User Manual 4.11 Frames configuration

User Manual 4.11 Frames configuration

Introduction

Scope

Frames configuration defines the models and data to use for each frame transformation. Several frames configurations are already available in PATRIUS (IERS 2003, IERS 2010, STELA? etc.) but the user may define its own convention.

Javadoc

Library	Javadoc
Patrius	Package fr.cnes.sirius.patrius.frames.configuration
Patrius	Package fr.cnes.sirius.patrius.frames.configuration.eop
Patrius	Package fr.cnes.sirius.patrius.frames.configuration.tides
Patrius	Package fr.cnes.sirius.patrius.frames.configuration.libration
Patrius	Package fr.cnes.sirius.patrius.frames.configuration.precessionnutation

Links

- IERS website
- IERS 2010 Conventions
- IERS 2003 Conventions

Useful Documents

None as of now.

Package Overview

None as of now.

Features Description

The Frames Configuration

In PATRIUS, a Frames configuration is defined as a set of models that are used to compute transformations between different frames. As of now, these models define the transformations between the frames defined by the IERS (see IERS TN36 Ch. 5). Thus, the configuration holds all the models required to transform a position or a velocity from the International Terrestrial Reference System (ITRS) and the Geocentric Celestial Reference System (GCRS) and vice versa.

When no configuration is created by the user, PATRIUS uses the IERS 2010 convention. Note that

this convention requires to provide EOP data. The library also allows the user to create configurations by setting up the adequate models.

The FramesConfigurationFactory contains public static instances of standard configurations, amongst which:

- the configuration following the IERS 2010 conventions.
- A simple configuration which can be retrieved by calling `FramesConfigurationFactory.getSimpleConfiguration(boolean)`. This configuration is useful because it removes the need for EOP data. Depending on the value of the provided boolean, this configuration leads to the following equalities:
  - Boolean = true (use of EOP models): GCRF != CIRF, TIRF = ITRF
  - Boolean = false (no use of EOP models): GCRF == CIRF, TIRF = ITRF. Hence only Earth Rotation Angle is taken into account from GCRF to ITRF frame.

The default configuration (IERS2010)

The default configuration (IERS2010) is convenient for most applications. In this case, no operation pertaining to frames configurations needs to be undertaken. At the first call requiring frames, the fr/cnes/sirius/patrius/frames/FramesFactory.html FramesFactory will create the default configuration and use it accordingly.

The code snippet below is valid as long as the user has NOT passed a custom configuration to the FramesFactory.

```
/*
/* Example of frame transformations using IERS2010 convention
/*/
// instance of ITRF with the default configuration (IERS2010)
final Frame itrf = FramesFactory.getITRF();
// instance of GCRF with the default configuration (IERS2010)
final Frame gcrf = FramesFactory.getGCRF();
// transformation relating ITRS to GCRS at currentDate (supposed as already
defined)
final Transform itrfTransform = itrf.getTransformTo(gcrf, currentDate);
// transformed currentPV (supposed as already defined)
final PVCoordinates Vtransformed =
itrfTransform.transformPVCoordinates(currentPV);
// ...
```

The default IERS 2010 configuration is as follows :

- IERS 2010 tidal correction model
- IERS 2010 libration correction model
- IERS 2010 S' model
- account of EOP UT1-UTC and pole corrections
- IERS 2010 precession and nutation (interpolated computation) with nutation correction to X and Y and account of angular derivatives.

Creating a customised configuration

PATRIUS allows creating customized configurations to define the intermediate transformations relating ITRS to TIRS, TIRS to CIRS and CIRS to GCRS.

Transformation relating ITRS to TIRS

Based on polar motion, it can be expressed as: [math]W(t)=R3(-s').R2(xp).R1(yp)[/math], where *xp* and *yp* are the polar coordinates of the Celestial Intermediate Pole (CIP) in the ITRS and *s'* is called the "Terrestrial Intermediate Origin locator".

The user can define the following parameters:

- polar motion (yes/no),
- interpolation method for *xp* and *yp* (linear or Lagrange 4),
- account of ocean tidal effect *(dx,dy),,ocean tides,,* in polar motion (IERS2010, IERS2003 or no effect),
- account of libration effect *(dx,dy),,libration,,* in polar motion (IERS2010 or no effect),
- account of the quantity *s'* (IERS2010, IERS2003 or no effect).

To create a polar motion model with chosen parameters, PATRIUS provides the constructor **PolarMotion** (see PolarMotion javadoc for the description of the above parameters).

Transformation relating TIRS to CIRS

Based on the rotation of the Earth around the axis of the CIP, it can be expressed as: [math]R(t)=R3(-ERA)[/math], where *ERA* is the Earth Rotation Angle between the CIO and the TIO.

