**Navigation events
April 2018 – updated June 2018**

This note describes some high level requirements / desired features for the navigation events message.

Made using:

* Initial ideas,
* Fruitful discussion at the spring 2018 meeting
* Ideas from Frank that he sent after the meeting (the introduction based on “Frank’s thoughts”)
* Concrete solutions from Fran

# Introduction

**Navigation Events** can cover all possible cases of events which occur or are planned for a certain mission. They are often accompanied by certain parameters. **Navigation Event Types** range from the classical orbit events such as NodeCossing, PenumbraStart and AOS/LOS to very mission specific and operational activity related events such “StartMomentumBIAS”. Navigation Events are often augmented with certain parameters, such as OrbitNumber, EclipseDepth or Elevation and PredictedDopplerShift. While it is relatively straightforward to come up with a list of the classical events and some of their obvious parameters, it is impossible to standardize all the other events. For a standard on “**Navigation Event Messages**” (NEM) to be successful (agreeable and applicable), it needs to cater for extension and tailoring. On the other hand, the standard also needs to limit the flexibility by offering a standard directly usable by “simple and standard enough” missions.

# (Pseudo) requirements

## Main requirement

The NEM enables exchange of navigation events information consisting of at least:

* Date / time
* Event name (that is to say the “type” of event)
* Explanatory information: parameters that are necessary to fully understand the event without ambiguity

## Message formats: both KVN (OEM style) AND XML

The NEM should be available in both XML and KVN formats.

Some users prefer XML. Some users would like to have KVN available.

Arguments in favor of one or the other are:

KVN could be more compact and easier to implement or to use.

XML format is able to capture a larger number of features (optional parameters, events timestamps function of other events …).

Then:

KVN could be limited to a subset of features to make it simple enough.

An example of limitation could be: event timestamps: absolute date only.

The full set of features would only be available in XML.

## No need for an ICD for standard uses, and if possible for less standard ones as well

Messages should be (as much as possible) self-contained.

An ICD should not be necessary to be able to process a particular message.

## list of Predefined events

A list of predefined events must exist. It defines the most used and most standard events in a particular context (for instance Earth missions).

It can also be used as examples to build user-defined events, or user-modified events.

The predefined list contains everything that is necessary to understand each event:

* Clear description (text)
* Name / type
* Parameters: types, description, units …

The standard should encourage users of the message:

* to use predefined events,
* or to use predefined events with only slight modifications.
* Or to design events that are as similar as possible to predefined events.

Reason: the standard is designed to ease exchange of events information. The predefined events provide names, meaning, typical parameters… all this helps understanding what the events really are. In addition, reusing the same events in various missions limits possible misunderstanding about what some events really mean.

## Data types (parameters) referenced in an events message should be defined without ambiguity

All the ancillary data (also called “parameters”) necessary to clearly “understand” an event should be unambiguously defined: meaning (description), type (integer, string …), range …

Basic types such as integer; real, string may be enough.

More complex type may be necessary (TBC).

## Limit to one spacecraft? probably noT

One events message may contain events related to one or more spacecraft.

However, many missions consist of only one spacecraft.

If events in a same message could be related to any one of a group of spacecraft (constellation, cluster…), then the name of the spacecraft the event is related to should be given in the event’s contents.

## Events timestamps

In events navigation message, timestamps can be:

* Absolute date related to a unique time scale
* Time in some time scale relative to an epoch possibly given in a different time scale
* Time in some time scale relative to an epoch defined by another event
* …

All options are TBC

Not all of the options above may be possible.

## Extensibility – 1

The standard enables the user to define new events.

It is hopeless to design a standard containing all imaginable events for all future and past missions.

So the rationale is to satisfy most common needs and to let the user define his own and possibly specific events.

Example:

The creator of the message would like to add an event:

ANGLE\_BETWEEN spacecraft1 and spacecraft2 as seen from spacecraft0 = X degrees

## Extensibility – 2

The standard enables the user to add information (parameters) to existing events.

This is a particular case where the event is supposed to be already defined, but some information is missing. For instance, the user of the message would like to add the value of the argument of latitude to all events. Not all users want that so the argument of latitude is probably not among the parameters of the template events.

An other case would be to add the name of the spacecraft as an additional parameters to all events.

There are 2 options:

* Either it is considered that any change forces the user to defined new events
* Or limited additional information can be added to existing events by design

TBD

# Predefined events

A list of events is provided as a starting point.

## Summary of the list given as an example

4 categories:

* Orbital events
* Satellite events
* Stations events
* Mission events

List will have to be updated, simplified, etc …

## parameters associated to an event

In the description, the type (or name) of event is sometimes called [X]something.

Example : [X]DEG\_AOS

X is the value for the elevation defining the event: AOS when elevation is X degrees.

For the events message, another way is preferred.

The type of events would be AOS, X would be a parameter value

Other parameters are sometimes defined as for an event related to a ground station.

