# Filling out the Geometry Orbiter Class

The *Geometry_Orbiter* class provides geometric values related to an orbiting or flyby spacecraft observation. The values in this class are specific to a single reference frame at a single reference time (or time range) for at most a single target. You may repeat this class with differing values for one or all of those, if appropriate to the observation being labelled.

## <geometry_reference_time>

*OPTIONAL*

This is the UTC time for which the values in this class have been calculated. It must be in the standard *YYYY-MM-DD***T***hh:mm:ss.sss***Z**" format.

Either this attribute, or the *geometry_start_time*/*geometry_stop_time* pair is required. You are not prevented from including both, but that would be rather confusing and there is no place in this class for an explanatory comment - so avoid it.

## <geometry_start_time> and <geometry_stop_time>

*OPTIONAL*

These are both optional, but if you use one you must also include the other. (See also the note above for *geometry_reference_time*.) Use the start/stop pair when you have an extended observation for which you plan to provide pairs of geometric values corresponding to these times in the other subclasses of <*Geometry_Orbiter*>.

These must be UTC times in the standard *YYYY-MM-DD***T***hh:mm:ss.sss***Z** format.

## <geometry_reference_time_tdb>

*OPTIONAL*

This should be the Barycentric Dynamical Time (or Ephemeris Time), equivalent to the <*geometry_reference_time_UTC*> listed above. This is defined in the SPICE toolkit sense of a number of seconds since the J2000 epoch.

## <Orbiter_Identification>

*OPTIONAL*

Use this class to identify geometric elements and reference points to be used in the *Geometry_Orbiter* subclasses that follow.

### <Central_Body_Identification>

*OPTIONAL*

If the target of the following geometry is orbiting another body, that body is the "central body" and should be identified using this class. It is filled out identically to the class of the same name on the Filling Out the Image_Display_Geometry Class page.

## <Geometry_Target_Identification>

*OPTIONAL*

This class locally defines the concept of "target" to be the thing on the "to" end of the various vectors that follow in this *Geometry_Orbiter* class. It may or may not be the same as the target of the product as a whole. It is filled out identically to the class of the same name in the *Image_Display_Geometry* class as seen here: Filling Out the Image_Display_Geometry Class.

## <Coordinate_System_Identification>

*OPTIONAL*

This class is used to identify a "coordinate system" in the NAIF SPICE toolkit sense - both a reference frame and a well-defined origin, with a specific type of coordinates.

*<coordinate_system_type>*

*REQUIRED*

You must indicate the type of coordinates to be applied, which must be one of the following values:

- Azimuth-Elevation
- Cartesian
- Planetocentric
- Planetodetic
- Planetographic
- Spherical

*<coordinate_system_time_utc>*

*OPTIONAL*

If your coordinate system has a time dependence, this attribute should hold the relevant instantiation time in UTC, in the standard *YYYY-MM-DD***T***hh:mm:ss.ss***Z** format.

*<comment>*

*OPTIONAL*

Here is a place to add some clarifying explanation for uncommon coordinate systems.

*<Coordinate_System_Origin_Identification>*

*REQUIRED*

This class indicates the fixed origin ([0,0,0] point, i.e.) of the coordinate system. Apart from the class name, the content is filled out identically to the *Geometry_Target_Identification* class in the *Image_Display_Geometry* class. That is, its attributes are:

- *body_spice_name*
- *name*
- *pds:Internal_Reference*

*<Reference_Frame_Identification>*

*REQUIRED*

This class indicates the "orthogonal axes" part of the coordinate system definition. It is filled out identically to the class of the same name in the *Image_Display_Geometry* class.

## <Pixel_Dimensions>

*OPTIONAL*

This class provides various ways of translating a pixel field of view to physical dimensions. You must provide units for these attributes as appropriate.

### <pixel_field_of_view_method>

*REQUIRED*

This attribute indicates how the pixel FOV values, following, were derived. It must have one of the following values:

- **average**: The pixel FOV varies across the image; the value stated are the average FOV.
- **central pixel**: The pixel FOV varies across the image; the value stated represents the FOV of the center pixel.
- **constant**: The pixel FOV does not vary across the image.

### <horizontal_pixel_field_of_view>

*OPTIONAL*

This attribute gives the horizontal (as defined by the *<Display_Direction>* class) FOV of a single pixel in terms of its angular size. You must specify angular units (typically "deg") for this value.

