Acceptor and Schema documentation

For correct dataflow configuration in VisNow engine it is crucial to properly configure module ports. Output ports should be aware of what data schema they can produce, while input ports should be aware what data schema it can accept for proper module functioning. Data schema is a formal description of a field containing its internal structure e.g. data components, grid type, grid size, etc.

Data schemas
Data schemas are set for output ports and define what field this port can produce. You can define several schemas for a single output port if it can produce different output fields.
Use <schema> and </schema> tags inside <output></output> section to define a schema.

Data acceptors
Data acceptors are set for input ports and define what field this port can accept. You can define several acceptors for a single input port if it can accept different fields.
Use <acceptor> and </acceptor> tags inside <input></input> section to define an acceptor.

Connection
While creating a connection between modules or while dragging a connection, module ports are tested for data schema compatibility.

1) Output port with data vs. input port
If an output port has already some data, the field itself is tested against input port acceptors list. If this field fits any acceptor, the connection is possible (green port glow). If this field doesn't fit any of the acceptors the connection is impossible. If there are no acceptors defined, only data type is tested.

2) Output port without data vs. input port.
If an output port is empty all of its data schemas are tested against input port acceptors list. If any of its schema fits any acceptor, the connection is conditionally possible (yellow port glow). If none of schema fit any of the acceptors the connection is impossible. If there are no schema or acceptors defined, only data type is tested.

Parameters
For both schemas and acceptors you define data parameters by using <param/> tag in the following syntax:
<param name="PARAM_NAME" value="PARAM_VALUE"/>
See list below for possible PARAM_NAME types and PARAM_VALUE values.
If the boolean parameter has its opposite parameter (see table) it means that setting one of them to TRUE automatically sets the other one to FALSE.

Parameter name	Variable type	Value	Opposite to	Description	Example
FIELD	boolean	true/false	-	Data is field.	<param name="FIELD" value="true"/>
TIME	boolean	true/false	-	Field has multiple time frames	<param name="TIME" value="true"/>
NSPACE	int	1/2/3	-	Number of space dimensions of field geometry	<param name="NSPACE" value="3"/>
NDATA	int	natural number	-	Exact number of data components in a field	<param name="NDATA" value="3"/>
DATA_VECLEN	int	natural number	-	Field containing at least one data component with an equal veclen (1 for scalar component)	<param name="DATA_VECLEN" value="1"/>
DATA_VECLENS	int[]	comma separated array of natural numbers	-	Field with exact number and order of data components with given veclens	<param name="DATA_VECLENS" value="1,1,3"/>
DATA_NAME	String	non empty string	-	Field containing at least one data component with an equal name	<param name="DATA_NAME" value="image"/>

DATA_NAMES	String[]	comma separated array of non empty strings	-	Field with exact number and order of data components with given names	<param name="DATA_NAMES" value="red,green,blue"/>
DATA_TYPE	String	BooleanDataArray, ByteDataArray, ShortDataArray, IntDataArray, LongDataArray, FloatDataArray, DoubleDataArray, ComplexDataArray, LogicDataArray, StringDataArray, ObjectDataArray, SimpleNumeric(macro)	-	Field containing at least one data component with an equal data type	<param name="DATA_TYPE" value="ByteDataArray"/>
DATA_TYPES	String[]	comma separated array of above types, excluding SimpleNumeric macro	-	Field with exact number and order of data components with given data types	<param name="DATA_TYPES" value="ByteDataArray,ByteDataArray,FloatDataArray"/>
REGULAR	boolean	true/false	IRREGULAR	Field has regular a structure	<param name="REGULAR" value="true"/>
NDIMS	int	1,2,3	-	Number of dimensions of a regular field	<param name="NDIMS" value="3"/>
DIMS	int[]	comma separated natural numbers	-	Exact dimensions of a regular field	<param name="DIMS" value="512,512,100"/>
AFFINE	boolean	true/false	COORD	Regular field with grid coordinates defined implicitly by base vectors and origin	<param name="AFFINE" value="true"/>
COORD	boolean	true/false	AFFINE	Regular field with explicitly defined grid coordinates	<param name="COORD" value="true"/>
IRREGULAR	boolean	true/false	REGULAR	Field has irregular structure	<param name="IRREGULAR" value="true"/>
CELLSETS	boolean	true/false		Irregular field with at least one cell set	<param name="CELLSETS" value="true"/>
NCELLSETS	int	natural number	-	Exact number of cell sets in the irregular field	<param name="NCELLSETS" value="2"/>
CELLSET_NAME	String	non empty string	-	Irregular field containing at least one cell set with an equal name	<param name="CELLSET_NAME" value="engine"/>
CELLSET_NAMES	String[]	comma separated array of non empty strings	-	Irregular field with exact number and order of cell sets with given names	<param name="CELLSET_NAMES" value="body,wings,engine"/>
NCELLDATA	int	natural number	-	Irregular field containing at least one cell set with exact number of cell data components	<param name="NCELLDATA" value="3"/>
CELLDATA_VECLE	*int*	*natural number*	-	*Irregular field*	*<param*

