

# User Manual 4.3 Attitude leg

De Wiki

Aller à : [navigation](#), [rechercher](#)  
[User Manual 4.3 Attitude leg](#)

## Introduction

### Scope

The purpose of this chapter is to describe the current Patrius attitude legs.

An attitude leg is a time-bounded attitude law. Generalities on attitude laws can be found [ATT\_ALW\_Home here].

### Javadoc

#### Library

#### Javadoc

Patrius [Package fr.cnes.sirius.patrius.attitudes](#)

Patrius [Package fr.cnes.sirius.patrius.attitudes](#)

### Links

Orekit attitudes : [Orekit Attitudes architecture description](#)

### Useful Documents

None as of now.

### Package Overview

The attitude leg `AttitudeLeg` interface inherits the `AttitudeProvider` interface. Its place in the global Attitude design can be found [ATT\_ALW\_PkgOverview here]. It also inherits the general `Leg` interface presented below.

## Features Description

### Generalities: the Leg and the LegsSequence interfaces

Attitude legs inherit the interface `AttitudeLeg`. In addition to `AttitudeProvider` services, they inherit the methods of the `Leg` interface which means they are time-bounded attitude providers.

The `Leg` interface is a generic interface for time-bounded data. It has only one method `getTimeInterval()`. This `Leg` interface allows to define an `AttitudeLeg`.

In order to define `AttitudeLegsSequence`, another generic interface is available: the `LegsSequence` interface. This interface defines a sequence of `Leg` and provides a method to get a leg given a date: `getLeg()`. The `AbstractLegsSequence` class inherits the `LegsSequence` and has numerous methods to automatically handles the underlying Legs: add, remove, iterate over legs, etc. This class sorts the legs by lower end-point date and does not handle any overlapping issues.

**Note:** the `Legand LegsSequence` interfaces are purely generic and therefore can be used for any other time-bounded data.

## Available attitude leg

### Attitude legs sequence

An attitude legs sequence is a container for several attitude legs, contiguous for their time intervals, in such a way that the attitude legs sequence can be processed like a single attitude leg by the propagator.

The attitude legs sequence is the equivalent of the `[ATT_ALW_Home#HAttitudesesquence Attitudes sequence]`, using attitude legs (`AttitudeLeg` instances) rather than attitude laws (`AttitudeLaw` instances). The switching from one attitude leg to another is based on the time interval of validity of the two legs.

### TabulatedAttitude

`TabulatedAttitude` is an implementation of `AttitudeLeg`. It represents a tabulated attitude leg.

In order to interpolate the attitude at a date, the user must specify a list of **ordered** attitudes, and can specify a number of points used by Hermite interpolation. If not specified, the number of points used by Hermite interpolation is set to a default number (currently 2).

```
final List<Attitude> attList = new ArrayList<Attitude>();
attList.add(attitude0);
attList.add(attitude1);
final int nbrInterpPoints = 2;
final TabulatedAttitude attLeg = new TabulatedAttitude(attList,
nbrInterpPoints);
```

It is possible to get the non-interpolated ordered attitudes :

```
final List<Attitude> attitudes = attLeg.getAttitudes();
```

Once the tabulated is defined, the computation can be performed on any orbital state using `getAttitude()` method:

```
Attitude attitude = attLeg.getAttitude(orbit, date,
FramesFactory.getEME2000());
```

### RelativeTabulatedAttitudeLeg

`RelativeTabulatedAttitudeLeg` is an implementation of `AttitudeLeg`. An instance of `RelativeTabulatedAttitudeLeg` can be created with a `List<Pair<Double, Rotation>>` or with a `List<Pair<Double, AngularCoordinates>>`. Each `Rotation` (or `AngularCoordinates`) is associated with a `double` representing its time elapsed in seconds since the reference date. Here is an example of a creation of an instance of `RelativeTabulatedAttitudeLeg` :

```

// date and frame
AbsoluteDate refDate = new AbsoluteDate(2008, 1, 1,
TimeScalesFactory.getTAI());
Frame frame = FramesFactory.getGCRF();
double timeEllapsedSinceRefDate = 1.0;

// List of AR
List<Pair<Double, AngularCoordinates>> listAr = new ArrayList<Pair<Double,
AngularCoordinates>>();
final AngularCoordinates ar = new AngularCoordinates(
    new Rotation(false, 0.48, 0.64, 0.36, 0.48), Vector3D.PLUS_I,
Vector3D.PLUS_J);
listAr.add(new Pair<Double, AngularCoordinates>(timeEllapsedSinceRefDate,
ar));

// create RelativeTabulatedAttitudeLeg
final RelativeTabulatedAttitudeLeg relativeTabulatedAttitudeLeg =
    new RelativeTabulatedAttitudeLeg(refDate, frame, listAr);

```

## Getting Started

### Building an attitude legs sequence

The attitude legs sequence was designed as a simple container, it performs only a few coherence checks on its inner attitude laws. Here's how an attitude sequence is built.

- An attitude legs sequence is created empty, associated to a single `PVCoordinatesProvider` instance.
- The sequence is mutable, attitude laws can be added to it one by one.
- Each attitude law is identified by a code.
- The sequence has a validity time interval, which is the grouping of the validity time intervals of all contained laws.
- The time interval of a newly added law must be contiguous to the grouped time interval of the already added laws. Otherwise an `PatriusException` is thrown.
- As soon as the sequence contains at least one law, methods of the `AttitudeLeg` interface can be called on the attitude sequence. The attitude sequence forwards the request to the appropriate attitude leg (according to the asking date), but replaces the `PVCoordinatesProvider` attribute of the call with the inner `PVCoordinatesProvider` instance.

