Schema documentation for component_schema.xsd

august 31, 2016

Table of Contents

Namespace: ""

Schema(s)

Main schema `component_schema.xsd`

Namespace	No namespace
Properties	attribute form default: `unqualified` element form default: `qualified`

Included schema `common_elements.xsd`

Namespace	No namespace
Properties	attribute form default: `unqualified` element form default: `qualified`

Included schema `common_types.xsd`

Namespace	No namespace
Properties	attribute form default: `unqualified` element form default: `qualified`

Included schema `channel_schema.xsd`

Namespace	No namespace
Properties	attribute form default: `unqualified` element form default: `qualified`

Included schema `command_schema.xsd`

Namespace	No namespace
Properties	attribute form default: `unqualified` element form default: `qualified`

Included schema `event_schema.xsd`

Namespace	No namespace
Properties	attribute form default: `unqualified` element form default: `qualified`

Included schema `internal_interface_schema.xsd`

Namespace	No namespace
Properties	attribute form default: `unqualified`

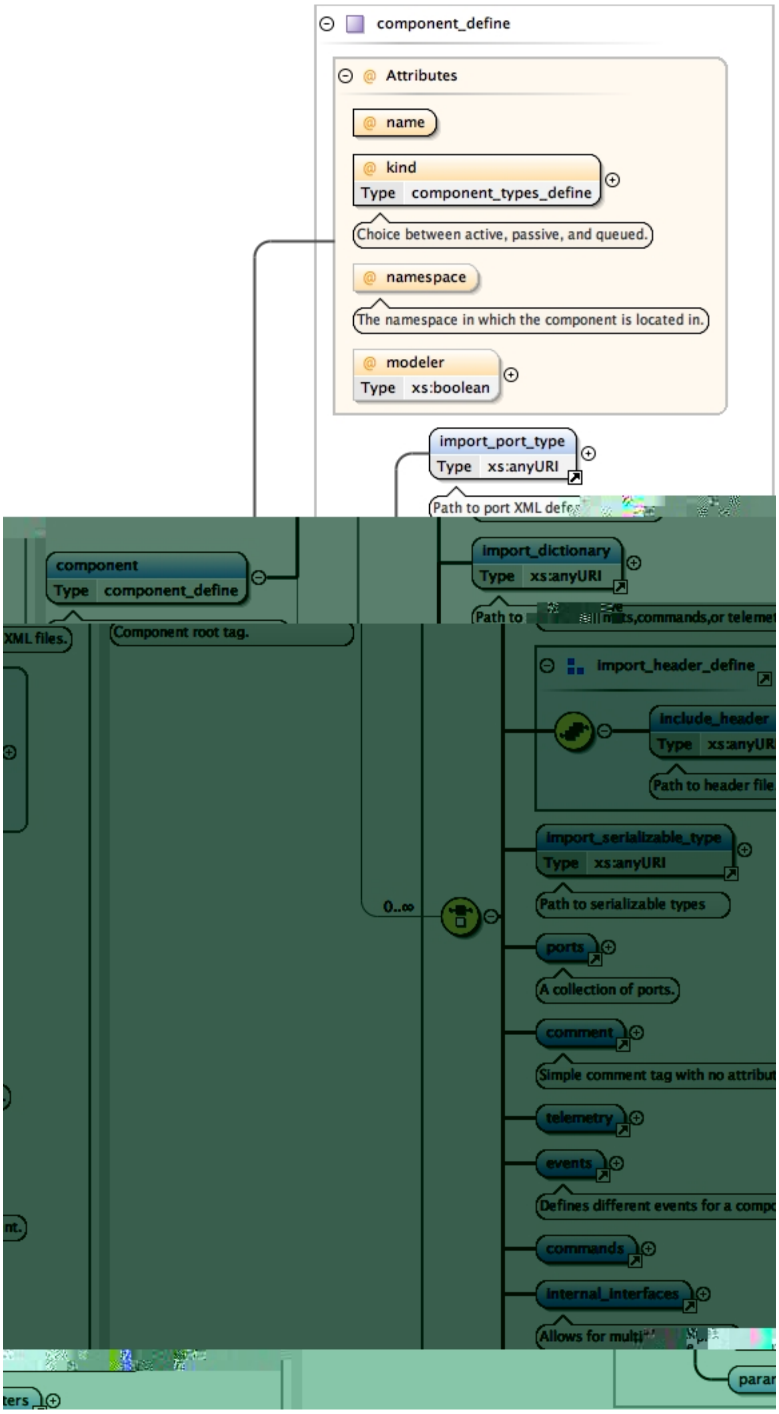
	element form default:	`qualified`

Included schema `parameters_schema.xsd`

Namespace	No namespace	
Properties	attribute form default:	`unqualified`
	element form default:	`qualified`

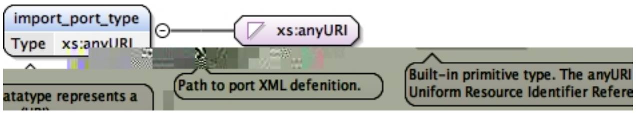
Element(s)

Element `component`

Namespace	No namespace
Annotations	`Component root tag.`
Diagram	 component_define Attributes name kind Type component_types_define Choice between active, passive, and queued. namespace The namespace in which the component is located in. modeler Type xs:boolean import_port_type Type xs:anyURI Path to port XML defi component Type component_define Component root tag. XML files. import_dictionary Type xs:anyURI Path to nts,commands,or telemet import_header_define include_header Type xs:anyUR Path to header file import_serializable_type Type xs:anyURI Path to serializable types 0..∞ ports A collection of ports. comment Simple comment tag with no attribut telemetry events Defines different events for a comp commands internal_interfaces Allows for multi nt. ters parar

<table>
<tr><td>Type</td><td>component_define</td></tr>
<tr><td>Properties</td><td>content: complex</td></tr>
<tr><td>Model</td><td>import_port_type | import_dictionary | (include_header) | import_serializable_type | ports | comment | telemetry | events | commands | internal_interfaces | parameters</td></tr>
<tr><td>Children</td><td>commands, comment, events, import_dictionary, import_port_type, import_serializable_type, include_header, internal_interfaces, parameters, ports, telemetry</td></tr>
<tr><td>Instance</td><td><component kind="" modeler="" name="" namespace="">
  <import_port_type>{1,1}</import_port_type>
  <import_dictionary>{1,1}</import_dictionary>
  <include_header>{1,1}</include_header>
  <import_serializable_type>{1,1}</import_serializable_type>
  <ports>{1,1}</ports>
  <comment>{1,1}</comment>
  <telemetry telemetry_base="">{1,1}</telemetry>
  <events event_base="">{1,1}</events>
  <commands opcode_base="">{1,1}</commands>
  <internal_interfaces>{1,1}</internal_interfaces>
  <parameters opcode_base="" parameter_base="">{1,1}</parameters>
</component></td></tr>
<tr><td>Attributes</td><td><table>
<tr><th>QName</th><th>Type</th><th>Use</th><th></th></tr>
<tr><td><b>kind</b></td><td>component_types_define</td><td>required</td><td></td></tr>
<tr><td></td><td colspan="3">Choice between active, passive, and queued.</td></tr>
<tr><td><b>modeler</b></td><td>xs:boolean</td><td>optional</td><td></td></tr>
<tr><td><b>name</b></td><td></td><td>required</td><td></td></tr>
<tr><td><b>namespace</b></td><td></td><td>optional</td><td></td></tr>
<tr><td></td><td colspan="3">The namespace in which the component is located in.</td></tr>
</table></td></tr>
<tr><td>Source</td><td><xs:element name="component" type="component_define">
  <xs:annotation>
    <xs:documentation>Component root tag.</xs:documentation>
  </xs:annotation>
</xs:element></td></tr>
</table>

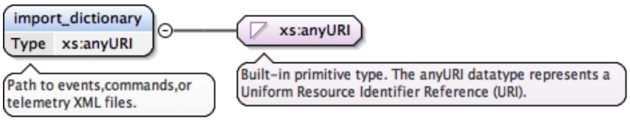
Element `import_port_type`

Namespace	No namespace
Annotations	Path to port XML defenition.
Diagram	
Type	xs:anyURI
Properties	content: simple
Used by	Complex Type component_define

Source	<xs:element name="import_port_type" type="xs:anyURI">
	  <xs:annotation>
	    <xs:documentation>Path to port XML defenition.</xs:documentation>
	  </xs:annotation>
	</xs:element>

import_port_type
Type xs:anyURI
xs:anyURI
atatype represents a
Path to port XML defenition.
Built-in primitive type. The anyURI
Uniform Resource Identifier Refere

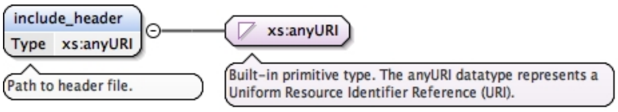
Element `import_dictionary`

Namespace	No namespace
Annotations	Path to events,commands,or telemetry XML files.
Diagram	
Type	xs:anyURI
Properties	content: simple
Used by	Complex Type component_define

import_dictionary
Type xs:anyURI
xs:anyURI
Path to events,commands,or telemetry XML files.
Built-in primitive type. The anyURI datatype represents a Uniform Resource Identifier Reference (URI).