### <vertical_pixel_field_of_view>

*OPTIONAL*

This attribute gives the vertical (as defined by the *<Display_Direction>* class) FOV of a single pixel in terms of its angular size. You must specify angular units (typically "deg") for this value.

### <Pixel_Size_Projected>

*OPTIONAL*

This class provides the size, in units of length, of a single pixel projected onto the surface of (or at the distance of) the relevant geometry target.

*<reference_location>*

*OPTIONAL*

This attribute identifies the point on the target at which the following pixel values are calculated. Either this or *<distance>* must be specified. It must have one of the following values:

- Boresight Intercept Point
- Constant
- Subspacecraft Point
- Target Center

*<distance>*

*OPTIONAL*

Distance to which the pixel is projected. This is particularly useful when your target object occupies a small number of pixels rather than the entire image frame. Either this or *<reference_location>* must be specified. You must provide units of length for this value.

*<horizontal_pixel_footprint>*

*REQUIRED*

This attribute gives the horizontal (as defined in the associated *Display_Direction* class) extent of a single pixel at the *reference_location* or *distance*, in units of length.

*<vertical_pixel_footprint>*

*REQUIRED*

This attribute gives the vertical (as defined in the associated *Display_Direction* class) extent of a single pixel at the *reference_location* or *distance*, in units of length.

## <Distances>

*OPTIONAL*

This class is a container for other classes used to list scalar distance measurements of various types. These classes collect point-to-point distance measurements (that is, magnitudes rather than vectors) for various named intervals of interest, as well as providing a class for specifying arbitrary distance measurements as needed. Unless otherwise stated, distances are measured center-to-center, and references to "target" and "central body" refer to the objects identified previously in this *<Geometry_Orbiter>* class.

To avoid a *lot* of tedious repetition, these class and attribute descriptions are somewhat abbreviated. In general, the containing class will be described briefly, and the individual distance attributes will merely be listed - their names being self-explanatory.

### General Notes:

- All distance attributes are optional in their respective classes; none may be repeated.
- All distances are considered positive.
- All distances are named by their start and end points; terms like "target" and "central body" are as defined for this *Geometry_Orbiter* class.
- Unless otherwise indicated by their names, all measurements are considered to be center-to-center.
- All values require units of length for each value, though that unit can vary among attributes in a class. For example:

```
    <spacecraft_target_center_distance unit="km">15.097</
spacecraft_target_center_distance>
    <target_geocentric_distance unit="AU">17.9</target_geocentric_distance>
```

### <comment>

*OPTIONAL*

Here is a place to provide any additional explanatory text that might be apropos.

## <Distances_Specific>

*OPTIONAL*

This container class lists specific named distances often used for searching through data collections from a single source. Unlike other distance classes, these attributes may be listed in any order.

*Distances Included:*

- *spacecraft_geocentric_distance*
- *spacecraft_heliocentric_distance*
- *spacecraft_central_body_distance*
- *spacecraft_target_center_distance*
- *spacecraft_target_boresight_intercept_distance*
- *spacecraft_target_subspacecraft_distance*
- *target_geocentric_distance*
- *target_heliocentric_distance*
- *target_ssb_distance* - "SSB" stands for "Solar System Barycenter"

## <Distances_Min_Max>

*OPTIONAL*

This container class lists minimum and maximum distances often used for constraining searches through data collections from a single source. You are required to provide these values in min/max pairs. If you don't have min/max pairs for one or more values, use the *<Distances_Specific>* class and the corresponding single attribute.

*Distances Included:*

- *minimum_spacecraft_geocentric_distance*
- *maximum_spacecraft_geocentric_distance*
- *minimum_spacecraft_heliocentric_distance*
- *maximum_spacecraft_heliocentric_distance*
- *minimum_spacecraft_central_body_distance*
- *maximum_spacecraft_central_body_distance*
- *minimum_spacecraft_target_center_distance*
- *maximum_spacecraft_target_center_distance*