### AttitudeLawLeg and AttitudeLegsSequence : Code sample

```

final BodyCenterPointing earthCenterAttitudeLaw = new
BodyCenterPointing(itrf);
final AttitudeLeg law1 = new AttitudeLawLeg(earthCenterAttitudeLaw, date1,
date2);
final AttitudeLeg law2 = ... ;
final AttitudeLeg law3 = ... ;

```

```

PVCoordinatesProvider pvProvider = new CartesianOrbit(pvCoordinates, gcrf,
date1, mu);
final AttitudeLegsSequence sequence = new AttitudeLegsSequence(pvProvider);
// After each add the sequence has to be contiguous, so the order may be
important
sequence.add("L1", law1);
sequence.add("L2", law2);
sequence.add("L3", law3);

// Call to getAttitude on the sequence ignores otherPvProvider and uses
pvProvider internally instead
final Attitude sAttitude = sequence.getAttitude(otherPvProvider, date, itrfr);

```

## Contents

### Interfaces

Interface	Summary	Javadoc
<b>Leg</b>	This interface is a generic interface for any kind of time-bounded data.	<a href="#">...</a>
<b>LegsSequence</b>	This interface is a generic interface for any sequence of legs.	<a href="#">...</a>
<b>AttitudeLeg</b>	This interface extends the AttitudeProvider interface and adds the time interval of validity notion to the attitude laws.	<a href="#">...</a>

### Classes

Class	Summary	Javadoc
<b>AttitudeLawLeg</b>	Object representing an attitude law for spacecraft attitude field purposes.	<a href="#">...</a>
<b>TabulatedAttitude</b>	Object representing a tabulated attitude leg : the attitude at a date is interpolated from a list of known ones.	<a href="#">...</a>
<b>AttitudeLegsSequence</b>	Object representing a sequence of contiguous attitude legs.	<a href="#">...</a>
<b>RelativeTabulatedAttitudeLeg</b>	This class implements a tabulated attitude leg with relative dates.	<a href="#">...</a>

Récupérée de « [http://patrius.cnes.fr/index.php?title=User\\_Manual\\_4.3\\_Attitude\\_leg&oldid=2335](http://patrius.cnes.fr/index.php?title=User_Manual_4.3_Attitude_leg&oldid=2335) »  
[Catégorie](#) :

- [User Manual 4.3 Attitude](#)

## Menu de navigation

### Outils personnels

- [3.135.190.107](#)
- [Discussion avec cette adresse IP](#)
- [Créer un compte](#)

- [Se connecter](#)

## Espaces de noms

- [Page](#)
- [Discussion](#)

## Variantes

## Affichages

- [Lire](#)
- [Voir le texte source](#)
- [Historique](#)
- [Exporter en PDF](#)

## Plus

## Rechercher

## PATRIUS

- [Welcome](#)

## Evolutions

- [Main differences between V4.15 and V4.14](#)
- [Main differences between V4.14 and V4.13](#)
- [Main differences between V4.13 and V4.12](#)
- [Main differences between V4.12 and V4.11](#)
- [Main differences between V4.11 and V4.10](#)
- [Main differences between V4.10 and V4.9](#)
- [Main differences between V4.9 and V4.8](#)
- [Main differences between V4.8 and V4.7](#)
- [Main differences between V4.7 and V4.6.1](#)
- [Main differences between V4.6.1 and V4.5.1](#)
- [Main differences between V4.5.1 and V4.4](#)
- [Main differences between V4.4 and V4.3](#)
- [Main differences between V4.3 and V4.2](#)
- [Main differences between V4.2 and V4.1.1](#)

- [Main differences between V4.1.1 and V4.1](#)
- [Main differences between V4.1 and V4.0](#)
- [Main differences between V4.0 and V3.4.1](#)

## **User Manual**

- [User Manual 4.15](#)
- [User Manual 4.14](#)
- [User Manual 4.13](#)
- [User Manual 4.12](#)
- [User Manual 4.11](#)
- [User Manual 4.10](#)
- [User Manual 4.9](#)
- [User Manual 4.8](#)
- [User Manual 4.7](#)
- [User Manual 4.6.1](#)
- [User Manual 4.5.1](#)
- [User Manual 4.4](#)
- [User Manual 4.3](#)
- [User Manual 4.2](#)
- [User Manual 4.1](#)
- [User Manual 4.0](#)
- [User Manual 3.4.1](#)
- [User Manual 3.3](#)

## **Tutorials**

- [Tutorials 4.15](#)
- [Tutorials 4.14](#)
- [Tutorials 4.13.5](#)
- [Tutorials 4.12.1](#)
- [Tutorials 4.8.1](#)
- [Tutorials 4.5.1](#)
- [Tutorials 4.4](#)
- [Tutorials 4.1](#)
- [Tutorials 4.0](#)

## **Links**

- [CNES freeware server](#)

## **Navigation**

- [Accueil](#)
- [Modifications récentes](#)
- [Page au hasard](#)
- [Aide](#)

## Outils

- [Pages liées](#)
- [Suivi des pages liées](#)
- [Pages spéciales](#)
- [Adresse de cette version](#)
- [Information sur la page](#)
- [Citer cette page](#)
  
- Dernière modification de cette page le 3 juin 2019 à 14:41.
  
- [Politique de confidentialité](#)
- [À propos de Wiki](#)
- [Avertissements](#)
  
- 