The user can define the following parameters:

- account of *UT1-UTC* (yes or no),
- interpolation method for *UT1-UTC* (linear or Lagrange 4),
- account of ocean tidal effect *dUT1,,ocean tides,,* in *UT1* (IERS2010, IERS2003 or no effect),
- account of libration effect *dUT1,,libration,,* in *UT1* (IERS2010 or no effect).**Note that this effect is not yet implemented. It is negligible in current practical applications due to its very small size**.

To create a diurnal rotation model with chosen parameters, PATRIUS provides the constructor **DiurnalRotation** (see DiurnalRotation javadoc for the description of the above parameters).

Transformation relating CIRS to GCRS

Based on the celestial motion of the CIP, it can be expressed as: [math]Q(t)=R3(-E).R2(-d).R3(E).R3(s)[/math], *E* and *d* being such that the coordinates of the CIP in the GCRS are: & and **IAU 2000A_R06 nutation**. The IAU series are given in the *tab5.2a* file (X model), in the *tab5.2b* file (Y model) and in the *tab5.2d* file (s model).

The user can define the following parameters:

- account of precession and nutation model (IERS2010, IERS2003 or no effect),
- interpolation method for *X*, *Y* and *s* (Neville method or direct computation),
- nutation corrections to the *X* and *Y* coordinates reported by the IERS as "celestial pole offsets" (yes or no),

- interpolation method for the corrections (linear or Lagrange 4),
- account of non constant rotation (yes or no) to compute properly the velocity

To create a precession nutation model with chosen parameters, PATRIUS provides the constructor **PrecessionNutation** (see PrecessionNutation javadoc for the description of the above parameters).

Using specific EOP history files / data

Earth Orientation Parameters (EOP) files provide angular corrections that account for short-period variations of Earths' Celestial Intermediate Pole. The corresponding data is used to compute the transformation from the TIRF to the ITRF frame. The amplitudes of the corrections are given in [FDY_FRAME_Home#HPolarmotion Polar Motion]. IN PATRIUS, the EOPHistory interface represents EOP data. It has two implementations : EOP1980History and EOP2000History. These classes differ because the EOP definition and representation has changed, according to the IERS conventions. Both remain available in PATRIUS.

EOP data can be retrieved in two ways in PATRIUS:

By providing your own history:

For the 2000 EOP paradigm, the user may manually create an EOP history with EOP entries. In order to do so, the user must create an instance of EOPHistory and feed it with EOP entries :

```
        // create EOP list- use a 1 day gap
        final List<EOP2000Entry> list = new ArrayList<EOP2000Entry>();
        list.add(entry1);
        list.add(entry2);
        list.add(entry3);
        ...
        list.add(entryN);

        // Create EOP history object- make sure to have enough entries around
the dates of interpolation
        final EOP2000History eop2000History = new
EOP2000History(EOPInterpolators.LAGRANGE4);

        // Fill history
        EOP2000History.fillHistory(list, eop2000History);
```

Also, the user may create an empty EOPHistory object and populate it using a Java InputStream (usually, a file). with the adequate loader (see eop package) :

```
        final InputStream is = new FileInputStream("eopdatafile");
        // Empty EOP history object-  choice of interpolator
        final EOP2000History eop2000History = new
EOP2000History(EOPInterpolators.LAGRANGE4);
        // EOPC04FilesLoader reads the input stream and fills the history
object
        EOPC04FilesLoader.fillHistory(eop2000History, is);
```

Of course, in this case, the file has to be in the proper format for the EOPC04FilesLoader (see its javadoc for information).

For the 1980 EOP paradigm, the user must proceed in a similar fashion by manually creating an EOP1980HistoryLoader instance and feeding it to the EOPHistoryFactory.

- **By automatically loading data using default EOP loaders:**

```
final EOP2000History eop2000History =
EOPHistoryFactory.getEOP2000History(EOPInterpolators.LAGRANGE4);
```

This previous line of code returns an EOP 2000 history filled up with found data. Found data includes (in order of data reading):

- IERS Rapid data and prediction files
- EOC04 files
- IERS bulletin B files

Warning: providing overlapping data may results in some erratic results since some dates will have several EOP data.

Finally the EOP history is provided to the frame configuration builder:

```
        FramesConfigurationBuilder builder = new
FramesConfigurationBuilder(config);
        builder.setEOPHistory(eop2000History);
        FramesFactory.setConfiguration(builder.getConfiguration());
```

Note 1: for the users and applications that do not require any EOP data, utility classes that return zero for all EOP corrections have been implemented. These classes possess validity intervals that span 100 years (1950 through 2050) for the EOP 1980 paradigm and infinity (-infinity through +infinity) for the EOP 2000 paradigm. This means that any application using these classes can request EOP data over these intervals.