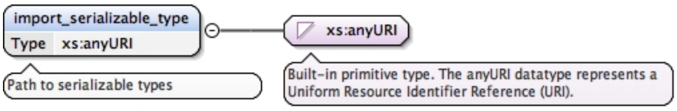
Source	`<xs:element name="import_dictionary" type="xs:anyURI">`
	`  <xs:annotation>`
	`    <xs:documentation>Path to events,commands,or telemetry XML files.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:element>`

Element import_header_define / include_header

Namespace	No namespace
Annotations	Path to header file.
Diagram	 include_header Type xs:anyURI xs:anyURI Path to header file. Built-in primitive type. The anyURI datatype represents a Uniform Resource Identifier Reference (URI).
Type	xs:anyURI
Properties	content: simple

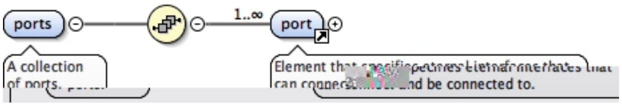
Source	`<xs:element name="include_header" type="xs:anyURI">`
	`  <xs:annotation>`
	`    <xs:documentation>Path to header file.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:element>`

Element import_serializable_type

Namespace	No namespace
Annotations	Path to serializable types
Diagram	 import_serializable_type Type xs:anyURI xs:anyURI Path to serializable types Built-in primitive type. The anyURI datatype represents a Uniform Resource Identifier Reference (URI).
Type	xs:anyURI
Properties	content: simple
Used by	Complex Type component_define

Source	`<xs:element name="import_serializable_type" type="xs:anyURI">`
	`  <xs:annotation>`
	`    <xs:documentation>Path to serializable types</xs:documentation>`
	`  </xs:annotation>`
	`</xs:element>`

Element ports

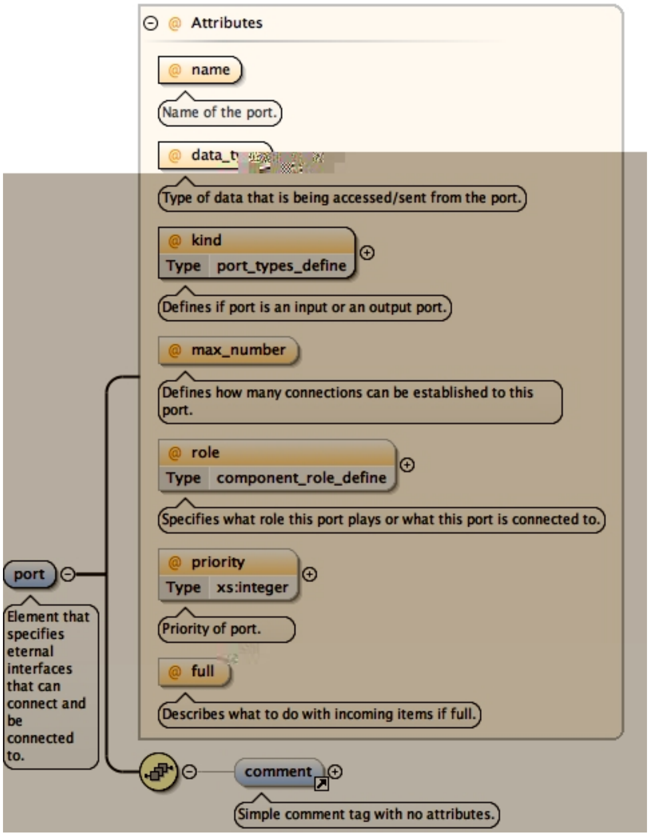
Namespace	No namespace
Annotations	A collection of ports.
Diagram	 ports 1..∞ port A collection of ports. and be connected to.
Properties	content: complex
Used by	Complex Type component_define
Model	port+
Children	port

Instance	`<ports>`
	`  <port data_type="" full="" kind="" max_number="" name="" priority="" role="">{1,unbounded}</port>`
	`</ports>`

Source	`<xs:element name="ports">`
	`  <xs:annotation>`
	`    <xs:documentation>A collection of ports.</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:complexType>`
	`    <xs:sequence>`

```
      <xs:element maxOccurs="unbounded" ref="port"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Element port

Namespace	No namespace
Annotations	Element that specifies eternal interfaces that can connect and be connected to.
Diagram	 Attributes name Name of the port. data_t Type of data that is being accessed/sent from the port. kind Type port_types_define Defines if port is an input or an output port. max_number Defines how many connections can be established to this port. role Type component_role_define Specifies what role this port plays or what this port is connected to. priority Type xs:integer Priority of port. full Describes what to do with incoming items if full. port Element that specifies eternal interfaces that can connect and be connected to. comment Simple comment tag with no attributes.
Properties	content: complex
Used by	Element ports
Model	comment{0,1}
Children	comment

Instance	`<port data_type="" full="" kind="" max_number="" name="" priority="" role="">`
	`  <comment>{0,1}</comment>`
	`</port>`

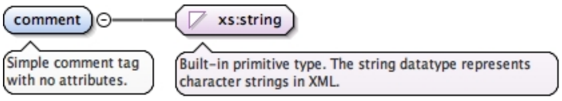
Attributes	QName	Type	Use	
	data_type		required	
		Type of data that is being accessed/sent from the port.		
	full		optional	
		Describes what to do with incoming items if full.		
	kind	port_types_define	required	
		Defines if port is an input or an output port.		
	max_number		optional	
		Defines how many connections can be established to this port.		
	name		required	
		Name of the port.		
	priority	xs:integer	optional	
		Priority of port.		
	role	component_role_define	optional	

	QName	Type	Use	
		Specifies what role this port plays or what this port is connected to.		

Source

```
<xs:element name="port">
  <xs:annotation>
    <xs:documentation>Element that specifies eternal interfaces that can connect and be connected
 to.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" ref="comment"/>
    </xs:sequence>
    <xs:attribute name="name" use="required">
      <xs:annotation>
        <xs:documentation>Name of the port.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="data_type" use="required">
      <xs:annotation>
        <xs:documentation>Type of data that is being accessed/sent from the port.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="kind" use="required" type="port_types_define">
      <xs:annotation>
        <xs:documentation>Defines if port is an input or an output port.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="max_number">
      <xs:annotation>
        <xs:documentation>Defines how many connections can be established to this port.</
xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="role" type="component_role_define">
      <xs:annotation>
        <xs:documentation>Specifies what role this port plays or what this port is connected to.</
xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="priority" type="xs:integer">
      <xs:annotation>
        <xs:documentation>Priority of port.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="full">
      <xs:annotation>
        <xs:documentation>Describes what to do with incoming items if full.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

Element comment

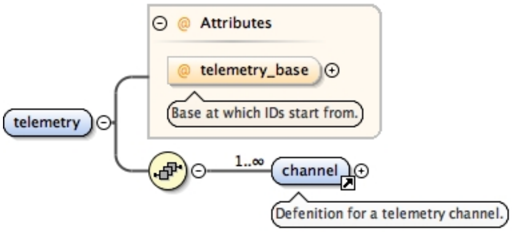
Namespace	No namespace
Annotations	Simple comment tag with no attributes.
Diagram	 comment xs:string Simple comment tag with no attributes. Built-in primitive type. The string datatype represents character strings in XML.
Type	xs:string
Properties	content: simple
Used by	Elements: arg_define/arg, channel, command, event, external_arg_define/arg, parameter, port, return Complex Types: component_define, interface_define

Source

```
<xs:element name="comment" type="xs:string">
  <xs:annotation>
    <xs:documentation>Simple comment tag with no attributes.</xs:documentation>
  </xs:annotation>
</xs:element>
```

Element telemetry

Namespace	No namespace

Diagram	 Attributes telemetry_base Base at which IDs start from. telemetry 1..∞ channel Defenition for a telemetry channel.
Properties	content: `complex`
Used by	Complex Type component_define
Model	channel+
Children	channel

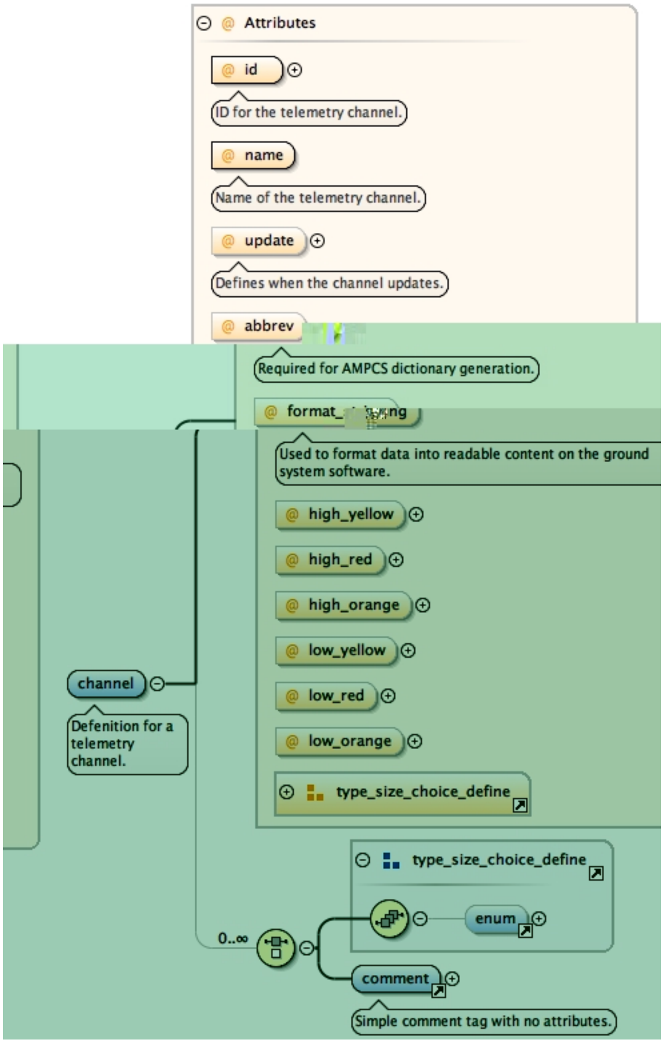
Instance	`<telemetry telemetry_base="">`
	`  <channel abbrev="" data_type="" format_string="" high_orange="" high_red="" high_yellow="" id="" low_orange="" lo`
	`channel>`
	`</telemetry>`

