Athena openEHR ITS-JAVA

Base Model (BASE) Component - 1.2.0

Jacob Hoeflaken

October 2024

Table of Contents

Copyright

This document has last been updated on October 2024.

1. Preface

1.1 Purpose

This document describes how the OpenEHR Base Model (BASE) component is implemented in the Athena OpenEHR SDK. See Base Model (BASE) Component - 1.2.0 for the official specification.

1.2 Naming conventions

OpenEHR class names will be transformed into Java *CamelCase* names. For example, the class `DV_TEXT` will be transformed to `DvText`.

Package names will be transformed into valid Java package names if needed. For example, the package `foundation_types` will *not* be transformed into `foundationtypes` or `foundation.types`.

1.3 Overview

The openEHR BASE component consists out of the three packages:

- **Foundation Types**. Specification of assumed, primitive and other foundation types required by all other openEHR specifications. Implemented in the `nl.athena.openehr.base.foundation_types` package.
- **Base Types**. Specification of basic openEHR and health informatics types used in other openEHR specifications. Implemented in the `nl.athena.openehr.base.base_types` package.
- **Resource Model**. The Resource specification defines a formal model of authoring and IP meta-data, language translation and annotations that can be used by classes defining any concrete type of authored resource, such as a document, archetype or template. Implemented in the `nl.athena.openehr.base.resource` package.

2. Foundation Types

The openEHR Foundation Types are a collection of built-in and library types whose semantics are assumed by all other openEHR specifications. Below you will find a mapping of the openEHR Foundation Types to the Athena OpenEHR SDK Java classes.

The base package for the foundation types is `nl.athena.openehr.base.foundation_types`. The `foundation_types` package has six sub-packages: `primitive_types`, `terminology`, `structure`, `interval`, `time` and `functional`.

2.1 Primitive Types

The custom primitive types are implemented in the `nl.athena.openehr.base.foundation_types.primitive_types` package and are listed unqualified in the table below.

openEHR type	Java type	Remarks
Any	java.lang.Object	The ultimate ancestor type.
Ordered	java.lang.Comparable<T>	Classes implementing `Comparable<T>` can be compared and ordered.
Numeric	java.lang.Number	Number implements `Comparable<Number>` and therefore can be compared and ordered. All numeric types in Java inherit from `Number` and are therefore can be ordered as well.
Ordered_Numeric	-	Numeric implements `Comparable<Number>` and therefore can be compared and ordered. No need for a separate class or interface.
Integer	java.lang.Integer	32-bit signed integer.
Integer64	java.lang.Long	64-bit signed integer.
Real	java.lang.Float	32-bit floating point number.
Double	java.lang.Double	64-bit floating point number.
Boolean	java.lang.Boolean	
Character	java.lang.Character	
Octet	java.lang.Byte	Java only supports signed bytes, so the range is -128 to 127.
String	java.lang.String	

openEHR type	Java type	Remarks
Uri	Uri	The built-in Java URI class is not RFC 3986 compliant. The Uri class is.

2.2 Terminology

The custom terminology types are implemented in the `nl.athena.openehr.base.foundation_types.terminology` package and are listed unqualified in the table below.

openEHR type	Java type	Remarks
Terminology_code	TerminologyCode	
Terminology_term	TerminologyTerm	

2.3 Structured Types

The custom structured types are implemented in the `nl.athena.openehr.base.foundation_types.structure` package and are listed unqualified in the table below.

openEHR type	Java type	Remarks
Container	java.util.Collection<T>	
List	java.util.List<T>	
Set	java.util.Set<T>	
Array	T[]	This is the built-in Java array type.
Hash	java.util.Map<K, V>	

2.4 Interval

The custom interval types are implemented in the `nl.athena.openehr.base.foundation_types.interval` package and are listed unqualified in the table below.

openEHR type	Java type	Remarks
Interval	Interval<T extends Comparable<T>>	`T` must implement `Comparable<T>`.
Point_Interval	PointInterval<T extends Comparable<T>>	`T` must implement `Comparable<T>`.
ProperInterval	ProperInterval<T extends Comparable<T>>	`T` must implement `Comparable<T>`.

openEHR type	Java type	Remarks
Multiplicity_Interval	MultiplicityInterval	
Cardinality	Cardinality	

2.5 Time

The custom time types are implemented in the `nl.athena.openehr.base.foundation_types.time` package and are listed unqualified in the table below.

openEHR type	Java type	Remarks
TimeDefinitions	TimeDefinitions	
Temporal	.Temporal	
Iso8601_type	.Iso8601Type	
Iso8601_date	Iso8601Date	Java built-in dates do not support partial dates. The Iso8601Date class does.
Iso8601_time	Iso8601Time	Java built-in dates do not support partial times. The Iso8601Time class does.
Iso8601_date_time	Iso8601DateTime	Java built-in dates do not support partial dates and times. The Iso8601DateTime class does.
Iso8601_duration	Iso8601Duration	Java built-in durations do not support weeks. The Iso8601Duration class does.
Iso8601_timezone	Iso8601Timezone	Java built-in timezones do not support partial timezones. The Iso8601Timezone class does.

2.6 Functional

The custom primitive types are implemented in the `nl.athena.openehr.base.foundation_types.functional` package and are listed unqualified in the table below.

openEHR type	Java type	Remarks
ROUTINE	Routine	Implemented as functional interface.
FUNCTION	Function	Implemented as functional interface.

openEHR type	Java type	Remarks
PROCEDURE	Procedure	Implemented as functional interface.
TUPLE	Tuple	
TUPLE1	Tuple1	
TUPLE2	Tuple2	

3. Base Types

4. Resource Model