- *minimum_spacecraft_target_boresight_intercept_distance*
- *maximum_spacecraft_target_boresight_intercept_distance*
- *minimum_spacecraft_target_subspacecraft_distance*
- *maximum_spacecraft_target_subspacecraft_distance*
- *minimum_target_geocentric_distance*
- *maximum_target_geocentric_distance*
- *minimum_target_heliocentric_distance*
- *maximum_target_heliocentric_distance*
- *minimum_target_ssb_distance* - "SSB" stands for "Solar System Barycenter"
- *maximum_target_ssb_distance* - "SSB" stands for "Solar System Barycenter"

## <Distances_Start_Stop>

*OPTIONAL*

This container class lists distances often used for constraining searches through data collections from a single source calculated at the *<geometry_start_time>* and *<geometry_stop_time>* defined at the top of this *Geometry_Orbiter* class. You are required to provide these values in start/stop pairs. If you don't have start/stop pairs for one or more values, use the *<Distances_Specific>* class and the corresponding single attribute.

*Distances Included:*

- *start_spacecraft_geocentric_distance*
- *stop_spacecraft_geocentric_distance*
- *start_spacecraft_heliocentric_distance*
- *stop_spacecraft_heliocentric_distance*
- *start_spacecraft_central_body_distance*
- *stop_spacecraft_central_body_distance*
- *start_spacecraft_target_center_distance*
- *stop_spacecraft_target_center_distance*
- *start_spacecraft_target_boresight_intercept_distance*
- *stop_spacecraft_target_boresight_intercept_distance*
- *start_spacecraft_target_subspacecraft_distance*
- *stop_spacecraft_target_subspacecraft_distance*
- *start_target_geocentric_distance*
- *stop_target_geocentric_distance*
- *start_target_heliocentric_distance*
- *stop_target_heliocentric_distance*
- *start_target_ssb_distance* - "SSB" stands for "Solar System Barycenter"
- *stop_target_ssb_distance* - "SSB" stands for "Solar System Barycenter"

## <Distance_Generic>

*OPTIONAL*

This class is used to define a distance measurement between a starting point, called the "observer", and an ending point, called the "target". You must explicitly identify the "observer" and "target", as well as providing the distance. You may repeat this class as many times as needed to define various distances.

*<Observer_Identification>*

*REQUIRED*

The class has the same content, and is filled out the same way, as the *<Geometry_Target_Identification>* class in the *Image_Display_Geometry* class.

*<Geometry_Target_Identification>*

*REQUIRED*

This class is filled out identically to the class of the same name in the *Image_Display_Geometry* class.

*<distance>*

*REQUIRED*

The center-to-center, positive distance between the observer and the target. You must provide units of length for this value.

## **<Surface_Geometry>**

*OPTIONAL*

This class is a container for other classes used to list surface coordinates of various types. These classes collect surface orientation for various named angles of interest either as single values, min/max ranges, or start/stop values. Concepts like "latitude" and "longitude" are interpreted with respect to the specific target in question, as identified previously in this *<Geometry_Orbiter>* class.

To avoid some tedious repetition, these class and attribute descriptions are somewhat abbreviated. In general, the containing class will be described briefly, attributes unique to each class will be described as usual, the individual angle attributes will merely be listed with abbreviated descriptions mainly concerning valid ranges.

## General Notes:

- All attributes are optional in their respective classes unless otherwise indicated; none may be repeated.
- All angles require units of angle for each value. Though that unit can vary among attributes in a class, typically all values will be given in units of degrees. For example:

```
<subsolar_azimuth unit="deg">15.097</subsolar_azimuth>
<spacecraft_longitude unit="deg">17.9</spacecraft_longitude>
```

## <comment>

*OPTIONAL*

Here is a place to provide any additional explanatory text that might be apropos.

## <Surface_Geometry_Specific>

*OPTIONAL*

This container class lists specific named distances as well as surface intercept points.

### *<Pixel_Intercept>*

*OPTIONAL*

This class defines the latitude and longitude on the surface of the target for the projection of a single, specified pixel. It may be repeated. Use it when the points involved do *not* represent the observation footprint.

<reference_pixel_location>

*OPTIONAL*

Either this attribute or the <*Reference_Pixel*> class, following, must be specified. This attribute is a human-readable indication of where the reference pixel is, when the image is displayed as indicated in the *Image_Display_Geometry* class. It must have one of the following values:

- Center
- Lower Left Corner
- Lower Right Corner
- Upper Left Corner
- Upper Right Corner

<Reference_Pixel>

*OPTIONAL*

Either <*reference_pixel_location*>, above, or this class must be specified. This class provides a more precise indication of where the reference pixel occurs in the image. It is filled out identically to the class of the same name in the *Image_Display_Geometry* class.