The type is a complex type (or structure) gathering specific information.

Such feature may exist or not for the NEM.

**Structure STATION**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name**  | **Type**  | **Optional/Required**  | **Description**  |
| EVENT\_TYPE  | A\_STATION\_EVENT : String  | Required  | ***Configurable item using STATION\_EVENT\_CONFIG.XML***  |
| PARAMETERS  | A\_POINTING  | Required  | A\_POINTING : Station event characteristics  |

**Type A\_POINTING**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name**  | **Type**  | **Optional/Required**  | **Description**  |
| STATION  | A\_STATION\_MNEMO : String  | Required  | Station mnemonic.***Configurable item using STATION\_MNEMO\_CONFIG.XML***  |
| EARTH\_ANTENNA \* n  | AN\_EARTH\_ANTENNA\_MNEMO : String  | Required  | Mnemonic of the Earth antenna(s).***Configurable item using EARTH\_ANTENNA\_MNEMO\_CONFIG.XML***  |
| SAT\_ANTENNA  | A\_SAT\_ANTENNA\_MNEMO : String  | Optional  | Mnemonic of the stellite antenna.***Configurable item using SAT\_ANTENNA\_MNEMO\_CONFIG.XML***  |
| ELEVATION  | AN\_ELEVATION : Float [-10.0 .. 90.0]  | Required  | Elevation angle (degree).**Attribute : unit** - "deg" - RequiredAN\_ELEVATION : A [-10 ; 90] angle in degree used for an elevation  |
| AZIMUTH  | AN\_ANGLE\_IN\_DEG\_0\_360 : Float [0.0 .. 360.0]  | Required  | Azimuth angle (degree).**Attribute : unit** - "deg" - RequiredAN\_ANGLE\_IN\_DEG\_0\_360 : A [0 ; 360] angle (can be used for azimut)  |
| TYPE  | Enumeration [ "CONT" "INTERM" ]  | Optional  | State = continuous or intermittent. |

# Definition of events - CSS (cross support services) definition

## What is defined:

* Parameters
	+ int
	+ double
	+ string
	+ time
	+ etc…
	+ + complex types
* Event attributes:
	+ type => category
	+ user => for whom
	+ latestOffset => tolerance on the event time
	+ earliestOffset
	+ identifier => for references

Time can be:

* absolute
* relative (to a reference epoch)
* relative to another event

## XML example from Fran’s instantiation examples

Example 1: orbital event

 <event name="**TRUE\_ANOMALY**" type="**ORBITAL\_EVENT**">

 <EPOCH xsi:type="ndm:absoluteEventEpochType" TIME\_SYSTEM="TAI">

 <ABSOLUTE\_TIME>2018-04-11T00:00:10</ABSOLUTE\_TIME>

 </EPOCH>

 <DURATION units="s">0</DURATION>

 <identifier>ID\_EVENT</identifier>

 <earliestOffset>0</earliestOffset>

 <latestOffset>0</latestOffset>

 <parameters >

 <parameter xsi:type="ndm:doubleParameterType" name="true\_anomaly">

 <value>45</value>

 <unit>deg</unit>

 </parameter>

 </parameters>

 </event>

Example 2: station event

 <event name="**GROUND\_STATION\_AOS**" type="**STATION\_EVENT**">

 <EPOCH xsi:type="ndm:absoluteEventEpochType" TIME\_SYSTEM="TT">

 <ABSOLUTE\_TIME>2018-04-11T00:00:10</ABSOLUTE\_TIME>

 </EPOCH>

 <DURATION units="s">0</DURATION>

 <identifier>ID\_EVENT</identifier>

 <earliestOffset>0</earliestOffset>

 <latestOffset>0</latestOffset>

 <parameters>

 <parameter xsi:type="ndm:doubleParameterType" name="elevation">

 <value>10</value>

 <unit>deg</unit>

 </parameter>

 <parameter xsi:type="ndm:stringParameterType" name="station">

 <value>Kourou</value>

 </parameter>

 <parameter xsi:type="ndm:stringParameterType" name="antenna">

 <value>antenna1</value>

 </parameter>

 </parameters>

 </event>

Example 3: orbital event – time relative to another event

 <event name="**TRUE\_ANOMALY**" type="**ORBITAL\_EVENT**">

 <EPOCH xsi:type="ndm:relativeEventEpochType" TIME\_SYSTEM="TDB">

 <RELATIVE\_TIME units="s">10.2</RELATIVE\_TIME>

 <EVENT\_IDENTIFIER>**ID007**</EVENT\_IDENTIFIER>

 </EPOCH>

 <identifier>ID008</identifier>

 <earliestOffset>0</earliestOffset>

 <latestOffset>0</latestOffset>

 <parameters >

 <parameter xsi:type="ndm:doubleParameterType" name="true\_anomaly">

 <value>45</value>

 <unit>deg</unit>

 </parameter>

 </parameters>

 </event>

## What could it be in KVN ?

### Version 1: KVN-like

Example :

EVENT\_START

NAME = TRUE\_ANOMALY

EPOCH = 2018-04-11T00:00:10

VALUE = 45 [deg]

EVENT\_STOP

Is it really useful ?

