IMPEx Simulation Data Model
User's guide

[Format of the XML Files describing the simulation inputs and outputs]

Title:*	Impex Data Model (XML file)
Date:*	22 February 2012
Editor:*	S. Hess
Contributors:*	M. Gangloff, R. Modolo, V. Génot, R. Jarvinen, E. Kallio, L. Häkkinen, B. Cecconi
Distribution:*	*project*
Level:*	*low level*
Roles:*	
User Requirement:	1.1.1.1
Related Workpackage(s):	WP3-WP2

Version History

Version	Date	Released by	Detail
0.0	21/02/12	S. Hess	
0.1	27/03/12	S. Hess	
0.2	12/04/12	S. Hess	
0.3	07/05/12	S. Hess	
0.4	24/05/12	S. Hess	
0.5	17/08/12	S. Hess	Not working
0.6	23/08/12	S. Hess	
0.7	04/09/12	S. Hess	
0.8		S. Hess	
0.9		S. Hess	
1.0-rc1	15/01/13	S. Hess	
1.0-rc2	01/02/13	S.Hess	

EC Project Nr.: **262863**

Version Notes

0.0 Changes

SH : Following the discussions at the last IMPEx meeting in Toulouse it appeared that :

(1) the data model should use Spase ontology, and be as much as possible Spase compliant.
(2) The data model should be easily extendable to new models.

Hence, building on the LATMOS preliminary version, we suggest a data model which includes a Spase Header, and uses Spase ontology and XML Schema.

The number of new tags is reduced for two reasons : (1) keep the data model general enough to make it extendable, and (2) the metadata are destined to be read by computers and not humans. If tags are to « make sense » to a computer, they have to be in a fairly low number.

In particular, <UPSTREAM_XXX> and <ATMOSPHERE_XXX> tags were removed, since the Spase <SimulatedRegion /> tag allows to specify to which region/object a parameter belongs to, without having to define new versions of each tags each time a new region is added. This also helps keeping the data model extendable. In short, four main tags were defined: one for particle populations, one for fields, one for chemical processes, and one general which permits to describe roughly anything. This should allow us to describe our runs in a simple way, and is limited enough so automated routines may be able to deal with it.

0.1 Changes

MG : Change in the metadata tree.
SH : Added mandatory information as suggested during discussions and defined with FMI and CDPP. Also added Recommanded information.
SH : Added a SolarUVFlux Keyword for the PropertyQuantity Enumeration.
VG : Important remark : How to describe variable grid resolution ?

0.2 Changes

SH : Added RegionParameters, which gives supplementary information for a given Region/Celestial Body.
SH: In <Repository/>, moved <Property/> inside <Extension/> to keep compatibility with Spase. Also added <SimulationProduct/>, which describes the kind of data the repository contains (3DCubes, 2DCuts, TimeSeries, SpatialSeries).
SH : Added information about DisplayData

0.3 Changes

Implementation of the decisions taken in the Graz ADD meeting.

SH : Added <InputTableURL/>, <PropertyTableURL/>, <DiagnosisTimeStep/> to describe dynamic simulations.

SH : Added discussion of <Service/> element which points toward the Webservice methods description file.

SH : Concatenate multiple XML files previously proposed in a single tree.xml file following ADD meeting decision.

SH : Cleaning in the doc and elements, removed useless tags.

SH : generated a complete description of the IMPEx+Spase Data Model, available in an companion document and online http://hess.page.latmos.ipsl.fr/impex/impex-0_3.html. Also generated the new impex-0_3.xsd schema, which has been made available online http://hess.page.latmos.ipsl.fr/impex/impex-0_3.xsd.

0.4 Changes

SH : Made a new XML example for FMI hybrid code.

RJ : Added attribute and element to <DiagnosisTimeStep/>.

SH : Changed <SavedQuantity/> type to enumeration.

SH : Added <PlaneNormalVector/> in <SpatialDescritption/> for 2DPlanes

BC : Review and corrections in the .xsd

0.5 Changes

MG : Numerous corrections on the present document (URL vs URIs)

SH : Added <LikelihoodRating> tag
Added Spectra to SimulationProducts

0.6 Changes

SH : Because of the URL vs URI problem, the link between <SimulationRun> and <Repository> was broken. In response : <Repository> removed as a way to classify data. There should be only one <Repository> per database.
Change Figure 1 accordingly, as well as section 2.1 and 2.2. Former section 3.3.1 (Sets of simulation products) removed. Lot of re-organization through the doc. Update of the examples.

0.7 Changes

SH :Add a <SimulationModel> tag and move <CodeLanguage> and <SimulationType> from <SimulationRun> to <SimulationModel>. <SimulationRun> would point toward <SimulationModel> through the <ModelID> tag. This allows to begin the tree from the simulation model and not from the simulation run. It makes particularly sense for FMI which has several codes. Also add a <ModelURL> tag which would link to a paper describing the code.

0.8 Changes

FMI : <AxisLabel> → <CoordinatesLabel>
<PlanePoint> to define a 2d plane with PlaneNormalVector
<Versions> and <ModelVersion> in <Simulation Model>
<Model> instead of <ModelID> in SimulationRun + <VersionID>
<ParticleBoundary> added in each <InputParticle> (local and optional) en kept in <SimulationDomain> (global and mandatory)
Removed <BoxSize> to avoid conflicts with <ValidMin> and <ValidMax>

Added <Mass> and <Charge> to <Particle> so that the type of Ion can be defined in <NumericalOutput>

SH : Added <Qualifier> in <Property>

0.9 Changes

<NumericalData> → <NumericalOutput> for simulated data only
<DisplayData> → <DisplayOutput> for simulated data only
<ObservedRegion> → <SimulatedRegion> for simulated data only (i.e. in NumericalOutput>, <DisplayOutput>, and all <Inputs....> and <RegionParameter>)

<Mass> → <PopulationMassNumber> in <InputParticle> and <Particle>, to match the defnition of Spase : in amu units. This avoid the Unit definition problem (see below). Add Population in front to differenciate from the keyword.
<Charge> → <PopulationChargeState> same as above (unit = proton charge).
This counts also for the <Mass> and <Charge> in <Particle>.

