User Manual 3.3 Events: ground stations and satellites

User Manual 3.3 Events: ground stations and satellites

Introduction

Scope

Here are presented all the events detectors of the theme "stations and satellites".

Javadoc

Those event detectors are available in the packages :

Library	Javadoc
Orekit	Package org.orekit.propagation.event
Patrius	Package fr.cnes.sirius.patrius.events

The classes to build the stations models and satellites models are in the packages (and their sub-packages) :

Library	Javadoc
Patrius	Package fr.cnes.sirius.patrius.assembly
Patrius	Package fr.cnes.sirius.patrius.groundstation

Features Description

g functions

This section describes the meaning of the g switching function for the "stations and satellites" event detectors :

ApparentElevationDetector

The g switching function computes the difference between the satellite apparent elevation [math]\theta_{app}[/math](the sum of geometrical elevation and refraction angle) and the threshold elevation [math]\theta_{thr}[/math].
This function is identical to the satellite elevation g function, except for the refraction angle term.
The tropospheric correction refraction angle is computed according to Saemundssen formula quoted by Meeus.

ElevationDetector

The g switching function is the difference between the current elevation [math]\theta

_{satellite}[/math]and the threshold elevation [math]\theta _{seuil}[/math].

- [math]g\lt0 :[/math]: the satellite is outside the cone delimited by the threshold elevation angle;
- [math]g\gt0 :[/math]: the satellite is inside the cone.

ExtremaElevationDetector

The `ExtremaElevationDetector` detects if the spacecraft is at a local extremum for the elevation. The choice of the extremum (maximum, minimum, or both) is done with the constructor, through the `elevationType` parameter.
The g switching function is the projection of the spacecraft's velocity on the Z axis of the topocentric station frame : its value is zero when the elevation (see the elevation [math]\theta _{satellite}[/math] definition of the `ElevationDetector`) reaches its minimal or maximal value.

GroundMaskElevationDetector

The g switching function computes the difference between the current elevation [math]\theta _{sat}[/math]and the elevation for current azimuth interpolated from azimuth-elevation mask : [math]\theta _{mask}(\alpha _{sat})[/math]. Since the [math]\theta _{mask}[/math] term can be considered as the threshold angle, the ground mask elevation detector g function is identical to the elevation detector function.

VisibilityFromStationDetector

The g function is positive when the spacecraft is in the station's field of view, negative otherwise. The station is defined by its topocentric frame. A field of view of the `IFieldOfView` type, defined in that frame, must be given, or directly an array of azimuths and elevations to define the field as it is done in the `GroundMaskElevationDetector`'s constructor. A tropospheric correction (`TroposphericCorrection` interface) can be taken into account in the computation. Set that correction as "null" to ignore it.
The station sensor model must have been correctly defined, its target being the spacecraft's sensor model (please refer to the next section : "PATRIUS models for satellites and stations antennas").

StationToSatMutualVisibilityDetector

The g function is positive when the spacecraft is in the station's field of view (same definitions as in the `VisibilityFromStationDetector` detector, with same use of a tropospheric correction, only for the sight from the station), AND the station is in the spacecraft's given sensor's field of view (same definition as in the `TargetInFieldOfViewDetector`), negative otherwise.
The spacecraft's sensor must have been correctly defined, its target being the station sensor model (please refer to the next section : "PATRIUS models for satellites and stations antennas").
A boolean set at construction allows the user to decide to take maskings into account or not. So if asked, the visibility is interrupted when an object is masking, computing it the same way as in the MaskingDetector.

SatToSatMutualVisibilityDetector

The g function is positive when the main spacecraft (the one concerned by the orbit propagation that inclues this detector) is the secondary one's given sensor's field of view AND the secondary spacecraft is in the main one's given sensor's field of view (both with the same definition as in the `TargetInFieldOfViewDetector`).
Each of the two sensors must have been correctly defined, with the other sensor being the main target (please refer to the next section : "PATRIUS models for satellites and stations antennas").
A boolean set at construction allows the user to decide to take maskings into account or not. So if asked, the visibility is interrupted when an object is masking, computing it the same way as in the MaskingDetector.

RFVisibilityDetector

This detector computes the link budget between the ground station antenna and the satellite antenna and compares it to a threshold value; the model representing the RF link budget (`RFLinkBudgetModel` class) must be known by the detector in order to perform this computation (see the Link budget properties and models page).
The g switching function is positive when the link budget is bigger than a threshold value; the threshold value is chosen by the user as it is an input parameter of the detector.