### Version 2: set of lines

2018-04-11T00:00:10 TRUE\_ANOMALY 45

2018-04-11T00:00:10 GROUND\_STATION\_AOS **10** Kourou

But:

* The order must be predefined
* Units are implicit
* Mix of strings, real values?
* String values with spaces?
* Add quotes to strings?
* What if some parameters are optional? how do we know which ones?
* What if some information has to be added?

# Message structure

This mainly comes from Frank’s thoughts.

Each Navigation Event Message structures the information in three distinct areas:

* The NEM Header
* The MetaData, which carries all information about the NEM itself, such as GenerationDate, Originator or PeriodCovered.
* The list of Navigation events together with their augmenting data (attributes and parameters).

### NEM Message Header

For consistency with other CCSDS NAV standards, the Message Header elements are named in UPPER\_CASE notation.

* CCSDS\\_NEM\\_VERS: Format version in the form of ‘x.y’, where ‘y’ is incremented for corrections and minor changes, and ‘x’ is incremented for major changes.
* COMMENT:
* CREATION\\_DATE: File creation date/time in UTC. (For format specification, see TBD)
* ORIGINATOR: Creating agency. The value for the “ORIGINATOR” keyword “should” come from the SANA registry.
* MESSAGE\\_ID: ID that uniquely identifies a message from a given originator. The format and content of the message identifier value are at the discretion of the originator.

With the exception of COMMENT and MESSAGE\\_ID, all message header elements are mandatory.

### NEM MetaData

For consistency with other CCSDS NAV standards, the MetaData elements are named in UPPER\_CASE notation.

* COMMENT
* ORIGINATOR\\_POC: Free text field containing Programmatic or Technical Point-of-Contact (PoC) for NEM
* ORIGINATOR\\_PHONE: Free text field containing PoC phone number
* ORIGINATOR\\_POSITION: Free text field containing contact position of the PoC
* ORIGINATOR\\_ADDRESS: Free text field containing Technical PoC information for ACM creator (suggest email, website, or physical address, etc.)
* OBJECT\\_NAME:
* OBJECT\\_ID:
* TIME\\_SYSTEM: time scale in which the event epochs covered by the NEM are defined.
* START\\_TIME: This keyword shall specify the start time of the total time span covered by the NEM, specified in the time scale given by TIME\\_SYSTEM. For format specification, see TBD
* STOP\\_TIME: This keyword shall specify the stop time of the total time span covered by the NEM , specified in the time scale given by TIME\\_SYSTEM. For format specification, see TBD
* EPOCH\\_TZERO: TBC

With the exception of OBJECT\\_NAME, OBJECT\\_ID, TIME\\_SYSTEM, START\\_TIME and STOP\\_TIME all MetaData elements are optional.

### NEM Events

A Navigation Event Message shall contain one or many Navigation Events. Each NEM can contain a number of different Event Types[[1]](#footnote-1). Each Navigation Event has a number of attributes, some of which may be optional, and may have no or any number of parameters.

### NEM Event Attributes

Navigation Event attributes are common to all Navigation Events in an instance of an NEM and do not depend on the event type. There are a number of attributes, which are mandatory. The one obvious attribute required for all Navigation Events is the event time here named: “Epoch”[[2]](#footnote-2).

### NEM Event Parameters

The parameters which go with the different events strongly depend on the Event Type. For an AOS event, typical parameters is the antenna identification - here referred to as Aperture and the actual Elevation at the time. Depending on the mission or user requirements many other parameters are conceivable: Azimuth, PredictedDopplerShift, MaximumElevationInPass or PassDuration[[3]](#footnote-3)

#  FrameWork

The NEM proposes a framework to easily build new messages, reuse existing ones, facilitate messages checking …

This framework should take into account XML users, KVN users, and maybe other users.

1. To allow strong validation, in the XML representation of the NEM, the event type is implemented as the element tag or name, e.g. <PenumbraStart> … </PenumbraStart>. [↑](#footnote-ref-1)
2. Note: while conceptually “Epoch” is an attribute to each Navigation Event, in the XML notation it is implemented as a child element. This was done to allow for complex epoch definitions utilising for relative timing and different time scales. [↑](#footnote-ref-2)
3. Note: Events do not have a duration, they are instantaneous, i.e. they each exist only at one point in time. Events may however be related to something that has a duration, such as a station pass or an eclipse. Events may be used to carry information about those durations. It is however generally discouraged to make use this option. It considered preferable to define durations by means of two related events, e.g. subsequent PenumbraStart and PenumbraStop for the same occulting object. [↑](#footnote-ref-3)