~~Other Elements with InputValue type (Density, temperature,....) Units and UnitsConversion passed from attribute to sub-element. The element value is now in the <Value> sub-element. This follows spase-group recommendations.~~ (Misunderstanding clarified by Esa, put back to its original form).

<Properties> and <SimulationProduct> put out of the <Extension> tag. Following spase-group recommendation.
<FlowSpeed>, <Density>, <Temperature>, added Population in front of the names.
<Mass> in <RegionParameter> → <ObjectMass>

EK :
<*Dimension> types set to integer.

SH :

EK, VG, SH : Added <ChemicalFormula> element in <Particle> and <InputPopulation>

1.0-rc1 Changes

None, except some minor correction to the xsd file itself.

1.0-rc2 Changes

Changes in SimulationModel to accomodate on-demand runs.

Index

1. Scope of this document

The aim of the present document is to describe the Data Model which is to be used to describe all simulation-related data within the IMPEx project. This project will provide simulated data to be compared to observational data through the use of several data analysis tools.

In order for these tools to retrieve any relevant information about the IMPEx synthetic data, it is necessary to add metadata - taking the form of XML files - which follow a strict and unified format, i.e. the present Data Model.

Although the data used through IMPEx are heterogeneous, it is important to keep some consistency in the manner of describing them, as for example keeping the same sense for the keywords used to describe any of these data. As the measurements to which the simulated data will be compared are described by XML files using the Spase Data Model (http://spase-group.org), it has been decided by the IMPEx Team to create a Data Model whose keywords are consistent with those of Spase.

2.What is the information to be described, and how is it searched?

2.1 Divisions and subdivisions of the data

Metadata within the IMPEx framework are related to three kinds of data. The first kind is the measured data themselves, whose description is not done by the IMPEx project but by the instrument/spacecraft teams. The two others, which are under the responsibility of the IMPEx team are:

- The description of each individual simulation run, including in particular the code inputs
- The metadata of each output dataset and of each field it contains.

In order to keep the highest level of consistency between those three kinds of metadata, and considering that the data model for measurements already in use by some of the IMPEx tools is that of the spase-group, it has been decided to develop an extension to the spase Data Model (http://spase-group.org) which allows one to format the three kinds of metadata.

The IMPEX data generated by an individual simulation are divided in datasets. These datasets come from the post-processing of the simulation results. In particular, different products are generated: 2D cuts of several quantities,

interpolation of time series along the trajectories of spacecraft. In all of these products there are different types of data: Magnetic field,...

Figure 1: How runs, data and datasets are described and linked together, within an IMPEx database.

The metadata description of the simulation results is a four level description:

_ File level (<Granule> tag - see section 3.3.2): this level gives basic information about a datafile (order within a dataset) and points toward the URL of the actual file (<Source> tag) and the URN of the dataset containing it (<ParentID> tag).

_ DataSet level (<NumericalOutput> or <DisplayOutput> tag - see section 3.3): this level gives advanced informations about a set of data files: what does it represents (magnetic field, density,...), under which form (3D Cubes, 2D Cuts, TimeSeries,...), if it relates to any spacecraft trajectory, as well as the content of the file (fields in the file). This level points the URN of the simulation run which generated the dataset (<InputResourceID> tag). For observational Data, the Spase model provides two elements (<NumericalData> or <DisplayData>) which have the same prupose, and a similar structure, than the elements dedicated to simulated data (<NumericalOutput> or <DisplayOutput>).

_ Simulation level (<SimulationRun> tag – see section 3.2): this gives information about the simulation run, and in particular the run inputs. This level is the biggest addition to the spase model. This level point toward the URN of the model description.

_ Model level (<SimulationModel> tag – see section 3.1) : this give minimal information about the simulation model (type:MHD, Hybrid,...) and is particularly used to keep track of the version used in the simulation run.

2.2 Data mining

There are two ways of searching for data:

From the simulation Inputs. It is possible to browse simulation by simulation looking for a particular set of input parameters. When an interesting set of parameters is found, it has to be possible to look at the catalog of post-processed outputs generated by this simulation, and then select a particular output. The metadata which should be made available to perform the search are in particular the plasma populations, the fields (magnitude and orientation), and the physical processes simulated. Other data have to be made available not for search, but rather for the scientific use of the results (Simulation domain, time steps, boundary conditions). This is typically how the users are expected to do the search.

By the Simulation outputs. It is possible to select a metadata describing a set of simlation outputs and find both the corresponding data and the information about the run which generated it. This is how tools should get the list of data, as it corresponds to the way the different level of metadata are linked to each other according to the IMPEx Simulation Datamodel.

To generate a user-friendly data tree (From simulation inputs) from the metadata tree in the XML file (From the simulation outputs), the typical algorithm is as follow:

```
Foreach <SimulationRun>
  add node to the root of the tree for the simulation run

  For each type of Simulation product (3dCuts,...)
   For each spacecraft (if any for this type of product)
    For each type of data (magnetic field,...)

     For each<NumericalOutput>
       if <InputResource> = SimulationRun ResourceID
     and <MeasurementType> = Correct type of data
     and <SimulationProduct> = correct type of  product
  if applies:
        and if there is a <Property> with
       <PropertyQuantity>Platform</PropertyQuantity>
       <PropertyValue> = correct name of spacecraft

        then add node to the simulation run node for the dataset
Endfors
```

This would lead to a tree with top level nodes for simulation runs, then nodes for type of simulation product, then for spacecraft if any, and then for type of data. To go to the file level, the same operation must be performed on <Granule> tags (for each granules, which get <ParentID> = NumericalOutput ResourceID and is in the good space or time range.

2.3 Minimal mandatory description

In order to permit users to retrieve data of interest, several parameters must be defined by all simulated data providers. These are minimal mandatory description, but simulation providers are encouraged to provide as much information as possible. Indeed, the simulation provider may provide more information than that useful for seach, in particular specific information which may reveal itself useful for scientific use.