Focus on visibility detection

PATRIUS models for station and satellite antennas

The `VisibilityFromStationDetector`, `StationToSatMutualVisibilityDetector` and `SatToSatMutualVisibilityDetector` detectors use the PATRIUS ground station antennas and satellite antennas models. Here is a short description of them.
For more details, please refer to the specific pages (mainly in the Spacecraft user manual).

Station antennas

For advanced use of stations antennas, the `GeometricStationAntenna` or `RFStationAntenna` objects are used (package `fr/cnes/sirius/patrius/groundstation`).
The `GeometricStationAntenna` is defined by :

- its topocentric frame (see the frames page);
- its field of view expressed in the topocentric frame (see the fields of view page).

The `RFStationAntenna` is defined by :

- its topocentric frame;
- its specific parameter for RF Link Budget computations;
- no limitation of the field (no mask, no field of view). It can be limited by the RF parameters.

For detection of simple visibility events, not depending on satellite attitude, these objects are not necessary (use Orekit `ElevationDetector`, `ApparentElevationDetector`, or `GroundMaskElevationDetector`).

Satellite antennas

A generic satellite antenna should be represented by a Part of an assembly. One or more properties can be added to this Part to model its physical and/or geometrical features.

The `SensorProperty` can be added to a satellite antenna in order to simulate the mutual visibility among antennas; it must contain at least a "main field of view", that will be used for all the computations.
The `SensorProperty` is used by a `SensorModel` of the assembly), which contains the algorithm to compute the antennas visibility. Once a `SensorModel` built, the proper target must be set, using the "setMainTarget" method :

- For the station-satellite mutual visibility, the station model, that is a PVCoordinatesProvider) must be set as this main target.
- For satellites mutual visibility, the `SensorModel` of the other satellite must be set as this main target, for both of them.

The `RFAntennaProperty` is used to set some RF features to a satellite antenna, when computing an RF link budget.

Detection of satellite/ground station visibility events

The `VisibilityFromStationDetector` and `StationToSatMutualVisibilityDetector` both use the same description of a station antenna. Both have two constructors :

- One that needs a `GeometricStationAntenna` object to define the station (frame and field of view)
- One that directly needs a topocentric frame, and an azimuth/elevation array. Internally, this one works as the first: it creates a field of view from the azimuth/elevation array. It is juste more simple to use.

A tropospheric correction can be considered when computing the visibility; in this case, the user should use the classes implementing the `TroposphericCorrection` interface (in the package `fr/cnes/sirius/patrius/signalpropagation`). When using a tropospheric correction, the temperature, moisture and pressure parameters of the station must be, if necessary, coherent with the considered station in the given tropospheric model.

Here is a code sample to create properly a mutual station/satellite visibility detector:

```
// tropospheric correction
//========================
// pressure (millibars)
final double pressure = 1020;
// temperature (celcius)
final double temperature = 20;
// moisture (percent)
final double moisture = 20;
// geodetic altitude (m)
final double altitude = 150;

final AngularCorrection correctionModel = new AzoulayModel(pressure,
temperature, moisture, altitude);

// station antenna model
//======================
// frame
```

```
final TopocentricFrame topoFrameStation = new TopocentricFrame(earth, point,
"Gstation");
// field of view
final String nameStationField = "circularStationField";
final IFieldOfView stationField = new CircularField(nameStationField,
FastMath.PI / 8., Vector3D.PLUS_K);
// ground station
final GeometricStationAntenna stationModel = new
GeometricStationAntenna(topoFrameStation, stationField);

// spacecraft sensor model
//=======================
// We assume here that an assembly has been built, with a sensor property on
this "sensorPart" part.
// model creation
final SensorModel spacecraftSensorModel = new SensorModel(assembly,
sensorPart);
// target adding (the antenna is here considered ponctual)
spacecraftSensorModel.setMainTarget(stationModel, new
ConstantRadiusProvider(0.));

// detector creation
//==================
// the "false" boolean set here indicates that no masking will be computed
final EventDetector detectorMainPart = new
StationToSatMutualVisibilityDetector(
          spacecraftSensorModel, stationModel, correctionModel, false,
maxCheck, threshold);
```

Detection of mutual satellites visibility events

The class `SatToSatMutualVisibilityDetector` contains the algorythm to detect satellites mutual visibility events; to use it, apply the following instructions:

- The detector is added to one of the spacecraft's propagator: this spacecraft becomes the "main" one;
- Each spacecraft is built out of an Assembly object and associated to a SensorProperty/SensorModel construction that represents its antenna;
- The secondary spacecraft's propagation is made in the detector, lead by the main one, to compute its SpacecraftState at each time step. The user shall provide a fully paramettered propagator for him in order to do so, and give it at the detector construction;
- If the maskings computation is wanted, all potentially masking objects set in both sensor models will be tested at each time step. No need to set the same masking object to both : they would by tested twice.