Attributes	**QName** | **Type** | **Use** **telemetry_base** | base_code_define | optional `Base at which IDs start from.`

Source	`<xs:element name="telemetry">`
	`  <xs:complexType>`
	`    <xs:sequence>`
	`      <xs:element maxOccurs="unbounded" ref="channel"/>`
	`    </xs:sequence>`
	`    <xs:attribute name="telemetry_base" type="base_code_define">`
	`      <xs:annotation>`
	`        <xs:documentation>Base at which IDs start from.</xs:documentation>`
	`      </xs:annotation>`
	`    </xs:attribute>`
	`  </xs:complexType>`
	`</xs:element>`

Element `channel`

Namespace	No namespace
Annotations	`Defenition for a telemetry channel.`

<table>
<tr><td>Diagram</td><td>
Attributes
id
ID for the telemetry channel.
name
Name of the telemetry channel.
update
Defines when the channel updates.
abbrev
Required for AMPCS dictionary generation.
format_
Used to format data into readable content on the ground system software.
high_yellow
high_red
high_orange
low_yellow
low_red
low_orange
type_size_choice_define
channel
Defenition for a telemetry channel.
type_size_choice_define
enum
0..∞
comment
Simple comment tag with no attributes.

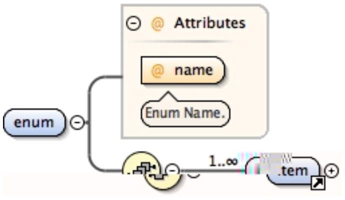
</td></tr>
<tr><td>Properties</td><td>content: complex</td></tr>
<tr><td>Used by</td><td>Element telemetry</td></tr>
<tr><td>Model</td><td>(enum{0,1}) | comment</td></tr>
<tr><td>Children</td><td>comment, enum</td></tr>
<tr><td>Instance</td><td><pre>&lt;channel abbrev="" data_type="" format_string="" high_orange="" high_red="" high_yellow="" id="" low_orange="" low_
  <enum name="">{0,1}</enum>
  <comment>{1,1}</comment>
</channel&gt;</pre></td></tr>
<tr><td>Attributes</td><td>
<table>
<tr><th>QName</th><th>Type</th><th>Use</th><th></th></tr>
<tr><td><b>abbrev</b></td><td></td><td>optional</td><td></td></tr>
<tr><td></td><td colspan="3">Required for AMPCS dictionary generation.</td></tr>
<tr><td><b>data_type</b></td><td>union of(xs:string, restriction of xs:token, restriction of xs:token)</td><td>optional</td><td></td></tr>
<tr><td><b>format_string</b></td><td></td><td>optional</td><td></td></tr>
<tr><td></td><td colspan="3">Used to format data into readable content on the ground system software.</td></tr>
<tr><td><b>high_orange</b></td><td>xs:decimal</td><td>optional</td><td></td></tr>
<tr><td><b>high_red</b></td><td>xs:decimal</td><td>optional</td><td></td></tr>
<tr><td><b>high_yellow</b></td><td>xs:decimal</td><td>optional</td><td></td></tr>
<tr><td><b>id</b></td><td>id_define</td><td>required</td><td></td></tr>
<tr><td></td><td colspan="3">ID for the telemetry channel.</td></tr>
<tr><td><b>low_orange</b></td><td>xs:decimal</td><td>optional</td><td></td></tr>
</table>
</td></tr>
</table>

QName	Type	Use	
low_red	xs:decimal	optional	
low_yellow	xs:decimal	optional	
name		required	
	Name of the telemetry channel.		
size	xs:nonNegativeInteger	optional	
	The size of the argument.		
type	union of(xs:string, restriction of xs:token, restriction of xs:token)	optional	
update	channel_update_define	optional	
	Defines when the channel updates.		

Source

```
<xs:element name="channel">
  <xs:annotation>
    <xs:documentation>Defenition for a telemetry channel.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:group ref="type_size_choice_define"/>
      <xs:element ref="comment"/>
    </xs:choice>
    <xs:attribute name="id" use="required" type="id_define">
      <xs:annotation>
        <xs:documentation>ID for the telemetry channel.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="name" use="required">
      <xs:annotation>
        <xs:documentation>Name of the telemetry channel.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="update" type="channel_update_define">
      <xs:annotation>
        <xs:documentation>Defines when the channel updates.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="abbrev">
      <xs:annotation>
        <xs:documentation>Required for AMPCS dictionary generation.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="format_string">
      <xs:annotation>
        <xs:documentation>Used to format data into readable content on the ground system software.</
xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="high_yellow" type="xs:decimal"/>
    <xs:attribute name="high_red" type="xs:decimal"/>
    <xs:attribute name="high_orange" type="xs:decimal"/>
    <xs:attribute name="low_yellow" type="xs:decimal"/>
    <xs:attribute name="low_red" type="xs:decimal"/>
    <xs:attribute name="low_orange" type="xs:decimal"/>
    <xs:attributeGroup ref="type_size_choice_define"/>
  </xs:complexType>
</xs:element>
```

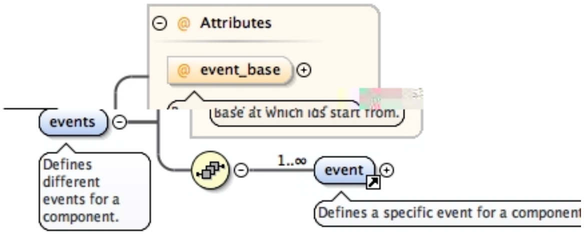
Element enum

Namespace	No namespace
Diagram	

	Complex Type	data_type_and_default_define
	Element	parameter

	`</xs:element>`

Element events

Namespace	No namespace
Annotations	Defines different events for a component.
Diagram	 Attributes event_base Base at which ids start from. events Defines different events for a component. 1..∞ event Defines a specific event for a component
Properties	content: complex
Used by	Complex Type component_define
Model	event+
Children	event

Instance	`<events event_base="">`
	`  <event format_string="" id="" name="" severity="" throttle="">{1,unbounded}</event>`
	`</events>`

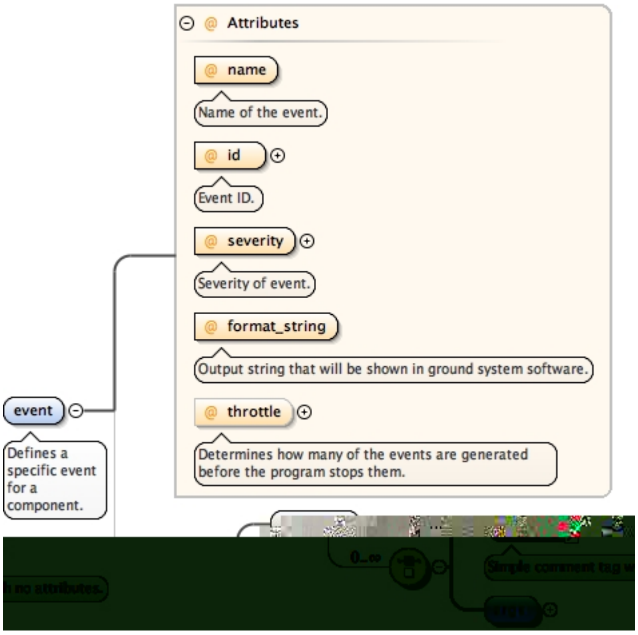
Attributes	**QName** | **Type** | **Use** **event_base** | base_code_define | optional Base at which ids start from.

Source	`<xs:element name="events">`
	`  <xs:annotation>`
	`    <xs:documentation>Defines different events for a component.</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:complexType>`
	`    <xs:sequence>`
	`      <xs:element maxOccurs="unbounded" ref="event"/>`
	`    </xs:sequence>`
	`    <xs:attribute name="event_base" type="base_code_define">`
	`      <xs:annotation>`
	`        <xs:documentation>Base at which ids start from.</xs:documentation>`
	`      </xs:annotation>`
	`    </xs:attribute>`
	`  </xs:complexType>`
	`</xs:element>`

Element event

Namespace	No namespace
Annotations	Defines a specific event for a component.

Diagram


Attributes
name
Name of the event.
id
Event ID.
severity
Severity of event.
format_string
Output string that will be shown in ground system software.
throttle
Determines how many of the events are generated before the program stops them.
event
Defines a specific event for a component.
0..∞

Properties	content:	complex
Used by	Element	events
Model	comment | args	
Children	args, comment	

Instance

```
<event format_string="" id="" name="" severity="" throttle="">
  <comment>{1,1}</comment>
  <args>{1,1}</args>
</event>
```

Attributes

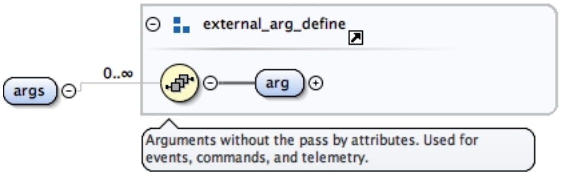
QName	Type	Use	
format_string		required	
	Output string that will be shown in ground system software.		
id	id_define	required	
	Event ID.		
name		required	
	Name of the event.		
severity	severity_define	required	
	Severity of event.		
throttle	xs:nonNegativeInteger	optional	
	Determines how many of the events are generated before the program stops them.		