2.3.1 Simulation Target

Obviously, it is important to specify which planet is simulated. To do so, a <SimulatedRegion>*Name_of_the_planet*</SimulatedRegion> element must be defined in <SimulationRun/>.

2.3.2 Incident Plasma Properties

The properties of the species in the incident plasma (Solar wind or magnetospheric plasma for satellites) must be defined in <SimulationRun/> through the use of <InputPopulation/> elements which contain an element : <SimulatedRegion> Incident</SimulatedRegion>. This can be added to another <SimulatedRegion> tag indicating the physical region (i.e. Heliosphere for solar wind...).

It means that the incident total density may be the sum of several ion densities,... Tools should be able to sum the species. If both electron and ion species are described only electrons OR ions should be declared as Incident.

2.3.3 IMF Properties

The Interplanetary Magnetic Field must be defined in <SimulationRun/> elements through the use of a <InputField/> element which must contain:

<SimulatedRegion>Heliosphere</SimulatedRegion>
<FieldQuantity>Magnetic</FieldQuantity>

The IMF must be a 3D vector (example: <InputValue>-1.6 2.5 0.0</InputValue>)

2.3.4 Planetary Ions

The planetary ions which exist in the simulation must be declared in SimulationRun/> through the use of <InputPopulation/> elements which contain a <SimulatedRegion>*Name_of_the_planet*</SimulatedRegion> element. For each of these ions, <Mass /> and <Charge /> elements must be set in order for automated tools to recognize them. Units for mass should be “amu” and for charge the proton one (“e”).

2.3.5 Simulation Box

The resolution of the simulation box must be defined in <SimulationRun><SimulationDomain> by the <GridCellSize> element if a grid resolution is defined in the simulation. Note that this element may contain a vector, in particular if the cell size is not the same along each dimension.

The dimension of the box must be defined too. For a better usage, the grid origin should be centered on the planet, and the <ValidMin/> and <ValidMax/> elements must contain the 3D vectors specifying the limit of the grid. Hence, <ValidMin/> should only have negative elements.

2.3.6 Solar UV Flux

The Solar UV Flux is an important parameter to compare simulations and observations and hence must be specified in a <Property/> element (inside an <InputParameter/> element), which contains <PropertyQuantity>SolarUVFlux </PropertyQuantity>

2.4 Recommanded information

2.4.1 Alfvén velocity & Mach number

The Spase definitions include the AlfvenVelocity and AlfvenMachNumber keyword. Hence, a <InputParameter/> element should include two <Property/> elements.

The first containing the Alfvén velocity value in the <PropertyValue/> element and:<PropertyQuantity>AlfvenVelocity</PropertyQuantity>, and the second containing the Alfvén Mach number value in the <PropertyValue/> element and:<PropertyQuantity> AlfvenMachNumber </PropertyQuantity>

2.4.2 Process Types

It is unlikely that a search on a particular Chemical Process will ever be performed automatically. However, it would be possible to perform search on whether or not a simulation included a particular type of process (i.e. electron impact ionization, photo-production,...). If the <ProcessType/> element is defined in each <InputProcess/> element, then it would be possible to automatically search if a type of process was used in the simulation. The list of allowed ProcessType values are given in Appendix II.

3.Data Model

3.1 Simulation Model

Gives basic information about the simulation code used for the simulations.

SimulationModel		
Element name	**Type**	**Describes**
ResourceID(Spase)	**String**	Unified Resource Name of the dataset
ResourceHeader(Spase)	**Element**	Generic header for Spase resources
Versions	Element	Describes the versions of the code
Caveats(Spase)	String	Any particular information.
SimulationType	**Enumeration**	Defines the type of simulation code
CodeLanguage	String	Language in which the code is written
TemporalDependence	String	Does the simulation results evolve with time ?
SimulatedRegion	String	Region targeted by the numerical code. For code accepting on-demand runs, at least one must be specified.
InputProperties	Element	Include <Property> tag defining all the input parameters(for on-demand run only).
OutputParameters	Element	Include <Parameter> tags defining the output of the code (for on-demand run).
ModelURL(Spase)	URL	URL of a reference for the model.

The <ResourceHeader> element is described hereafter in the Simulation Runs section.

For on-demand runs :

The <Property> element is fully defined later. In the InputProperty context, it defines the additional parameters accepted by the methods performing on-demand runs. The tags may be used as follow :

<Property>
  <Name>*Name*</Name> Name of parameter used in the method
  <PropertyQuantity>*Quantity*</PropertyQuantity> Kind of parameter used in the method
  <Units>*units*</Units> Units in which the parameter used in the method must be given
  <PropertyValue>*val1 val2 … valn*</PropertyValue> Define a list of allowed values of the parameter
  <ValidMin>*value*</ValidMin> Define the minimum allowed value of the parameter (same for ValidMax)
</Property>

All other tags are ignored by the tools.

3.2 Simulation Runs

Simulation runs are described inside a <SimulationRun /> element. This element purpose is to describe the type of simulation, all inputs from the simulation. It can also provide a link to the simulation outputs.

SimulationRun		
Element name	**Type**	**Describes**
ResourceID(Spase)	**String**	Unified Resource Name of the dataset
ResourceHeader(Spase)	**Element**	Generic header for Spase resources
ProviderResourceName(Spase)	String	A short textual description of a resource used by the provider which may be used to identify a resource.
ProviderProcessingLevel(Spase)	String	The provider specific classification of the processing performed on the product.
ProviderVersion(Spase)	String	Describes the release or edition of the product used by the provider.
Model	**Element**	Contains ModelID :Unified Resource Name of the model and VersionID
TemporalDependence	Yes/No	Are the outputs time dependent ?
SimulatedRegion	Enumeration	Region(s) which is simulated. See Appendix I
LikelihoodRating	Enumeration	Likelihood of the phenomenon simulated, with the actual input parameters. Either Strong, Probable, Weak, Unlikely
Caveats(Spase)	String	Any particular information.
SimulationDomain	**Element**	Definition of the spatial domain of the simulation
SimulationTime	Element	Definition of the time domain of the simulation
Input*, RegionParameters	Element	Definition of the simulation inputs (see hereafter)
Extension	Element	For non IMPEx extensions

3.2.1 Header

This element contains a header giving the most general information about the run. For each element, the type of expected entry is given. (Spase) means that the element is defined in the usual Spase Data Model, and is not an IMPEx extension. For those elements, further explanations may be obtained from the Spase group. Bold characters stand for mandatory entries.