<pixel_latitude>

*REQUIRED*

Planetocentric latitude of the pixel projected onto the target, in the range ±90°

<pixel_longitude>

*REQUIRED*

Planetocentric longitude of the pixel projected onto the target, in the range 0-360°

### *<Footprint_Vertices>*

*OPTIONAL*

If you have a series of pixel intercept points that define the footprint of the observation, use this wrapper class to list them. It contains a list of two or more <*Pixel_Intercept*> classes that should be filled out the same way as described above. You should list the pixels either in clockwise or counterclockwise sequence around the perimeter of the footprint, so that line segments drawn between successive pixels (the last connecting back to the first) will produce the footprint on the image.

<Pixel_Intercept>

*REQUIRED*

You must have at least two occurrences of this class; you may have as many as needed to define the footprint.

*Angles Included:*

- *subsolar_azimuth*
- *subsolar_latitude*
- *subsolar_longitude*
- *subspacecraft_azimuth*
- *subspacecraft_latitude*
- *subspacecraft_longitude*

## <Surface_Geometry_Min_Max>

*OPTIONAL*

This class lists min/max pairs of geometric angles calculated for the target surface. You must provide both minimum and maximum for any particular angle. If you have geometry for specific points rather than ranges, use the <*Surface_Geometry_Specific*> class to provide those values.

*Angles Included:*

- *minimum_latitude*
- *maximum_latitude*
- *minimum_longitude*
- *maximum_longitude*
- *minimum_subsolar_azimuth*
- *maximum_subsolar_azimuth*
- *minimum_subsolar_latitude*
- *maximum_subsolar_latitude*
- *minimum_subsolar_longitude*
- *maximum_subsolar_longitude*
- *minimum_subspacecraft_azimuth*
- *maximum_subspacecraft_azimuth*
- *minimum_subspacecraft_latitude*
- *maximum_subspacecraft_latitude*
- *minimum_subspacecraft_longitude*
- *maximum_subspacecraft_longitude*

## <Surface_Geometry_Start_Stop>

*OPTIONAL*

If you have surface geometry calculated at the *geometry_start* and *geometry_stop* times included in this <*Geometry_Orbiter*> class, use this class to list the corresponding values. You must list these angles in start/stop pairs. If you have geometry for specific points rather than time ranges, use the <*Surface_Geometry_Specific*> class to provide those values.

*<lat_long_method>*

*OPTIONAL*

This attribute is required if your *Surface_Geometry_Start_Stop* class includes start/stop latitude or longitude. It must have one of the following values:

**center**
The latitude/longitude values are those at the center of the field of view at the beginning and end of the observation
**median**
The latitude/longitude values are the median values at the beginning and end of the observation.
**mean**
The latitude/longitude values are the mean values at the beginning and end of the observation.

*Angles Included:*

- *start_latitude*
- *stop_latitude*
- *start_longitude*
- *stop_longitude*
- *start_subsolar_azimuth*
- *stop_subsolar_azimuth*
- *start_subsolar_latitude*
- *stop_subsolar_latitude*
- *start_subsolar_longitude*
- *stop_subsolar_longitude*
- *start_subspacecraft_azimuth*
- *stop_subspacecraft_azimuth*
- *start_subspacecraft_latitude*
- *stop_subspacecraft_latitude*
- *start_subspacecraft_longitude*
- *stop_subspacecraft_longitude*

## **<Illumination_Geometry>**

*OPTIONAL*

This class is a container for other classes used illumination angles of various types. These classes collect various named angles of interest either as single values, min/max ranges, or start/stop values. Angle names should be self-explanatory.

To avoid some tedious repetition, these class and attribute descriptions are somewhat abbreviated. In general, the containing class will be described briefly, attributes unique to each class will be described as usual, the individual angle attributes will merely be listed with abbreviated descriptions mainly concerning valid ranges where appropriate.