Source

```
<xs:element name="event">
  <xs:annotation>
    <xs:documentation>Defines a specific event for a component.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="comment"/>
      <xs:element name="args">
        <xs:complexType>
          <xs:group minOccurs="0" maxOccurs="unbounded" ref="external_arg_define"/>
        </xs:complexType>
      </xs:element>
    </xs:choice>
    <xs:attribute name="name" use="required">
      <xs:annotation>
        <xs:documentation>Name of the event.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="id" use="required" type="id_define">
      <xs:annotation>
        <xs:documentation>Event ID.</xs:documentation>
      </xs:annotation>
```

	`</xs:attribute>`
	`<xs:attribute name="severity" use="required" type="severity_define">`
	`<xs:annotation>`
	`<xs:documentation>Severity of event.</xs:documentation>`
	`</xs:annotation>`
	`</xs:attribute>`
	`<xs:attribute name="format_string" use="required">`
	`<xs:annotation>`
	`<xs:documentation>Output string that will be shown in ground system software.</xs:documentation>`
	`</xs:annotation>`
	`</xs:attribute>`
	`<xs:attribute name="throttle" type="xs:nonNegativeInteger">`
	`<xs:annotation>`
	`<xs:documentation>Determines how many of the events are generated before the program stops them.</xs:documentation>`
	`</xs:annotation>`
	`</xs:attribute>`
	`</xs:complexType>`
	`</xs:element>`

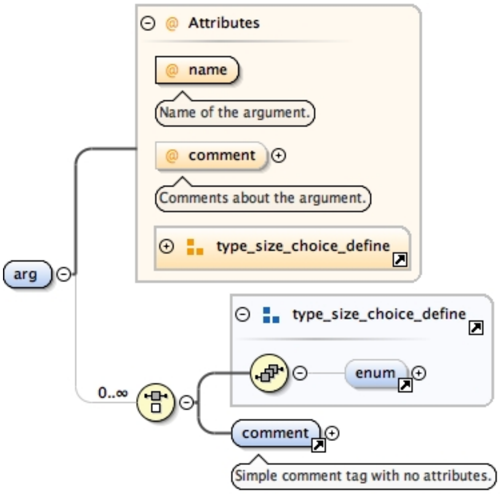
Element event / args

Namespace	No namespace
Diagram	 args 0..∞ external_arg_define arg Arguments without the pass by attributes. Used for events, commands, and telemetry.
Properties	content: complex
Model	arg
Children	arg

Instance	`<args>`
	`<arg comment="" data_type="" name="" size="" type="">{1,1}</arg>`
	`</args>`

Source	`<xs:element name="args">`
	`<xs:complexType>`
	`<xs:group minOccurs="0" maxOccurs="unbounded" ref="external_arg_define"/>`
	`</xs:complexType>`
	`</xs:element>`

Element external_arg_define / arg

Namespace	No namespace
Diagram	 Attributes name Name of the argument. comment Comments about the argument. type_size_choice_define arg 0..∞ type_size_choice_define enum comment Simple comment tag with no attributes.
Properties	content: complex
Model	(enum{0,1}) | comment
Children	comment, enum
Instance	`<arg comment="" data_type="" name="" size="" type="">`

```
  <enum name="">{0,1}</enum>
  <comment>{1,1}</comment>
</arg>
```

Attributes

QName	Type	Use	
comment	xs:string	optional	
	Comments about the argument.		
data_type	union of(xs:string, restriction of xs:token, restriction of xs:token)	optional	
name		required	
	Name of the argument.		
size	xs:nonNegativeInteger	optional	
	The size of the argument.		
type	union of(xs:string, restriction of xs:token, restriction of xs:token)	optional	

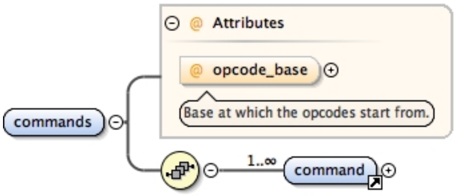
Source

```
<xs:element name="arg">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:group ref="type_size_choice_define"/>
      <xs:element ref="comment"/>
    </xs:choice>
    <xs:attribute name="name" use="required">
      <xs:annotation>
        <xs:documentation>Name of the argument.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="comment" type="xs:string">
      <xs:annotation>
        <xs:documentation>Comments about the argument.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attributeGroup ref="type_size_choice_define"/>
  </xs:complexType>
</xs:element>
```

Element commands

Namespace	No namespace

Diagram



Properties	content:	complex
Used by	Complex Type	component_define
Model	command+	
Children	command	

Instance

```
<commands opcode_base="">
  <command full="" kind="" mnemonic="" opcode="" priority="">{1,unbounded}</command>
</commands>
```

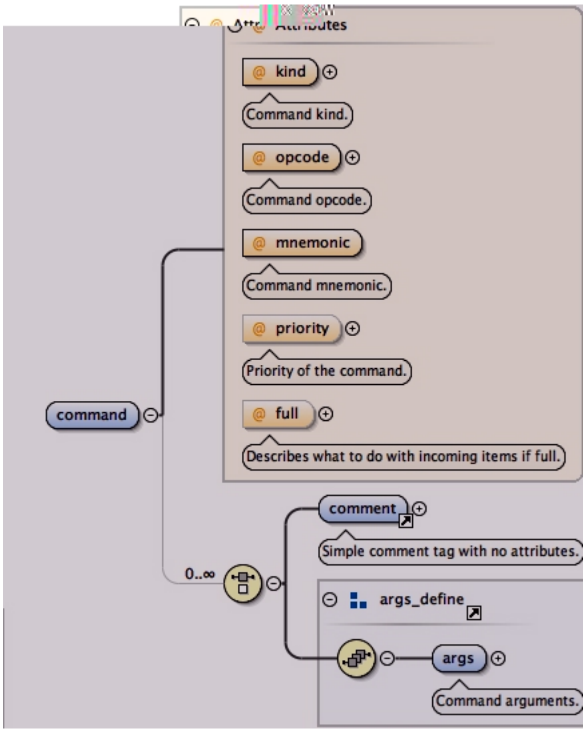
Attributes

QName	Type	Use	
opcode_base	base_code_define	optional	
	Base at which the opcodes start from.		

Source

```
<xs:element name="commands">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="command"/>
    </xs:sequence>
    <xs:attribute name="opcode_base" type="base_code_define">
      <xs:annotation>
        <xs:documentation>Base at which the opcodes start from.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

Element command

Namespace	No namespace
Diagram	
Properties	content: `complex`
Used by	Element commands
Model	comment | (args)
Children	args, comment

Instance

```
<command full="" kind="" mnemonic="" opcode="" priority="">
  <comment>{1,1}</comment>
  <args>{1,1}</args>
</command>
```

Attributes

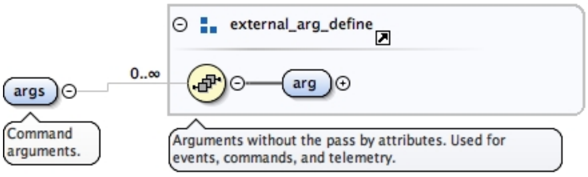
QName	Type	Use	
full	full_items_define	optional	
	`Describes what to do with incoming items if full.`		
kind	command_kind_define	required	
	`Command kind.`		
mnemonic		required	
	`Command mnemonic.`		
opcode	id_define	required	
	`Command opcode.`		
priority	xs:integer	optional	
	`Priority of the command.`		

Source

```
<xs:element name="command">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="comment"/>
      <xs:group ref="args_define"/>
    </xs:choice>
    <xs:attribute name="kind" use="required" type="command_kind_define">
      <xs:annotation>
        <xs:documentation>Command kind.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
```

	`    <xs:attribute name="opcode" use="required" type="id_define">`
	`      <xs:annotation>`
	`        <xs:documentation>Command opcode.</xs:documentation>`
	`      </xs:annotation>`
	`    </xs:attribute>`
	`    <xs:attribute name="mnemonic" use="required">`
	`      <xs:annotation>`
	`        <xs:documentation>Command mnemonic.</xs:documentation>`
	`      </xs:annotation>`
	`    </xs:attribute>`
	`    <xs:attribute name="priority" type="xs:integer">`
	`      <xs:annotation>`
	`        <xs:documentation>Priority of the command.</xs:documentation>`
	`      </xs:annotation>`
	`    </xs:attribute>`
	`    <xs:attribute name="full" type="full_items_define">`
	`      <xs:annotation>`
	`        <xs:documentation>Describes what to do with incoming items if full.</xs:documentation>`
	`      </xs:annotation>`
	`    </xs:attribute>`
	`  </xs:complexType>`
	`</xs:element>`

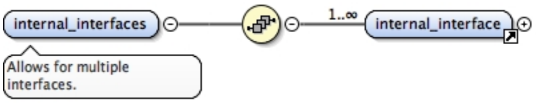
Element `args_define / args`

Namespace	No namespace
Annotations	`Command arguments.`
Diagram	 external_arg_define 0..∞ args arg Command arguments. Arguments without the pass by attributes. Used for events, commands, and telemetry.
Properties	content: `complex`
Model	arg
Children	arg

Instance	`<args>`
	`  <arg comment="" data_type="" name="" size="" type="">{1,1}</arg>`
	`</args>`

Source	`<xs:element name="args">`
	`  <xs:annotation>`
	`    <xs:documentation>Command arguments.</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:complexType>`
	`    <xs:group minOccurs="0" maxOccurs="unbounded" ref="external_arg_define"/>`
	`  </xs:complexType>`
	`</xs:element>`

Element `internal_interfaces`

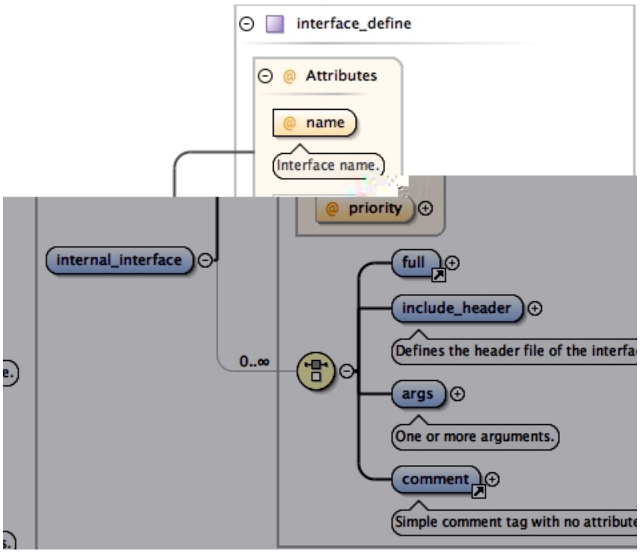
Namespace	No namespace
Annotations	`Allows for multiple interfaces.`
Diagram	 internal_interfaces 1..∞ internal_interface Allows for multiple interfaces.
Properties	content: `complex`
Used by	Complex Type component_define
Model	internal_interface+
Children	internal_interface

Instance	`<internal_interfaces>`
	`  <internal_interface name="" priority="">{1,unbounded}</internal_interface>`
	`</internal_interfaces>`

Source	`<xs:element name="internal_interfaces">`
	`  <xs:annotation>`
	`    <xs:documentation>Allows for multiple interfaces.</xs:documentation>`
	`  </xs:annotation>`