ResourceHeader		
Element name	**Type**	**Describes**
ResourceName(Spase)	**String**	Name of this dataset
AlternateName(Spase)	String	Alternate name of this dataset
ReleaseDate(Spase)	**Date**	Date at which the element was released
ExpirationDate(Spase)	Date	
Description(Spase)	**String**	Description
Acknowledgement(Spase)	String	Acknowledgement
Contact(Spase)	**Element**	Contact
InformationURL(Spase)	Element	
Association(Spase)	String	
PriorID(Spase)	String	

Example of header, from the descritptor of an hybrid simulation performed at LATMOS, for the sake of clarity, tags are in bold font :

```
<SimulationRun>
  <ResourceID>impex://LATMOS/Hybrid/210212/SimRun</ResourceID>
  <ResourceHeader>
    <ResourceName>Hybrid_Latmos_21_02_12</ResourceName>
    <ReleaseDate>2012-02-21T00:00:00.000</ReleaseDate>
    <Description>Description of the simulation run</Description>
    <Contact>
     <PersonID>LATMOS</PersonID><Role>DataProducer</Role>
    </Contact>
  </ResourceHeader>
  <ModelID>impex://LATMOS/Hybrid</ModelID>
  <TemporalDependence>No</TemporalDependence>
  <SimulatedRegion>Mars</SimulatedRegion>
```

In this example, we describe a simulation by an Hybrid (SimulationType) simulation code of Mars (SimulatedRegion:Mars) made by the Hybrid_Latmos (ModelID) code at LATMOS (PersonID) on February 21th, 2012 (ReleaseDate).

The <ResourceHeader /> element is defined in Spase. It is used to give some description of the data to the user, but also to link the metadata describing the post-processed data (the present file) to the information about the simulation run which created it.3.2.2 Description of Run Inputs

3.2.2.1 Simulation parameters

This category includes all the parameters related to the definition of space and time in the simulation

SimulationTime

Information about the time parameters in the simulation.

Element name	Type	Describes
Description(Spase)	String	Any particular information.
Caveats(Spase)	Element	Any particular information.
Duration	xsd:duration	Duration of the simulation.
TimeStart	xsd:time	Start time
TimeStop	xsd:time	End time
TimeStep	xsd:duration	Time step
DiagnosisTimeStep	Element	Time step between diagnosis (for dynamic run purpose)

Example:

```
<SimulationTime>
  <Duration>P429S</Duration>
  <TimeStart>00:00:00</TimeStart>
  <TimeStop>00:07:09</TimeStop>
  <TimeStep>P0.165S</TimeStep>
</SimulationTime>
```

DiagnosisTimeStep		
Attribute name	**Type**	**Describes**
TimeStart	xsd:time	The time at which the time step between two diagnosis starts
Duration	**xsd:duration**	Time step between two diagnosis
Element name	**Type**	**Describes**
SavedQuantity	Enumeration	Quantity that is saved at the end of this step. Italic terms in the Appendix II list.

3.2.2.2 Particular case of dynamic simulation

In the case of dynamic simulations, inputs may evolve with time so that specifying a given value does not make sense. In that case, a time dependant table of the parameter values has to be made available. As these tables may be large, it is not wanted that they are in the database description file, but rather that they exist as independent VOTable files, which are referenced in the <InputTableURL/> and <PropertyTableURL/> elements.

SimulationDomain

Information about the spatial parameters in the simulation.

Element name	**Type**	**Describes**
CoordinateSystem(Spase)	**Element**	Definition Coordinate system.
Description(Spase)	String	Any particular information.
Caveats(Spase)	Element	Any particular information.
SpatialDimension	**String**	Number of spatial dimensions
VelocityDimension	String	Number of velocity dimensions
FieldDimension	String	Number of field dimensions
Units(Spase)	**String**	Specify the units
UnitsConversion(Spase)	String	Specify how to convert Units to SI: xxx > Y, with xxx a number and Y the SI units.
CoordinatesLabel	String list	Useful for > 1D simulation. Specifies the dimension names. Elements of a list are separated by a blank. Example: x y z. Note that these are only labels, coordinates are defined in CoordinateSystem
ValidMin(Spase)	**String list**	Minimum value along each axis. See list format above. Origin is target center.
ValidMax(Spase)	**String list**	Maximum value along each axis. See list format above. Origin is target center.
GridStructure	String	Structure of the grid.
GridCellSize	***Float list***	Size of a cell. See list format above. Recommended
Symmetry	Enumeration	Either: Axial, Plane, Central, None
BoundaryConditions	Element	See Below

The element BoundaryConditions separates conditions on the particles and fields :

Element name	**Type**	**Describes**
ParticleBoundary	Element	Boundary conditions for particles
FieldBoundary	Element	Boundary conditions for fields

Each of the *Boundary elements have the same structure:

Element name	**Type**	**Describes**
Caveats(Spase)	Element	Definition of the time domain of the simulation
FrontWall	String	Boundary conditions on the wall upstream if any, on the x=0 plane otherwise
BackWall	String	Boundary conditions on the wall downstream if any, on the x=x_max plane otherwise
SideWall	String	Boundary conditions of the other walls if any.
Obstacle	String	Boundary conditions of the obstacle(s) if any.