- For the 1980 EOP paradigm, a loader class has been implemented : NoEOP1980HistoryLoader. The user need only to create an instance of that class and feed it the EOPHistoryFactory in order for all transformations to use zero for EOP corrections (only 1980). This done thanks to the following instruction : `EOPHistoryFactory.addEOP1980HistoryLoader(new NoEOP1980HistoryLoader())`. This loader then feeds an internal instance of EOP1980History, that the users does not manipulate directly.
- For the 2000 EOP paradigm, a utility class, NoEOP2000History, was created and represents an EOP history with no corrections (or more precisely 0.0). It has an infinite time interval of validity, from PAST_INFINITY to FUTURE_INFINITY.

Note 2: Note that most EOP providers are valid on data-provided timespan. The class `EOP2000HistoryConstantOutsideInterval` circumvents that limitation by returning boundaries values when requested date is out of data range. Warning: in this case, as the UT1-TAI remains constant the (UT1-UTC) absolute value returned may become higher than 0.9 second depending on leap seconds (for UTC time scale) existing outside this interval.

Note on the internal representation of EOP Data

As of 2.1, EOP data sets have been modified to store the value of UT1-TAI instead of UT1-UTC. Additionally, new constructors that allow creating EOPEntry instances from UT1-TAI data have been added. The frames tranformations have thus become independent of the UTC-TAI.history file, provided that the user provides a correctly calculated UT1-TAI. Please refer to the javadoc for more information.

It should be noted that UT1 is defined with respect to UTC. Consequently, the only way to correctly compute UT1-TAI is per UT1-TAI = UT1-UTC + UTC-TAI.

Available models

As of now, available models include :

- IERS 2003 and 2010 Precession Nutation models,
- IERS 2010 S' Model,
- IERS 2003 and 2010 Tidal corrections models,
- IERS 2010 Libration correction model (without UT1-UTC correction).

Amplitudes of different effects

This section provides some useful information about the different effect of each transformation. The distances are given at one terrestrial radius. These data come from the document "*DCT/SB/OR/2010-12497", issue 1.3, 2010/11/23*".

Polar motion

- *(xp,yp),,IERS,,* < 1 *(30m)*
- *(dx,dy),,ocean tides,,* < 0.002 *(6cm)*
- *(dx,dy),,libration,,* < 0.00003 *(1mm)*
- *s* **< *0.00005*** **per century**

Diurnal rotation

- *UT1-UTC,,IERS,,* < 1 sec. (460m)
- *UT1-UTC,,ocean tides,,* < 33E-6 sec. (1.5cm)
- *UT1-UTC,,libration,,* < 4E-6 sec. (2mm)

Celestial motion of the CIP

- *(X,Y),,IAU2000/2006,,* < 11 *(340m)*
- *(dX,dY),,IERS,,* < 0.001 *(3cm)*

Validation of IERS2010 implementation

PATRIUS implementation of IERS2010 convention has been validated with reference software routines from *SOFA*, *ZOOM*, and other routines available in IERS website (like *interp.f* routine for EOP data interpolation).

The first level of validation allows to compare the values for each intermediate transformation:

- comparison of *xp*, *yp* and *s'* for the transformation relating ITRS to TIRS,
- comparison of *UT1-UTC* for the transformation relating TIRS to CIRS,

- comparison of *X*, *Y*, *s* and *dX*, *dY*, *ds* for the transformation relating CIRS to GCRS.

The second level of validation allows to compare the rotation matrices (*Q(t)*, *R(t)*, *W(t)*) by computing angular deviations with the reference.

Several configurations are used to take into account the different effects. Tests have been done for short and long period.

The following graphs represent angular deviations (in log scale) for the following transformations: ITRS to GCRS in dark green, ITRS to TIRS in red, TIRS to CIRS in blue-green and CIRS to GCRS in violet. The first one is related to a short period (10 days), the second one is related to a long period (40 years). The blue line is the validation threshold of**0.1E-3 arcseconds**. The deviations computed for all transformations are smaller than 0.1E-3 arcseconds.

As a consequence, some deviations are observed on position and velocity of points used in the context of this validation. The points are taken on LEO and GEO orbits, and also on the terrestrial crust. To give the user some indication, the magnitude of errors is between 1E-4 and 1E-5 m.

For more details, please refer to the validation report "*SIRIUS-SVS-DV-10108-THA, issue 1.0, 2012/07/23*".

Getting Started

Example of frame configuration

The example describes how to create a frame configuration defined by the following parameters:

- polar motion with IERS2003 corrections (ocean tidal effect, no libration effect and account of quantity *s'*),
- account of *UT1-UTC* with IERS2003 corrections (ocean tidal effect),
- account of precession and nutation (direct or interpolated computation) with nutation correction to *X* and *Y* and account of angular derivatives.