```
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="internal_interface"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Element internal_interface

Namespace	No namespace
Diagram	 interface_define Attributes name Interface name. priority internal_interface 0..∞ full include_header Defines the header file of the interfa args One or more arguments. comment Simple comment tag with no attribute e. s.
Type	interface_define
Properties	content: `complex`
Used by	Element internal_interfaces
Model	full | include_header | args | comment
Children	args, comment, full, include_header

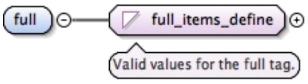
Instance	`<internal_interface name="" priority="">`
	`  <full>{1,1}</full>`
	`  <include_header>{1,1}</include_header>`
	`  <args>{1,1}</args>`
	`  <comment>{1,1}</comment>`
	`</internal_interface>`

Attributes

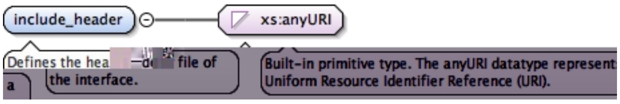
QName	Type	Use	
name		required	
	`Interface name.`		
priority	xs:integer	optional	

Source	`<xs:element name="internal_interface" type="interface_define"/>`

Element full

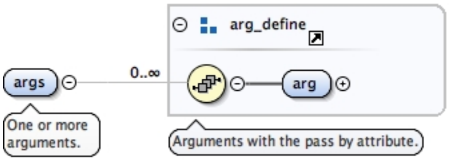
Namespace	No namespace
Diagram	
Type	full_items_define
Properties	content: `simple`
Facets	enumeration `drop` enumeration `assert` enumeration `block`
Used by	Complex Type interface_define
Source	`<xs:element name="full" type="full_items_define"/>`

Element `interface_define / include_header`

Namespace	No namespace
Annotations	`Defines the header file of the interface.`
Diagram	 include_header xs:anyURI Defines the hea file of a the interface. Built-in primitive type. The anyURI datatype represent Uniform Resource Identifier Reference (URI).
Type	xs:anyURI
Properties	content: `simple`

Source	`<xs:element name="include_header" type="xs:anyURI">`
	`  <xs:annotation>`
	`    <xs:documentation>Defines the header file of the interface.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:element>`

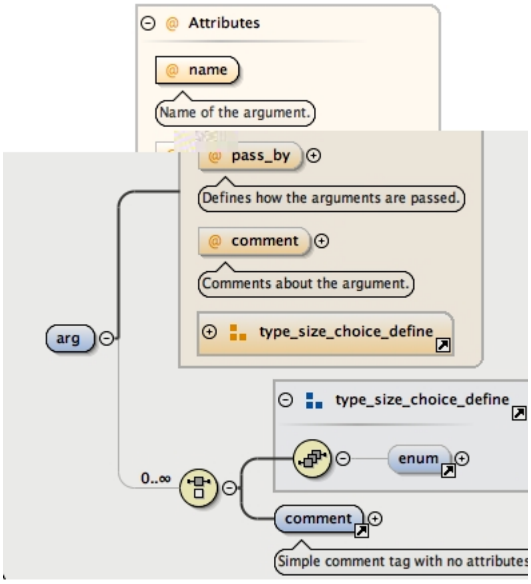
Element `interface_define / args`

Namespace	No namespace
Annotations	`One or more arguments.`
Diagram	 arg_define args 0..∞ arg One or more arguments. Arguments with the pass by attribute.
Properties	content: `complex`
Model	arg
Children	arg

Instance	`<args>`
	`  <arg comment="" data_type="" name="" pass_by="" size="" type="">{1,1}</arg>`
	`</args>`

Source	`<xs:element name="args">`
	`  <xs:annotation>`
	`    <xs:documentation>One or more arguments.</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:complexType>`
	`    <xs:group minOccurs="0" maxOccurs="unbounded" ref="arg_define"/>`
	`  </xs:complexType>`
	`</xs:element>`

Element `arg_define / arg`

Namespace	No namespace
Diagram	 Attributes name Name of the argument. pass_by Defines how the arguments are passed. comment Comments about the argument. type_size_choice_define arg type_size_choice_define enum 0..∞ comment Simple comment tag with no attributes

Properties	content: complex
Model	(enum{0,1}) | comment
Children	comment, enum

Instance	`<arg comment="" data_type="" name="" pass_by="" size="" type="">`
	`  <enum name="">{0,1}</enum>`
	`  <comment>{1,1}</comment>`
	`</arg>`

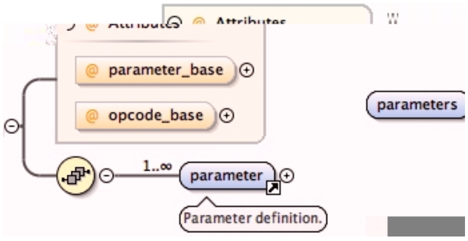
Attributes	

QName	Type	Use	
comment	xs:string	optional	
	Comments about the argument.		
data_type	union of(xs:string, restriction of xs:token, restriction of xs:token)	optional	
name		required	
	Name of the argument.		
pass_by	pass_by_define	optional	
	Defines how the arguments are passed.		
size	xs:nonNegativeInteger	optional	
	The size of the argument.		
type	union of(xs:string, restriction of xs:token, restriction of xs:token)	optional	

Source

```
<xs:element name="arg">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:group ref="type_size_choice_define"/>
      <xs:element ref="comment"/>
    </xs:choice>
    <xs:attribute name="name" use="required">
      <xs:annotation>
        <xs:documentation>Name of the argument.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="pass_by" type="pass_by_define">
      <xs:annotation>
        <xs:documentation>Defines how the arguments are passed.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="comment" type="xs:string">
      <xs:annotation>
        <xs:documentation>Comments about the argument.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attributeGroup ref="type_size_choice_define"/>
  </xs:complexType>
</xs:element>
```

Element parameters

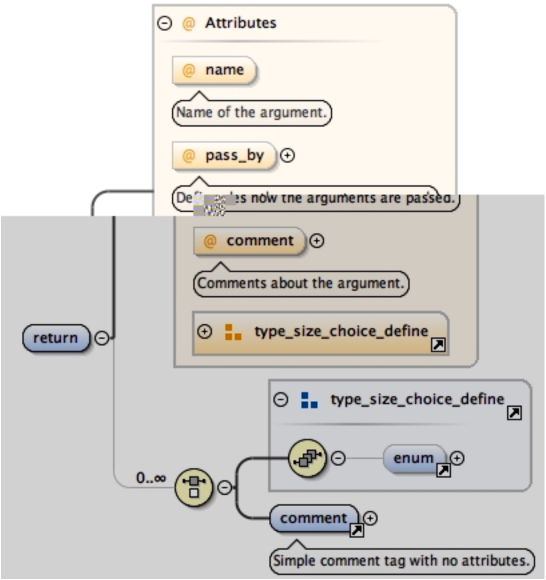
Namespace	No namespace
Diagram	 Attributes parameter_base opcode_base parameters 1..∞ parameter Parameter definition.
Properties	content: complex
Used by	Complex Type component_define
Model	parameter+
Children	parameter
Instance	`<parameters opcode_base="" parameter_base="">`

<table>
<tr><td></td><td><pre> <parameter data_type="" default="" id="" name="" save_opcode="" set_opcode="" size="">{1,unbounded}</parameter>
</parameters></pre></td></tr>
<tr><td>Attributes</td><td><table><tr><th>QName</th><th>Type</th></tr></table></td></tr>
</table>

Properties	content:	`complex`
Used by	Element	parameters

```
</xs:element>
```

Element return

Namespace	No namespace
Diagram	
Properties	content: complex
Model	(enum{0,1}) | comment
Children	comment, enum

Instance

```
<return comment="" data_type="" name="" pass_by="" size="" type="">
  <enum name="">{0,1}</enum>
  <comment>{1,1}</comment>
</return>
```