Example:

```
<SimulationDomain>
 <CoordinateSystem>
  <CoordinateRepresentation>Cartesian</CoordinateRepresentation>
  <CoordinateSystemName>MSO</CoordinateSystemName>
 </CoordinateSystem>
 <SpatialDimension>3D</SpatialDimension>
 <VelocityDimension>3D</VelocityDimension>
 <FieldDimension>3D</FieldDimension>
 <Units>km</Units>
 <AxesLabel>x y z</AxesLabel>
 <BoxSize>18471 34191 34191</BoxSize>
 <GridStructure>Constant</GridStructure>
 <GridCellSize>131 131 131</GridCellSize>
 <Symmetry>Axial</Symmetry>
 <BoundaryConditions>
   <ParticleBoundary>
      <FrontWall> absorbing </FrontWall>
      <BackWall> absorbing </BackWall>
     <SideWall> absorbing </SideWall>
      <Obstacle> absorbing </Obstacle>
   </ParticleBoundary>
   <FieldBoundary>
      <FrontWall> Neuman general </FrontWall>
      <BackWall> Neuman zero-gradient </BackWall>
      <SideWall> periodic </SideWall>
      <Obstacle> none </Obstacle>
   </FieldBoundary>
 </BoundaryConditions>
</SimulationDomain>
```

3.2.2.3 User-Defined Type

After the very global simulation parameters defined previously, it may be needed to describe more specific inputs to the model. There are to ways to proceed: One is to use a general template: <InputParameters /> which is defined in this section. The other is to use specific templates which are defined for particle populations, fields and chemical processes. Note that the general template is general enough so that it can also be used to describe particles, fields or chemical reaction if one was to decide not to use the specific templates.

InputParameter		
Element name	**Type**	**Describes**
Name(Spase)	**String**	Name
Set(Spase)	String	Used to define sets of parameters
ParameterKey(Spase)	**String**	A Unique identifier from cross-references
Description(Spase)	String	Any particular information.
Caveats(Spase)	String	Any particular information.
SimulatedRegion	Enumeration	Region(s) where the parameter applies.
ParameterQuantity	**Enumeration**	A keyword which defines the physical type of property. See Appendix II.
Property	**Element**	Parameter properties. At least one is mandatory.

Property

The property element is actually the main piece of the definition of the input parameters. It is the properties which can have a value, and the parameters themselves. This way a given parameter can have multiple properties, which makes more sense in some cases (if one want to define a particle, one may want to define at once its mass, charge... and not to write independent parameters). The <Property /> element contains:

Property		
Element name	**Type**	**Describes**
Name(Spase)	**String**	Name
Description(Spase)	String	Any particular information.
Caveats(Spase)	Element	Any particular information.
PropertyQuantity	Enumeration	A keyword which defines the physical type of property. See Appendix II.
Qualifier (Spase)	Enumeration	A qualifier for the PropertyQuantity
Units(Spase)	String	Specify the units
UnitsConversion(Spase)	String	Specify how to convert Units to SI: xxx > Y, with xxx a number and Y the SI units.
PropertyLabel	String list	If PropertyValue is a list, specifies labels Elements of a list are separated by a blank. Example: x y z
PropertyValue	String list	Value(s) of the property.
PropertyTableURL	URL	URL of a VOTable file containing the value(s) of the field versus time.
ValidMin(Spase)	String list	Minimum value(s).
ValidMax(Spase)	String list	Maximum value(s).
PropertyModel	String	Name or description of a model used as property.
ModelURL	URL	URL of a reference for the model.

Example:
```
<InputParameter>
 <Name>Solar UV Flux index</Name>
 <Property>
   <Name>Solar F10.7 Flux</Name>
   <PropertyQuantity>SolarUVFlux</PropertyQuantity>
   <PropertyValue>170</PropertyValue>
 </Property>
</InputParameter>
```

3.2.2.4 Pre-Defined Types

RegionParameters

Following the present Data Model, it is not necessary to define the celestial bodies in the simulation. In order to relate a parameter to a celestial body, one may just use the <SimulatedRegion/> element and refer to an object by its name, as registered in the Spase list (which may need some extension...). Even though, it may still be necessary to precise some parameters of the celestial bodies in the simulation, because they do not necessary match the actual ones, or because some of them are time varying and may need to be precised (solar sublongitude,...). To do so, one may use RegionParameters:

RegionParameters		
Element name	**Type**	**Describes**
SimulatedRegion	**Enumeration**	Region to which the parameters apply.
Description(Spase)	String	Any particular information.
Caveats(Spase)	String	Any particular information.
Radius	String + Units and UnitsConversion attribute	Radius of the body.
Sublongitude	String + Units and UnitsConversion attribute	Solar sublongitude of the celestial body. *(At simulation time 0 if rotation).*
Period	String + Units and UnitsConversion attribute	Rotational period, useful if the celestial body rotates.
Mass	String + Units and UnitsConversion attribute	Celestial Body mass, if needed.
InputTableURL	URL	URL of a VOTable file containing the value(s) of the region versus time.
Property	Element	Properties of the parameter.

Example: Defines the radius of Mars used in the simulation:

```
<RegionParameters>
      <Name>Mars</Name>
      <SimulatedRegion>Mars</SimulatedRegion>
      <Radius Units="km">3396</Radius>
      <Sublongitude Units="degrees">0</Sublongitude>
</RegionParameters>
```

InputField

The <InputField /> element is essentially a single-Property parameter. Hence, its content is close to that of a <Property /> element. It is destined to define a field set as input of the simulation run.

