indices of the values in the data array. The size of this data should be size(itype)\*valueCount. |
| data | byte | M | The actual data of the tensor of size size(dtype)\*valueCount. |

* + 1. MIME Type Registration

The MIME type for the sparse tensor data as defined in this document shall be “application/mpeg.arf.sparse”.

* + 1. Registration Form

Type name: application

Subtype name: mpeg.arf.sparse

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. (informative)  
     
   Examples

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