Attributes

QName	Type	Use	
comment	xs:string	optional	
	Comments about the argument.		
data_type	union of(xs:string, restriction of xs:token, restriction of xs:token)	optional	
name		optional	
	Name of the argument.		
pass_by	pass_by_define	optional	
	Defines how the arguments are passed.		
size	xs:nonNegativeInteger	optional	
	The size of the argument.		
type	union of(xs:string, restriction of xs:token, restriction of xs:token)	optional	

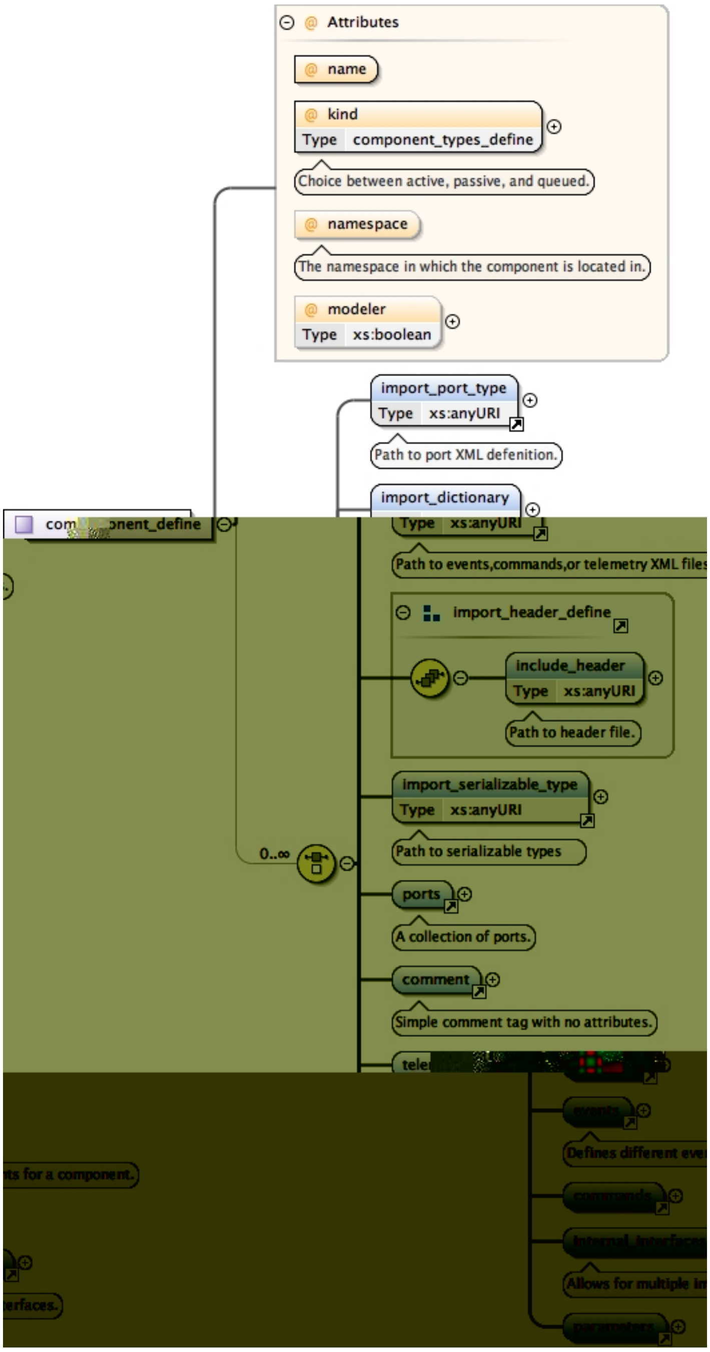
Source

```
<xs:element name="return">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:group ref="type_size_choice_define"/>
      <xs:element ref="comment"/>
    </xs:choice>
    <xs:attribute name="name">
      <xs:annotation>
        <xs:documentation>Name of the argument.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="pass_by" type="pass_by_define">
      <xs:annotation>
        <xs:documentation>Defines how the arguments are passed.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="comment" type="xs:string">
```

```
        <xs:annotation>
          <xs:documentation>Comments about the argument.</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attributeGroup ref="type_size_choice_define"/>
    </xs:complexType>
  </xs:element>
```

Complex Type(s)

Complex Type `component_define`

Namespace	No namespace
Diagram	
Used by	Element component
Model	import_port_type | import_dictionary | (include_header) | import_serializable_type | ports | comment | telemetry | events | commands | internal_interfaces | parameters
Children	commands, comment, events, import_dictionary, import_port_type, import_serializable_type, include_header, internal_interfaces, parameters, ports, telemetry

Attributes

QName	Type	Use	
kind	component_types_define	required	
	`Choice between active, passive, and queued.`		
modeler	xs:boolean	optional	
name		required	
namespace		optional	
	`The namespace in which the component is located in.`		

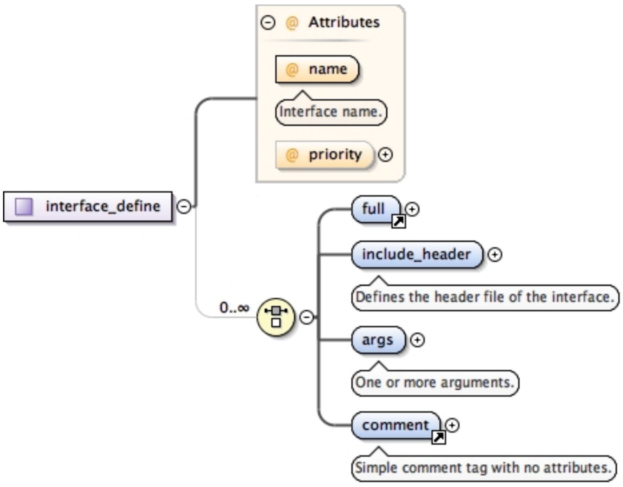
Source

```
<xs:complexType name="component_define">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="import_port_type"/>
    <xs:element ref="import_dictionary"/>
    <xs:group ref="import_header_define"/>
    <xs:element ref="import_serializable_type"/>
    <xs:element ref="ports"/>
    <xs:element ref="comment"/>
    <xs:element ref="telemetry"/>
    <xs:element ref="events"/>
    <xs:element ref="commands"/>
    <xs:element ref="internal_interfaces"/>
    <xs:element ref="parameters"/>
  </xs:choice>
  <xs:attribute name="name" use="required"/>
  <xs:attribute name="kind" use="required" type="component_types_define">
    <xs:annotation>
      <xs:documentation>Choice between active, passive, and queued.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="namespace">
    <xs:annotation>
      <xs:documentation>The namespace in which the component is located in.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="modeler" type="xs:boolean"/>
</xs:complexType>
```

Complex Type `interface_define`

Namespace	No namespace

Diagram



Used by	Element	internal_interface
Model	full | include_header | args | comment	
Children	args, comment, full, include_header	

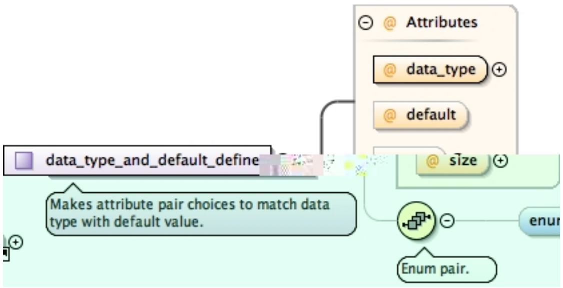
Attributes

QName	Type	Use	
name		required	
	`Interface name.`		
priority	xs:integer	optional	

Source

```
<xs:complexType name="interface_define">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="full"/>
    <xs:element name="include_header" type="xs:anyURI">
      <xs:annotation>
        <xs:documentation>Defines the header file of the interface.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="args">
      <xs:annotation>
        <xs:documentation>One or more arguments.</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:group minOccurs="0" maxOccurs="unbounded" ref="arg_define"/>
      </xs:complexType>
    </xs:element>
    <xs:element ref="comment"/>
  </xs:choice>
  <xs:attribute name="name" use="required">
    <xs:annotation>
      <xs:documentation>Interface name.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="priority" type="xs:integer"/>
</xs:complexType>
```

Complex Type `data_type_and_default_define`

Namespace	No namespace
Annotations	Makes attribute pair choices to match data type with default value.
Diagram	 Attributes data_type default size data_type_and_default_define Makes attribute pair choices to match data type with default value. enur Enum pair.
Used by	Element parameter
Model	enum{0,1}
Children	enum

Attributes

QName	Type	Use	
data_type	union of(not_user_cpp_type_define, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token)	required	
default		optional	
size	positive_integer_define	optional	

Source

```
<xs:complexType name="data_type_and_default_define">
  <xs:annotation>
    <xs:documentation>Makes attribute pair choices to match data type with default value.</
xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:annotation>
      <xs:documentation>Enum pair.</xs:documentation>
    </xs:annotation>
    <xs:element minOccurs="0" ref="enum"/>
  </xs:sequence>
  <xs:attribute name="data_type" use="required">
    <xs:simpleType>
      <xs:union memberTypes="not_user_cpp_type_define">
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="ENUM"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="string"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="I8"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="U8"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="I16"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="U16"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="I32"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="U32"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="I64"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="U64"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="F32"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="F64"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="NATIVE_INT_TYPE"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="NATIVE_UINT_TYPE"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:union>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="default"/>
  <xs:attribute name="size" type="positive_integer_define"/>
</xs:complexType>
```

Simple Type(s)

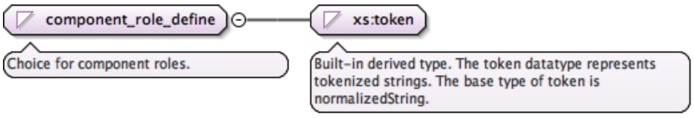
Simple Type `port_types_define`

<table>
<tr><td>Namespace</td><td colspan="2">No namespace</td></tr>
<tr><td>Annotations</td><td colspan="2"><code>Choice between different port types.</code></td></tr>
<tr><td>Diagram</td><td colspan="2">

port_types_define
xs:token
Choice between different port types.
Built-in derived type. The token datatype represents tokenized
pe of token is
normalizedSt