InputField		
Element name	**Type**	**Describes**
Name(Spase)	**String**	Name
Set(Spase)	String	Used to define sets of parameters
ParameterKey(Spase)	String	A Unique identifier from cross-references
Description(Spase)	String	Any particular information.
Caveats(Spase)	String	Any particular information.
SimulatedRegion	Enumeration	Region(s) where the parameter applies. See Appendix I
CoordinateSystem(Spase)	Element	Definition Coordinate system. May be useful is not the same than that of the global simulation (e.g. spherical model, in a cartesian simulation).
FieldQuantity(Spase)	Enumeration	A Spase defined keyword which defines the physical type of property. See Appendix II.
Units(Spase)	String	Specify the units
UnitsConversion(Spase)	String	Specify how to convert Units to SI: xxx > Y, with xxx a number and Y the SI units.
InputLabel	String list	If the field is a vector field, specifies component labels. Elements of a list are separated by a blank.
FieldValue	String list	Value(s) of the field.
InputTableURL	URL	URL of a VOTable file containing the value(s) of the field versus time.
ValidMin(Spase)	String list	Minimum value(s).
ValidMax(Spase)	String list	Maximum value(s).
FieldModel	String	Name or description of a field model.
ModelURL	URL	URL of a reference for the model.

Example:

```
<InputField>
    <Name>IMF</Name>
    <Description>Interplanetary Magnetic Field</Description>
    <SimulatedRegion>Heliosphere</SimulatedRegion>
    <FieldQuantity>Magnetic</FieldQuantity>
    <Units>nT</Units>
    <InputLabel>Bx By Bz</InputLabel>
    <FieldValue>-1.634 2.516 0.0</FieldValue>
</InputField>
```

InputPopulation

Contrary to the previous pre-defined input types, <InputPopulation /> is a multi-properties type, i.e. many different properties have to be defined to describe a plasma population. This introduces a slightly different syntax for this element when compared to the others. This is particularly the case for the Units definitions, which is not an element any more but the attribute of an element (see example).

InputPopulation		
Element name	**Type**	**Describes**
Name(Spase)	**String**	Name
Set(Spase)	String	Used to define sets of parameters
ParameterKey(Spase)	String	A Unique identifier from cross-references
Description(Spase)	String	Any particular information.
Caveats(Spase)	String	Any particular information.
SimulatedRegion	Enumeration	Region(s) where the parameter applies. See Appendix I
ChemicalFormula	String	Chemical formula of the specie
AtomicNumber	String	
PopulationMassNumber	String + Units and UnitsConversion attribute	Specify the population mass.
PopulationChargeState	String + Units and UnitsConversion attribute	Specify the population charge.
PopulationDensity	String + Units and UnitsConversion attribute	Specify the population density.
PopulationTemperature	String + Units and UnitsConversion attribute	Specify the population temperature.
PopulationFlowSpeed	String + Units and UnitsConversion attribute	Specify the population bulk velocity.
Distribution	String	Specify the population distribution.
ProductionRate	String + Units and UnitsConversion attribute	Specify the production rate
TotalProductionRate	String + Units and UnitsConversion attribute	Specify the production rate integrated over the simulation domain.
InputTableURL	URL	URL of a VOTable file containing the value(s) of the population versus time.
Profile	String	Defines the population profile.
ModelURL	URL	URL of a reference for the profile.
ParticleBoundary	Element	Overrides the global particle boundary definition in SimulayionDomain for this particular species

Example :

The following presents the description of three populations. The first one are solar wind protons flowing through the simulation domain. The second one are solar wind electrons which are massless (hybrid simulation). Both are related to the Heliosphere region (Solar wind), which permits one to separate the upstream species from the ones tied to the planet. The third population are neutrals tied to the planet (Mars) and defined with a profile.

```
<InputPopulation>
  <Name>Solar Wind Protons</Name>
  <Description>Solar wind Protons</Description>
  <SimulatedRegion>Heliosphere</SimulatedRegion>
  <PopulatioMassNumber>1</PopulatioMassNumber>
  <PopulationChargeState>1</PopulationChargeState>
  <PopulationDensity="cm-3">2.8</PopulationDensity>
  <PopulationTemperature Units="K">5e4</PopulationTemperature>
  <PopulationFlowSpeed Units="km/s">400</PopulationFlowSpeed>
  <Distribution>Drifting Maxwellian</Distribution>
</InputPopulation>
<InputPopulation>
  <Name>Solar Wind electrons</Name>
  <Description>Solar wind electron fluid</Description>
  <SimulatedRegion>Incident</SimulatedRegion>
  <SimulatedRegion>Heliosphere</SimulatedRegion>
  <PopulationMassNumber>0</PopulationMassNumber>
  <PopulationChargeState>-1</PopulationChargeState>
  <PopulationDensity Units="cm-3">3</PopulationDensity>
  <PopulationTemperature Units="eV">20</PopulationTemperature>
  <PopulationFlowSpeed Units="km/s">400</PopulationFlowSpeed>
</InputPopulation>
<InputPopulation>
  <Name>Atmospheric Hydrogen</Name>
  <Description>Extended neutral Oxygen atmosphere</Description>
  <SimulatedRegion>Mars</SimulatedRegion>
  <PopulationMassNumeber>1</PopulationMassNumber>
  <PopulationCharge Units="e">0</PopulationChargeState>
  <Profile>Krasnopolsky et al, 1993</Profile>
  <ModelURL>http://dx.doi.org/10.1006/icar.1993.1003</ModelURL>
</InputPopulation>
```

InputProcess

The <InputProcess /> element is destined to define a chemical process set as input of the simulation run.

InputProcess		
Element name	**Type**	**Describes**
Name(Spase)	**String**	Name
Set(Spase)	String	Used to define sets of parameters
ParameterKey(Spase)	String	A Unique identifier from cross-references
Description(Spase)	String	Any particular information.
Caveats(Spase)	String	Any particular information.
SimulatedRegion	Enumeration	Region(s) where the parameter applies. See Appendix I
ProcessType	**Enumeration**	Type of Process to be chosen in a list. See Appendix II.
Units(Spase)	String	Specify the units
UnitsConversion(Spase)	String	Specify how to convert Units to SI: xxx > Y, with xxx a number and Y the SI units.
ProcessCoefficient	String	Coefficient of the reaction
ProcessCoeffType	Enumeration	Type of coefficient. Either: frequency, rate, CrossSection or Other
ProcessModel	String	Name of a model used
ModelURL	URL	URL of a reference for the reaction coefficient.