PATRIUS offers a configuration factory to make the creation easier. Several methods are available to set-up the configuration with the user parameters.

```
        // Configurations builder
        final FramesConfigurationBuilder builder = new
FramesConfigurationBuilder();

        // Tides and libration
        final TidalCorrectionModel tides =
TidalCorrectionModelFactory.TIDE_IERS2003_INTERPOLATED;
        final LibrationCorrectionModel lib =
LibrationCorrectionModelFactory.NO_LIBRATION;

        // Polar Motion
        final PolarMotion defaultPolarMotion = new PolarMotion(true, tides,
lib, SPrimeModelFactory.SP_IERS2003);
```

```
        // Diurnal rotation
        final DiurnalRotation defaultDiurnalRotation = new
DiurnalRotation(tides, lib);

        // Precession Nutation
        final PrecessionNutation precNut = new PrecessionNutation(
                true,
PrecessionNutationModelFactory.PN_IERS2003_INTERPOLATED_NON_CONSTANT);

        builder.setDiurnalRotation(defaultDiurnalRotation);
        builder.setPolarMotion(defaultPolarMotion);
        builder.setPrecessionNutation(precNut);

builder.setEOPHistory(EOPHistoryFactory.getEOP2000History(EOPInterpolators.
LAGRANGE4));

        final FramesConfiguration config = builder.getConfiguration();
        // and pass it to the frames factory
        FramesFactory.setConfiguration(config);

        // Get the frames
        final Frame itrf = FramesFactory.getITRF();
        final Frame gcrf = FramesFactory.getGCRF();
```

Contents

Interfaces

Interfaces used within the Frames configurations

Interface	Summary	Javadoc
FramesConfiguration	Interface providing the basic services for frame configurations.	...
EOPHistory	Interface for retrieving Earth Orientation Parameters history throughout a large time range.	...
LibrationCorrectionModel	This interface provides the pole corrections as well as the ut1-utc corrections due to libration.	...
PrecessionNutationModel	This interface provides the Celestial Intermediate Pole motion (CIP) in the GCRS, those coordinates are used for the GCRF to CIRF transformation.	...
SPrimeModel	This interface provides the s' correction (used for the following transformation : TIRF -> ITRF).	...
TidalCorrectionModel	This interface provides the pole corrections as well as the ut1-utc corrections due to tidal effects.	...

Classes

The classes are mentioned according to their packages for better visibility. Note : The aforementioned EOP2000History class is to be used with the PATRIUS IERS Frames Configuration.

General	Summary	Javadoc
FramesConfigurationBuilder	Frame configuration builder utility, to assist the user in creating a FramesConfiguration.	...
FramesConfigurationFactory	Frames configuration factory. Contains useful configurations.	...

Models container classes	Summary	Javadoc
DiurnalRotation	This class contains the different ut1-utc corrections (libration, tidal effects).	...
PolarMotion	This class contains the different polar motion corrections (libration, tidal effects, sp correction).	...
PrecessionNutation	This class contains the precession nutation model used within the FramesConfigurationImplementation class.	...

Libration models classes	Summary	Javadoc
IERS2010LibrationCorrection	This class computes the diurnal lunisolar effect. It is a java translation of the fortran subroutine PM_GRAVI (provided by CNES and from IERS conventions, see chapter 5, tables 5.1a and 5.2a).	...
NoLibrationCorrection	This class ignores the libration effects.	...

Precession Nutation models classes	Summary	Javadoc
PrecessionNutationConvention	IERS Precession Nutation enumerate. Each enumerate provides data file locations.	...
IERS20032010PrecessionNutation	Compute the precession nutation correction according to the new IAU-2000 model.	...

S' models classes	Summary	Javadoc
IERS2010SPCorrection	Compute s' correction (IERS 2010 model).	...
NoSpCorrection	This class ignores the quantity s'.	...

Tidal correction models classes	Summary	Javadoc
IERS2010TidalCorrection	Compute tidal correction of the pole motion.	...
IERS2003TidalCorrection	Compute tidal correction of the pole motion.	...
NoTidalCorrection	This class ignores the tidal effects.	...

EOP classes	Summary	Javadoc
EOP2000History	This class holds Earth Orientation Parameters (IAU2000) data throughout a large time range.	...
EOP2000HistoryConstantOutsideInterval	This class extends the EOP2000History but returns boundaries values when date is out of data range.	...
EOPC04FilesLoader	This class is a generic reader for EOPC04 files.	...

Menu de navigation

Outils personnels

- 3.145.52.195
- 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

Rechercher Lire

PATRIUS

- Welcome

Evolutions

- 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 4.14
- 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 4.14
- 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 23 mai 2023 à 08:22.

- Politique de confidentialité
- À propos de Wiki
- Avertissements

-