N				containing at least one cell set with cell data component with an equal veclen (1 for scalar component)	name="CELLDATA_VECLEN" value="1"/>
CELLDATA_VECLENS	int[]	comma separated array of natural numbers	-	Irregular field containing at least one cell set with exact number and order of cell data components with given veclens	<param name="CELLDATA_VECLENS" value="1,1,3"/>
CELLDATA_NAME	String	non empty string	-	Irregular field containing at least one cell set with cell data component with an equal name	<param name="CELLDATA_NAME" value="image"/>
CELLDATA_NAMES	String[]	comma separated array of non empty strings	-	Irregular field containing at least one cell set with exact number and order of cell data components with given names	<param name="CELLDATA_NAMES" value="red,green,blue"/>
CELLDATA_TYPE	String	BooleanDataArray, ByteDataArray, ShortDataArray, IntDataArray, LongDataArray, FloatDataArray, DoubleDataArray, ComplexDataArray, LogicDataArray, StringDataArray, ObjectDataArray, SimpleNumeric(macro)	-	Irregular field containing at least one cell set with one cell data component with an equal data type	<param name="CELLDATA_TYPE" value="ByteDataArray"/>
CELLDATA_TYPES	String[]	comma separated array of above types, excluding SimpleNumeric macro	-	Irregular field containing at least one cell set with exact number and order of cell data components with given data types	<param name="CELLDATA_TYPES" value="ByteDataArray,ByteDataArray,FloatDataArray"/>
CELLS_2D	boolean	true/false	-	Irregular field with at least one cell set or boundary set containing 2D cells (triangle or quad)	<param name="CELLS_2D" value="true"/>
CELLS_3D	boolean	true/false	-	Irregular field with at least one cell set or boundary set containing 3D cells (tetra, pyramid, prism or hexahedron)	<param name="CELLS_3D" value="true"/>
CELLS_POINT	boolean	true/false	-	Irregular field with at least one cell set containing POINT cells	<param name="CELLS_POINT" value="true"/>
CELLS_SEGMENT	boolean	true/false	-	Irregular field with at least one cell set containing SEGMENT cells	<param name="CELLS_SEGMENT" value="true"/>
CELLS_TRIANGLE	boolean	true/false	-	Irregular field with	*<param*

				at least one cell set containing TRIANGLE cells	name="CELLS_TRIANGLE" value="true"/>
CELLS_QUAD	boolean	true/false	-	Irregular field with at least one cell set containing QUAD cells	<param name="CELLS_QUAD" value="true"/>
CELLS_TETRA	boolean	true/false	-	Irregular field with at least one cell set containing TETRA cells	<param name="CELLS_TETRA" value="true"/>
CELLS_PYRAMID	boolean	true/false	-	Irregular field with at least one cell set containing PYRAMID cells	<param name="CELLS_PYRAMID" value="true"/>
CELLS_PRISM	boolean	true/false	-	Irregular field with at least one cell set containing PRISM cells	<param name="CELLS_PRISM" value="true"/>
CELLS_HEXAHEDRON	boolean	true/false	-	Irregular field with at least one cell set containing HEXAHEDRON cells	<param name="CELLS_HEXAHEDRON" value="true"/>

Parameter hierarchy

You don't have to always define all the parameters implicitly due to succesion. If you define any parameter and its presence requires some ancestors, those ancestors are also automatically set.
E.g. if you set *<param name="NDIMS" value="3"/>* it automatically sets *<param name="REGULAR" value="true"/>* and *<param name="FIELD" value="true"/>*, because data structure with number of dimensions equal 3 means it is a 3D regular field.
The following list shows all available parameter names and their hierarchy:

- FIELD
  - TIME
  - NSPACE
  - NDATA
    - DATA_VECLENS
    - DATA_NAMES
    - DATA_TYPES
  - DATA_VECLEN
  - DATA_NAME
  - DATA_TYPE
  - REGULAR
    - NDIMS
      - DIMS
    - AFFINE
    - COORD
  - IRREGULAR
    - CELLSETS
      - NCELLSETS
        - CELLSET_NAMES
      - CELLSET_NAME
      - NCELLDATA
        - CELLDATA_VECLENS
        - CELLDATA_NAMES
        - CELLDATA_TYPES
      - CELLDATA_VECLEN
      - CELLDATA_NAME
      - CELLDATA_TYPE
      - CELLS_2D
      - CELLS_3D
      - CELLS_POINT
      - CELLS_SEGMENT
      - CELLS_TRIANGLE
      - CELLS_QUAD
      - CELLS_TETRA

- CELLS_PYRAMID
- CELLS_PRISM
- CELLS_HEXAHEDRON

Example

Below you can find an example module.xml with acceptors and schemas defined for input ports and output ports.