## General Notes:

- The container classes themselves may not be repeated.
- All attributes are optional in their respective classes unless otherwise indicated; none may be repeated.
- All angles require units of angle for each value. Typically all values will be given in units of degrees. For example:

```
        <emission_angles unit="deg">15.097</emission_angle>
        <start_solar_elongation unit="deg">17.9</start_solar_elongation>
```

## <comment>

*OPTIONAL*

Here is a place to provide any additional explanatory text that might be appropriate.

## <Illumination_Specific>

*OPTIONAL*

This class provides illumination angles at a specific point in the image. This point must be specified in exactly one way: either through the *reference_location* attribute;
the *reference_pixel_location* attribute; or the <*Reference_Pixel*> subclass.

*<reference_location>*

*OPTIONAL*

This defines a reference location in terms of some property of or point in the field of view. If present, this must have one of the following values:

- **Boresight Intercept Point**
- **Constant**
- **Subspacecraft Point**
- **Target Center**

*<reference_pixel_location>*

*OPTIONAL*

This defines a reference location in terms of a particular pixel location in the image when displayed according to the display settings established for this *Geometry* class. If present, it must have one of the following values:

- **Center**

- **Lower Left Corner**
- **Lower Right Corner**
- **Upper Left Corner**
- **Upper Right Corner**

*<Reference_Pixel>*

*OPTIONAL*

This class defines a reference location in terms of pixel subscripts in the image array. It is filled out identically to the class of the same name in the *Image_Display_Geometry* class.

*Angles Included:*

At least one of these must be present.

- *emission_angle*
- *incidence_angle*
- *phase_angle*
- *solar_elongation*

## <Illumination_Min_Max>

*OPTIONAL*

This class gives illumination angles as min/max ranges. You must supply these values in min/max pairs. If you have single values, use the <*Illumination_Specific*> class, above.

*Angles Included:*

- *minimum_emission_angle*
- *maximum_emission_angle*
- *minimum_incidence_angle*
- *maximum_incidence_angle*
- *minimum_phase_angle*
- *maximum_phase_angle*
- *minimum_solar_elongation*
- *maximum_solar_elongation*

## <Illumination_Start_Stop>

*OPTIONAL*

This class lists illumination angles calculated for the corresponding *geometry_start_time* and *geometry_stop_time* of this <*Geometry_Orbiter*> class. You must provide start/stop time pairs for this class.

*Angles Included:*

- *start_emission_angle*

- *stop_emission_angle*
- *start_incidence_angle*
- *stop_incidence_angle*
- *start_phase_angle*
- *stop_phase_angle*
- *start_solar_elongation*
- *stop_solar_elongation*

## <Vectors>

*OPTIONAL*

This class is a container for other classes used to define vectors for various positions and velocities of interest. Two of the optional subclasses collect vectors in specific coordinate systems types - Cartesian and Planetocentric. The third optional subclass provides a mechanism for defining position and velocity vectors outside those included in the first two classes. Unless otherwise stated, positions and velocities are measured center-to-center, and references to "target" and "central body" refer to the objects identified previously in this <*Geometry_Orbiter*> class.

To avoid a *lot* of tedious repetition, these class and attribute descriptions are somewhat abbreviated. In general, the containing class will be described briefly, and the individual vectors classes will merely be listed - their names being self-explanatory. All vectors of similar type (all Cartesian velocity vectors, for example) will have the same attribute content.

### General Notes:

The various vectors in the Geometry Discipline dictionary are derived from common parent classes, providing consistent structure and terminology. The "specific' vectors listed below have been defined for the set of vectors that appear in the majority of orbiter/flyby data products. For these vectors the start and end points ("observer" and "target", in general terms) and coordinate system are part of the definition of the vector itself. You should use these specific vectors whenever they might reasonably apply, as they provide a common nomenclature for these values that users can key into across the archive. When you need to specify a vector for a quantity outside this standard set, use the generic vectors, which allow you to specify observer, target, and coordinate system.

The details of the structure of each family of vectors in the Geometry Discipline dictionary are summarized on the Filling Out the Geometry Vector Classes page.

**Additional Notes:**

Within each of the wrapper classes and unless otherwise noted:

- All vectors are optional.
- Vectors may be in any order (this is *not* the usual case for PDS4 label classes and attributes).