Here is a code sample to create properly a mutual satellite /satellite visibility detector:

```
// We assume here that TWO assemblies have been built (main and secondary),
with a sensor property
// on each of them, on their "mainSpcSensorPart" and "secondarySpcSensorPart"
```

```
part.

// sensor models (the order of those operation is important !)
//============================================================
// main spacecraft model creation
final SensorModel mainSpcModel = new SensorModel(mainAssembly,
mainSpcSensorPart);
// secondary spacecraft model creation
final SensorModel secondarySpcModel = new SensorModel(secondaryAssembly,
secondarySpcSensorPart);
// target addings (the antennas are here considered ponctual)
mainSpcModel.setMainTarget(secondarySpcModel, new
ConstantRadiusProvider(0.));
secondarySpcModel.setMainTarget(mainSpcModel, new
ConstantRadiusProvider(0.));

// propagators
//============
// We assume here that TWO propagators avec been created (main and
secondary),
// and that the secondary one is here fully parametered (forces, initial
state... but NO event detector !!)

// detector creation and adding
//=============================
// the "false" boolean set here indicates that no masking will be computed
final EventDetector detector = new
SatToSatMutualVisibilityDetector(mainSpcModel,
                secondarySpcModel , secondaryPropagator, false, maxCheck,
threshold);

mainPropagator.addEventDetector(detector);
```

Detection of RF visibility events

The class `RFVisibilityDetector` contains the algorythm to detect ground station/satellite RF visibility events. The ground station antenna is represented by a `RFStationAntenna` class and the satellite antenna is represented by a Part of an assembly with `RFAntennaProperty`. See the Link budget properties and models page for more details on the parameters of the station antenna, the satellite antenna and for the link budget computation algorythm.

It is **NOT** possible to use `RFAntennaProperty` for RF LinkBudget computations between two satellites. Only the geometrical aspect (using a `Sensor Model`) is available.

Getting Started

Modèle:SpecialInclusion prefix=$theme sub section="GettingStarted"/

Contents

Interfaces

Interface	Summary	Javadoc
EventDetector	This interface represents an event finder.	EventDetector
LocalRadiusProvider	Provider for local radius.	LocalRadiusProvider

Classes

Class	Summary	Javadoc
ApparentElevationDetector	This class handles satellite apparent elevation (apparent raising and setting for a terrestrial viewpoint) events.	ApparentElevationDetector
ElevationDetector	This class handles satellite elevation (raising and setting for a terrestrial viewpoint) events.	ElevationDetector
ExtremaElevationDetector	This class handles events representing the reaching of the minimal or maximal elevation in a given topocentric frame (local frame of a station).	ExtremaElevationDetector
GroundMaskElevationDetector	This class handles satellite elevation (raising and setting for a terrestrial viewpoint) events with respect to an azimuth-elevation mask.	GroundMaskElevationDetector
VisibilityFromStationDetector	This class handles events representing the satellite apparent entering in a station's field of view.	VisibilityFromStationDetector
SatToSatMutualVisibilityDetector	This class handles events representing the mutual visibility between two spacecraft's sensors.	SatToSatMutualVisibilityDetector

StationToSatMutualVisibilityDetector	This class handles events representing the mutual visibility between a spacecraft's sensors and a station antenna.	StationToSatMutualVisibilityDetector
RFVisibilityDetector	This class handles RF visibility events (when the link budget between a ground station antenna and a satellite antenna exceeds a threshold value).	RFVisibilityDetector
Constant radius provider	Provider for constant radius body.	ConstantRadiusProvider
Variable radius provider	Provider for variable radius body (ex: one axis ellipsoid).	VariableRadiusProvider

Tutorials

Tutorial 1

Modèle:SpecialInclusion prefix=$theme sub section="Tuto1"/

Tutorial 2

Modèle:SpecialInclusion prefix=$theme sub section="Tuto2"/

Tips & Tricks

Modèle:SpecialInclusion prefix=$theme sub section="Tips"/

Menu de navigation

Outils personnels

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 V4.15 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 avril 2018 à 09:02.

- Politique de confidentialité
- À propos de Wiki
- Avertissements

-