Example:

```
<InputProcess>
   <Name>H+ + H –> H+ + H</Name>
   <ProcessType>ChargeExchange</ProcessType>
   <Units>cm-2</Units>
   <ProcessCoefficient>2.5e-15</ProcessCoefficient>
   <ProcessCoeffType>CrossSection</ProcessCoeffType>
</InputProcess>
<InputProcess>
   <Name>H + e- -> H+ + 2e-</Name>
   <ProcessType>ElectronImpact</ProcessType>
   <Units>s-1</Units>
   <ProcessCoefficient>7.6e-9</ProcessCoefficient>
   <ProcessCoeffType>Frequency</ProcessCoeffType>
</InputProcess>
```

3.3 Description of Run Outputs (Datasets and Data)

By sets of simulated data, we mean a set of data of exactly the same type, such as a time series of interpolated magnetic field along a given spacecraft orbit, all with the same simulation and post-processing inputs. There can be several data files in each datasets corresponding for example to sub time series. To make the distinction, there are two highest level elements in the XML file describing the Individual DataSets. <NumericalOutput />, which describe the whole DataSet, and <Granule/> which describes an individual file. These two elements are taken from Spase, and are only slightly modified for IMPEx.

3.3.1 NumericalOutput

Each set of data is described by a <NumericalOutput /> element.

NumericalOutput		
Element name	**Type**	**Describes**
ResourceID(Spase)	**String**	Name
ResourceHeader(Spase)	**String**	A Unique identifier from cross-references.
AccessInformation(Spase)	**String**	Information on how accessing the data
ProcessingLevel(Spase)	String	Level of Processing
ProviderRessourceName(Spase)	String	Who did provide the data
ProviderProcessingLevel(Spase)	String	Level of Processing
ProviderVersion(Spase)	String	Version
SimulatedInstrumentID(Spase)	String	ID of the instrument that has been simulated if any.
MeasurementType(Spase)	**Enumeration**	Type of measure
TemporalDescription(Spase) or SpatialDescription	**Element**	Specify the temporal or spatial span of the data. (/!\ Spase compliant only for time series /!\ )
SpectralRange(Spase)	Enumeration	
SimulatedRegion	Enumeration	Region(s) which is simulated (may be used more than once if there are several regions in the simulation). See Appendix I
Caveats(Spase)	String	Any particular information.
Keyword(Spase)	String	Any particular information.
InputResourceID(Spase)	**String**	ID of the SimulationRun
Parameters(Spase)	Element	Describe the parameters in the data
SimulationProduct	**Enumeration**	Type of product. See Appendix II
Property	Element	Property specific to this data(set)
Extension(Spase)	**Element**	Specify inside any property related to the data.

The IMPEx add-on to this element resumes to the <SpatialDescription /> element.

SpatialDescription		
Element name	**Type**	**Describes**
Dimension	**String**	Number of Dimensions of the spatial domain
CoordinateSystem(Spase)	**Element**	Coordinate System
Units(Spase)	String	Units
UnitsConversion(Spase)	String	Conversion factor to SI Units
CoordinatesLabel	String	Name of each dimension
RegionBegin	Float list	Start position of the series, 2D cut or 3D box
RegionEnd	Float list	Stop position of the series, 2D cut or 3D box
Step	Float list	Spatial step between two values in the series
PlaneNormalVector	Float list	For 2D Cuts, normal vector to the plane.
PlanePoint	Float list	Coordinates of a point of the 2D cut.

<Extension/> may contain <Property/> elements or <SimulationProduct/> element. This latest element describes which kind of data the repository contains: *3DCubes, 2DCuts, TimeSeries or SpatialSeries*.

3.3.2 Granule

The <Granule /> element is a simple element which declares a file as a part of a series previously described in a <NumericalOutput /> element (reference through <ParentID />).

Granule		
Element name	**Type**	**Describes**
ResourceID(Spase)	**String**	A Unique identifier for cross-references
ReleaseDate(Spase)	**String**	Date at which the data has been released
ExpirationDate(Spase)	String	Date at which the data expires
ParentID(Spase)	**String**	ID of the parent NumericalOutput
PriorID(Spase)	String	
StartDate(Spase) or RegionBegin	**Date** or **String list**	Start Date of the time serie or Start location of the spatial domain, same structure that Start in <NumericalOutput/>. (/!\ Spase compliant only for time series /!\ )
StopDate(Spase) or RegionEnd	**Date** or **String list**	Stop Date of the time serie or Stop location of the spatial domain, with the same structure that Start in <NumericalOutput/>
Source(Spase)	**Element**	See below

The Source element points toward the actual data file:

Source		
Element name	**Type**	**Describes**
SourceType(Spase)	**Enumeration**	A Unique identifier for cross-references
URL(Spase)	**String**	URL at which the Data can be reached
MirrorURL(Spase)	String	Mirror Uniform Ressource Locator
Checksum(Spase)	Element	ID of the parent NumericalOutput
DataExtent(Spase)	Element	Define the size of the file (see spase doc.)

3.3.3 DisplayOutput

If the data are not numerical sets but rather a graphical representation (like a pdf), one must use DisplayOutput instead of NumericalOutput. As for NumericalOutput, the only IMPEx add-on to the spase definition is the possibility to have <SpatialDescription/> instead of <TemporalDescription>.

DisplayOutput		
Element name	**Type**	**Describes**
ResourceID(Spase)	**String**	Name
ResourceHeader(Spase)	**String**	A Unique identifier from cross-references.
AccessInformation(Spase)	**String**	Information on how accessing the data
ProcessingLevel(Spase)	String	Level of Processing
ProviderRessourceName(Spase)	String	Who did provide the data
ProviderProcessingLevel(Spase)	String	Level of Processing
ProviderVersion(Spase)	String	Version
SimulatedInstrumentID(Spase)	String	ID of the instrument (may not be relevant)
MeasurementType(Spase)	**Enumeration**	Type of measure
TemporalDescription(Spase) or SpatialDescription	Element	Specify the temporal or spatial span of the data. (/!\ Spase compliant only for time series /!\ )
SpectralRange(Spase)	Enumeration	
SimulatedRegion	Enumeration	Region(s) which is simulated . See Appendix I
Caveats(Spase)	String	Any particular information.
Keyword(Spase)	String	Any particular information.
InputResourceID(Spase)	**String**	ID of the SimulationRun
Parameters(Spase)	Element	Describe the parameters in the data
SimulationProduct	**Enumeration**	Type of product. See Appendix II
Property	Element	Property specific to this data(set)
Extension(Spase)	Element	Specify inside any data related property.