**Note:** *Vectors can also be repeated an arbitrary number of times within their respective classes. This seems like a bad idea for the non-Generic classes, on the face of it. If you have a reason for duplicating vectors inside this class, please use the* <comment> *attribute to provide some explanation for this unusual situation.*

## <comment>

*OPTIONAL*

If there is any additional explanation needed or desired regarding the contents of the following classes, here's a place for it. Note that there are no comment fields in the individual wrapper classes

## <Vectors_Cartesian_Specific>

*OPTIONAL*

If present, this class must contain at least one of the following vectors. The abbreviation "SSB" stands for *Solar System Barycenter*.

The *Position* vectors below are filled out as described for the *Vector_Cartesian_Position_Generic* class on the Filling Out the Geometry Vector Classes page, with the exceptions noted for specific vectors vs. generic ones.

The *Velocity* vectors below are filled out as described for the *Vector_Cartesian_Velocity_Generic* class on the Filling Out the Geometry Vector Classes page, with the exceptions noted for specific vectors vs. generic ones.

*Vectors Included:*

- *Vector_Cartesian_Position_Central_Body_To_Spacecraft*
- *Vector_Cartesian_Position_Central_Body_To_Target*
- *Vector_Cartesian_Position_Earth_To_Central_Body*
- *Vector_Cartesian_Position_Earth_To_Spacecraft*
- *Vector_Cartesian_Position_Earth_To_Target*
- *Vector_Cartesian_Position_SSB_To_Central_Body*
- *Vector_Cartesian_Position_SSB_To_Spacecraft*
- *Vector_Cartesian_Position_SSB_To_Target*
- *Vector_Cartesian_Position_Spacecraft_To_Target*
- *Vector_Cartesian_Position_Sun_To_Central_Body*
- *Vector_Cartesian_Position_Sun_To_Spacecraft*
- *Vector_Cartesian_Position_Sun_To_Target*
- *Vector_Cartesian_Velocity_Spacecraft_Relative_To_Central_Body*
- *Vector_Cartesian_Velocity_Spacecraft_Relative_To_Earth*
- *Vector_Cartesian_Velocity_Spacecraft_Relative_to_SSB*
- *Vector_Cartesian_Velocity_Spacecraft_Relative_To_Sun*
- *Vector_Cartesian_Velocity_Spacecraft_Relative_To_Target*
- *Vector_Cartesian_Velocity_Target_Relative_To_Central_Body*
- *Vector_Cartesian_Velocity_Target_Relative_To Earth*
- *Vector_Cartesian_Velocity_Target_Relative_To_SSB*
- *Vector_Cartesian_Velocity_Target_Relative_To_Spacecraft*
- *Vector_Cartesian_Velocity_Target_Relative_To_Sun*

## <Vectors_Planetocentric_Specific>

*OPTIONAL*

If present, this class must contain at least one of the following vectors.

The *Position* vectors below are filled out as described for the *Vector_Planetocentric_Position_Generic* class on the Filling Out the Geometry Vector Classes page, with the exceptions noted for specific vectors vs. generic ones.

The *Velocity* vectors below are filled out as described for the *Vector_Planetocentric_Velocity_Generic* class on the Filling Out the Geometry Vector Classes page, with the exceptions noted for specific vectors vs. generic ones.

*Vectors Included:*

- *Vector_Planetocentric_Position_Central_Body_To_Spacecraft*
- *Vector_Planetocentric_Position_Central_Body_To_Target*
- *Vector_Planetocentric_Position_Spacecraft_To_Target*
- *Vector_Planetocentric_Velocity_Spacecraft_Relative_To_Target*
- *Vector_Planetocentric_Velocity_Target_Relative_To_Central_Body*
- *Vector_Planetocentric_Velocity_Target_Relative_To_Spacecraft*

## <Generic_Vectors>

*OPTIONAL*

This class is used when the start point, end point, or both are not covered by the vectors of the preceding specific classes. Five generic vector types are provided. You may repeat these classes as needed for however many vectors you need to define.

The contents and procedures for filling out the generic vector classes are detailed on the Filling Out the Geometry Vector Classes page.

*Vectors Included:*

- *Vector_Cartesian_Acceleration_Generic*
- *Vector_Cartesian_Position_Generic*
- *Vector_Cartesian_Velocity_Generic*
- *Vector_Planetocentric_Position_Generic*
- *Vector_Planetocentric_Velocity_Generic*