</td></tr>
<tr><td>Type</td><td colspan="2">restriction of xs:token</td></tr>
<tr><td>Facets</td><td>enumeration</td><td><code>input</code></td></tr>
<tr><td></td><td>enumeration</td><td><code>sync_input</code></td></tr>
<tr><td></td><td>enumeration</td><td><code>guarded_input</code></td></tr>
<tr><td></td><td>enumeration</td><td><code>async_input</code></td></tr>
<tr><td></td><td>enumeration</td><td><code>model_input</code></td></tr>
<tr><td></td><td>enumeration</td><td><code>output</code></td></tr>
<tr><td>Used by</td><td>Attribute</td><td>port/@kind</td></tr>
<tr><td>Source</td><td colspan="2"><pre>
<xs:simpleType name="port_types_define">
  <xs:annotation>
    <xs:documentation>Choice between different port types.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:token">
    <xs:enumeration value="input"/>
    <xs:enumeration value="sync_input"/>
    <xs:enumeration value="guarded_input"/>
    <xs:enumeration value="async_input"/>
    <xs:enumeration value="model_input"/>
    <xs:enumeration value="output"/>
  </xs:restriction>
</xs:simpleType>
</pre></td></tr>
</table>

Simple Type `component_role_define`

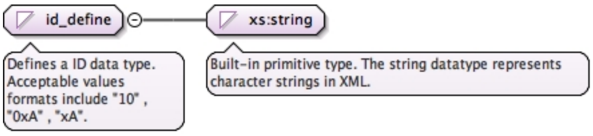
<table>
<tr><td>Namespace</td><td colspan="2">No namespace</td></tr>
<tr><td>Annotations</td><td colspan="2"><code>Choice for component roles.</code></td></tr>
<tr><td>Diagram</td><td colspan="2">

component_role_define
xs:token
Choice for component roles.
Built-in derived type. The token datatype represents tokenized strings. The base type of token is normalizedString.

</td></tr>
<tr><td>Type</td><td colspan="2">restriction of xs:token</td></tr>
<tr><td>Facets</td><td>enumeration</td><td><code>LogEvent</code></td></tr>
<tr><td></td><td>enumeration</td><td><code>LogTextEvent</code></td></tr>
<tr><td></td><td>enumeration</td><td><code>TimeGet</code></td></tr>
<tr><td></td><td>enumeration</td><td><code>ParamSet</code></td></tr>
<tr><td></td><td>enumeration</td><td><code>ParamGet</code></td></tr>
<tr><td></td><td>enumeration</td><td><code>Telemetry</code></td></tr>
<tr><td></td><td>enumeration</td><td><code>CmdRegistration</code></td></tr>
<tr><td></td><td>enumeration</td><td><code>Cmd</code></td></tr>
<tr><td></td><td>enumeration</td><td><code>CmdResponse</code></td></tr>
<tr><td>Used by</td><td>Attribute</td><td>port/@role</td></tr>
<tr><td>Source</td><td colspan="2"><pre>
<xs:simpleType name="component_role_define">
  <xs:annotation>
    <xs:documentation>Choice for component roles.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:token">
    <xs:enumeration value="LogEvent"/>
    <xs:enumeration value="LogTextEvent"/>
    <xs:enumeration value="TimeGet"/>
    <xs:enumeration value="ParamSet"/>
</pre></td></tr>
</table>

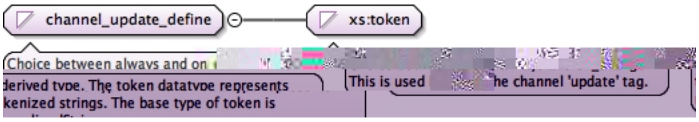
	`    <xs:enumeration value="ParamGet"/>`
	`    <xs:enumeration value="Telemetry"/>`
	`    <xs:enumeration value="CmdRegistration"/>`
	`    <xs:enumeration value="Cmd"/>`
	`    <xs:enumeration value="CmdResponse"/>`
	`  </xs:restriction>`
	`</xs:simpleType>`

Simple Type `id_define`

Namespace	No namespace
Annotations	Defines a ID data type. Acceptable values formats include "10" , "0xA" , "xA".
Diagram	 id_define xs:string Defines a ID data type. Acceptable values formats include "10" , "0xA" , "xA". Built-in primitive type. The string datatype represents character strings in XML.
Type	restriction of xs:string
Facets	pattern ((0?x\d+)|\d+)
Used by	Attributes channel/@id, command/@opcode, event/@id, parameter/@id, parameter/@save_opcode, parameter/@set_opcode

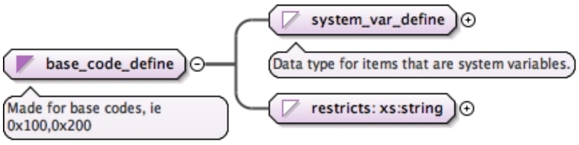
Source	`<xs:simpleType name="id_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Defines a ID data type. Acceptable values formats include "10" , "0xA" , "xA".</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="xs:string">`
	`    <xs:pattern value="((0?x\d+)|\d+)"/>`
	`  </xs:restriction>`
	`</xs:simpleType>`

Simple Type `channel_update_define`

Namespace	No namespace
Annotations	Choice between always and on_change. This is used in the channel 'update' tag.
Diagram	 channel_update_define xs:token Choice between always and on derived type. The token datatype represents kenized strings. The base type of token is This is used he channel 'update' tag.
Type	restriction of xs:token
Facets	enumeration always enumeration on_change
Used by	Attribute channel/@update

Source	`<xs:simpleType name="channel_update_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Choice between always and on_change. This is used in the channel 'update' tag.</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="xs:token">`
	`    <xs:enumeration value="always"/>`
	`    <xs:enumeration value="on_change"/>`
	`  </xs:restriction>`
	`</xs:simpleType>`

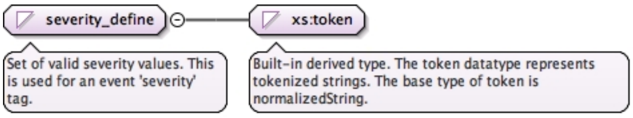
Simple Type `base_code_define`

Namespace	No namespace
Annotations	Made for base codes, ie 0x100,0x200
Diagram	 base_code_define Made for base codes, ie 0x100,0x200 system_var_define Data type for items that are system variables. restricts: xs:string
Type	union of(system_var_define, restriction of xs:string)

Used by	Attributes	commands/@opcode_base, events/@event_base, parameters/@opcode_base, parameters/@parameter_base, telemetry/@telemetry_base

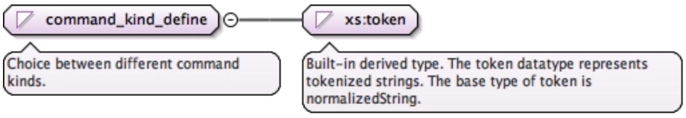
Source	`<xs:simpleType name="base_code_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Made for base codes, ie 0x100,0x200</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:union memberTypes="system_var_define">`
	`    <xs:simpleType>`
	`      <xs:restriction base="xs:string">`
	`        <xs:pattern value="(((0?x\d+)|\d+)(,?))+"/>`
	`      </xs:restriction>`
	`    </xs:simpleType>`
	`  </xs:union>`
	`</xs:simpleType>`	

Simple Type severity_define

Namespace	No namespace	
Annotations	Set of valid severity values. This is used for an event 'severity' tag.	
Diagram	 severity_define xs:token Set of valid severity values. This is used for an event 'severity' tag. Built-in derived type. The token datatype represents tokenized strings. The base type of token is normalizedString.	
Type	restriction of xs:token	
Facets	enumeration	COMMAND
	enumeration	ACTIVITY_LO
	enumeration	ACTIVITY_HI
	enumeration	WARNING_LO
	enumeration	WARNING_HI
	enumeration	DIAGNOSTIC
	enumeration	FATAL
Used by	Attribute	event/@severity

Source	`<xs:simpleType name="severity_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Set of valid severity values. This is used for an event 'severity' tag.</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="xs:token">`
	`    <xs:enumeration value="COMMAND"/>`
	`    <xs:enumeration value="ACTIVITY_LO"/>`
	`    <xs:enumeration value="ACTIVITY_HI"/>`
	`    <xs:enumeration value="WARNING_LO"/>`
	`    <xs:enumeration value="WARNING_HI"/>`
	`    <xs:enumeration value="DIAGNOSTIC"/>`
	`    <xs:enumeration value="FATAL"/>`
	`  </xs:restriction>`
	`</xs:simpleType>`	

Simple Type command_kind_define

Namespace	No namespace	
Annotations	Choice between different command kinds.	
Diagram	 command_kind_define xs:token Choice between different command kinds. Built-in derived type. The token datatype represents tokenized strings. The base type of token is normalizedString.	
Type	restriction of xs:token	
Facets	enumeration	async
	enumeration	sync
	enumeration	guarded
Used by	Attribute	command/@kind
Source	`<xs:simpleType name="command_kind_define">`	

```
  <xs:annotation>
    <xs:documentation>Choice between different command kinds.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:token">
    <xs:enumeration value="async"/>
    <xs:enumeration value="sync"/>
    <xs:enumeration value="guarded"/>
  </xs:restriction>
</xs:simpleType>
```