APPENDIX I : SimulatedRegion List

List of all possibilities for the <SimulatedRegion /> element.

Asteroid
Comet
Earth
Earth.Magnetosheath
Earth.Magnetosphere
Earth.Magnetosphere.Magnetotail
Earth.Magnetosphere.Main
Earth.Magnetosphere.Polar
Earth.Magnetosphere.Radiation Belt
Earth.Near Surface
Earth.Near Surface.Atmosphere
Earth.Near Surface.Auroral Region
Earth.Near Surface.Equatorial Region
Earth.Near Surface.Ionosphere
Earth.Near Surface.Ionosphere.D-Region
Earth.Near Surface.Ionosphere.E-Region
Earth.Near Surface.Ionosphere.F-Region
Earth.Near Surface.Ionosphere.Topside
Earth.Near Surface.Mesosphere
Earth.Near Surface.Plasmasphere
Earth.Near Surface.Polar Cap
Earth.Near Surface.SouthAtlanticAnomalyRegion
Earth.Near Surface.Stratosphere
Earth.Near Surface.Thermosphere
Earth.Near Surface.Troposphere
Earth.Surface
Heliosphere
Heliosphere.Heliosheath
Heliosphere.Inner
Heliosphere.Near Earth
Heliosphere.Outer
Heliosphere.Remote 1AU
Interstellar
Jupiter
Mars
Mercury
Neptune
Pluto
Saturn
Sun
Sun.Chromosphere
Sun.Corona
Sun.Interior
Sun.Photosphere
Sun.Transition Region
Uranus
Venus

APPENDIX II : PropertyQuantity List

List of all possibilities for PropertyQuantity. This list is the fusion of several lists defined by the Spase DataModel, or by the IMPEx one. Italic terms are those usable as SavedQuantity.

From DirectionAngle

Azimuth Angle
Elevation Angle
Polar Angle

From FieldQuantity

Current
Electric
Electromagnetic
Gyrofrequency
Magnetic
Plasma Frequency
Potential
Poynting Flux

From InstrumentType

Antenna
Channeltron
Coronograph
Double Sphere
Dust Detector
Electron Drift Instrument
Electrostatic Analyser
Energetic Particle Instrument

Faraday Cup
Flux Feedback
Fourier Transform Spectrograph
Geiger-Mueller Tube
Imager
Imaging Spectrometer
Interferometer
Ion Chamber
Ion Drift
Langmuir Probe
Long Wire
Magnetometer
Mass Spectrometer
Microchannel Plate
Multispectral Imager
Neutral Atom Imager
Neutral Particle Detector
Particle Correlator
Particle Detector
Photometer
Photopolarimeter
Platform
Proportional Counter
Quadrispherical Analyser
Radar
Radiometer
Resonance Sounder
Retarding Potential Analyser
Riometer
Scintillation Detector
Search Coil
Sounder
Spacecraft Potential Control
Spectral Power Receiver
Spectrometer
Time Of Flight
Unspecified
Waveform Receiver

From MeasurementType

Activity Index
Dopplergram
Dust
Electric Field
Energetic Particles
Ephemeris
Image Intensity
Instrument Status
Ion Composition
Irradiance
Magnetic Field
Magnetogram
Neutral Atom Images
Neutral Gas
Profile
Radiance
Spectrum
Thermal Plasma
Waves
Waves.Active
Waves.Passive

From MixedQuantity

Akasofu Epsilon
Alfven Mach Number
Alfven Velocity
Frequency-To-Gyrofrequency Ratio
Magnetosonic Mach Number
Other
Plasma Beta
Total Pressure
V Cross B

From ParticleQuantity

Arrival Direction
Atomic Number Detected
Average Charge State
Charge State
Count Rate
Counts
Energy
Energy Density
Energy Flux
Flow Speed
Flow Velocity
Fluence
Gyrofrequency
Heat Flux
Mass
Mass Density
Mass Number
Number Density
Number Flux

Phase-Space Density
Plasma Frequency
Pressure
Sonic Mach Number
Sound Speed
Temperature
Thermal Speed
Velocity

From ParticleType

Aerosol
Alpha Particle
Atom
Dust
Electron
Ion
Molecule
Neutron
Proton

From SpectralRange

Ca-K
Extreme Ultraviolet
Far Ultraviolet
Gamma Rays
H-alpha
Hard X-rays
He-10830
He-304
Infrared
K-7699
LBH Band
Microwave
Na-D
Ni-6768
Optical
Radio Frequency
Soft X-Rays
Ultraviolet
White-Light
X-Rays

From WaveQuantity

AC-Electric Field
AC-Magnetic Field
Absorption
Doppler Frequency
Emissivity
Energy Flux
Equivalent Width
Frequency
Gyrofrequency
Intensity
Line Depth
Magnetic Field
Mode Amplitude
Plasma Frequency
Polarization
Poynting Flux
Propagation Time
Stoke's Parameters
Velocity
Wavelength

From WaveType

Electromagnetic
Electrostatic
Hydrodynamic
MHD
Photon
Plasma Waves

From ProcessType

ChargeExchange
DissociativeRecombination
ElectronImpact
PhotoIonization

From ProcessCoeffType

Frequency
CrossSection
Rate
Other

From ImpexQuantity

IMFClockAngle
SolarUVFlux

From SimulationProduct

3DCubes
2DCuts
SpatialSeries
TimeSeries
Spectra
Lines