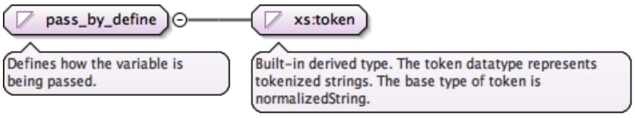
Simple Type `full_items_define`

Namespace	No namespace	
Annotations	Valid values for the full tag.	
Diagram	full_items_define xs:token Valid values for the full tag. Built-in derived type. The token datatype represents tokenized strings. The base type of token is normalizedString.	
Type	restriction of xs:token	
Facets	enumeration	drop
	enumeration	assert
	enumeration	block
Used by	Attribute	command/@full
	Element	full

Source

```
<xs:simpleType name="full_items_define">
  <xs:annotation>
    <xs:documentation>Valid values for the full tag.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:token">
    <xs:enumeration value="drop"/>
    <xs:enumeration value="assert"/>
    <xs:enumeration value="block"/>
  </xs:restriction>
</xs:simpleType>
```

Simple Type `pass_by_define`

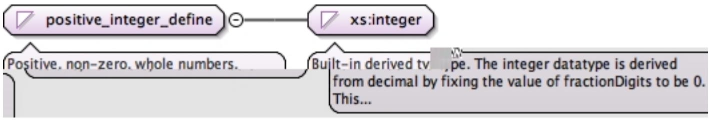
Namespace	No namespace	
Annotations	Defines how the variable is being passed.	
Diagram	 pass_by_define xs:token Defines how the variable is being passed. Built-in derived type. The token datatype represents tokenized strings. The base type of token is normalizedString.	
Type	restriction of xs:token	
Facets	enumeration	reference
	enumeration	value
	enumeration	pointer
Used by	Attributes	arg_define/arg/@pass_by, return/@pass_by

Source

```
<xs:simpleType name="pass_by_define">
  <xs:annotation>
    <xs:documentation>Defines how the variable is being passed.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:token">
    <xs:enumeration value="reference"/>
    <xs:enumeration value="value"/>
    <xs:enumeration value="pointer"/>
  </xs:restriction>
</xs:simpleType>
```

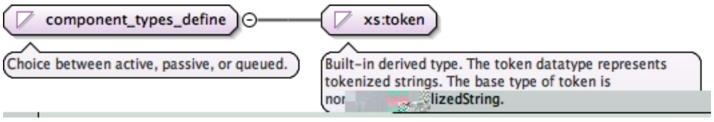
Simple Type `positive_integer_define`

Namespace	No namespace
Annotations	Positive, non-zero, whole numbers.

Diagram	 positive_integer_define xs:integer Positive, non-zero, whole numbers. Built-in derived tv pe. The integer datatype is derived from decimal by fixing the value of fractionDigits to be 0. This...
Type	restriction of xs:integer
Facets	minInclusive 1
Used by	Attribute data_type_and_default_define/@size

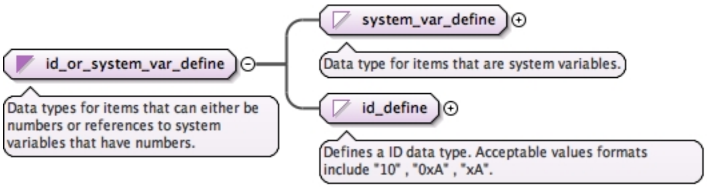
Source	`<xs:simpleType name="positive_integer_define">`
	`<xs:annotation>`
	`<xs:documentation>Positive, non-zero, whole numbers.</xs:documentation>`
	`</xs:annotation>`
	`<xs:restriction base="xs:integer">`
	`<xs:minInclusive value="1"/>`
	`</xs:restriction>`
	`</xs:simpleType>`

Simple Type `component_types_define`

Namespace	No namespace
Annotations	Choice between active, passive, or queued.
Diagram	 component_types_define xs:token Choice between active, passive, or queued. Built-in derived type. The token datatype represents tokenized strings. The base type of token is nor lizedString.
Type	restriction of xs:token
Facets	enumeration active enumeration passive enumeration queued
Used by	Attribute component_define/@kind

Source	`<xs:simpleType name="component_types_define">`
	`<xs:annotation>`
	`<xs:documentation>Choice between active, passive, or queued.</xs:documentation>`
	`</xs:annotation>`
	`<xs:restriction base="xs:token">`
	`<xs:enumeration value="active"/>`
	`<xs:enumeration value="passive"/>`
	`<xs:enumeration value="queued"/>`
	`</xs:restriction>`
	`</xs:simpleType>`

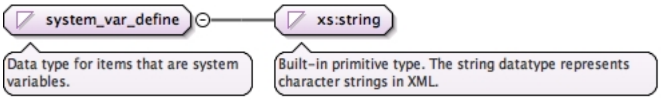
Simple Type `id_or_system_var_define`

Namespace	No namespace
Annotations	Data types for items that can either be numbers or references to system variables that have numbers.
Diagram	 system_var_define Data type for items that are system variables. id_or_system_var_define Data types for items that can either be numbers or references to system variables that have numbers. id_define Defines a ID data type. Acceptable values formats include "10" , "0xA" , "xA".
Type	union of(system_var_define, id_define)

Source	`<xs:simpleType name="id_or_system_var_define">`
	`<xs:annotation>`
	`<xs:documentation>Data types for items that can either be numbers or references to system variables that have numbers.</xs:documentation>`
	`</xs:annotation>`
	`<xs:union memberTypes="system_var_define id_define"/>`
	`</xs:simpleType>`

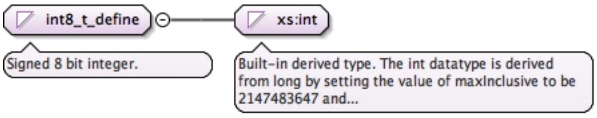
Simple Type `system_var_define`

Namespace	No namespace

<table>
<tr><td>Annotations</td><td>Data type for items that are system variables.</td></tr>
<tr><td>Diagram</td><td>
system_var_define
xs:string
Data type for items that are system variables.
Built-in primitive type. The string datatype represents character strings in XML.

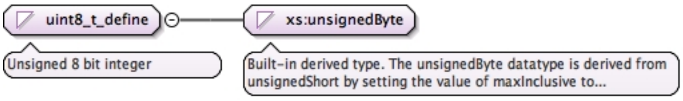
</td></tr>
<tr><td>Type</td><td>restriction of xs:string</td></tr>
<tr><td>Facets</td><td>pattern $[\w|_|\-]+</td></tr>
<tr><td>Source</td><td><xs:simpleType name="system_var_define">
  <xs:annotation>
    <xs:documentation>Data type for items that are system variables.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="$[\w|_|\-]+"/>
  </xs:restriction>
</xs:simpleType></td></tr>
</table>

Simple Type int8_t_define

<table>
<tr><td>Namespace</td><td>No namespace</td></tr>
<tr><td>Annotations</td><td>Signed 8 bit integer.</td></tr>
<tr><td>Diagram</td><td>
int8_t_define
xs:int
Signed 8 bit integer.
Built-in derived type. The int datatype is derived from long by setting the value of maxInclusive to be 2147483647 and...

</td></tr>
<tr><td>Type</td><td>restriction of xs:int</td></tr>
<tr><td>Facets</td><td>maxInclusive 127
minInclusive -128</td></tr>
<tr><td>Used by</td><td>Simple Type I8_define</td></tr>
<tr><td>Source</td><td><xs:simpleType name="int8_t_define">
  <xs:annotation>
    <xs:documentation>Signed 8 bit integer.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:int">
    <xs:minInclusive value="-128"/>
    <xs:maxInclusive value="127"/>
  </xs:restriction>
</xs:simpleType></td></tr>
</table>

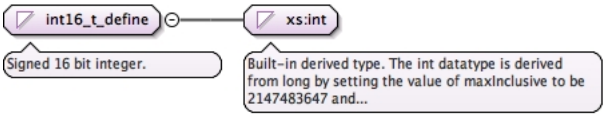
Simple Type uint8_t_define

<table>
<tr><td>Namespace</td><td>No namespace</td></tr>
<tr><td>Annotations</td><td>Unsigned 8 bit integer</td></tr>
<tr><td>Diagram</td><td>
uint8_t_define
xs:unsignedByte
Unsigned 8 bit integer
Built-in derived type. The unsignedByte datatype is derived from unsignedShort by setting the value of maxInclusive to...

</td></tr>
<tr><td>Type</td><td>restriction of xs:unsignedByte</td></tr>
<tr><td>Facets</td><td>maxInclusive 255
minInclusive 0</td></tr>
<tr><td>Used by</td><td>Simple Type U8_define</td></tr>
<tr><td>Source</td><td><xs:simpleType name="uint8_t_define">
  <xs:annotation>
    <xs:documentation>Unsigned 8 bit integer</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:unsignedByte">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="255"/>
  </xs:restriction>
</xs:simpleType></td></tr>
</table>

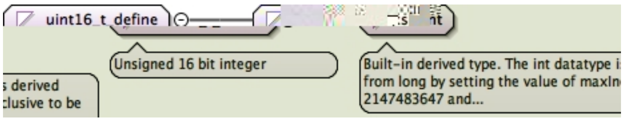
Simple Type int16_t_define

<table>
<tr><td>Namespace</td><td>No namespace</td></tr>
</table>

Annotations	`Signed 16 bit integer.`
Diagram	 int16_t_define xs:int Signed 16 bit integer. Built-in derived type. The int datatype is derived from long by setting the value of maxInclusive to be 2147483647 and...
Type	restriction of xs:int
Facets	maxInclusive `32767` minInclusive `-32768`
Used by	Simple Type I16_define

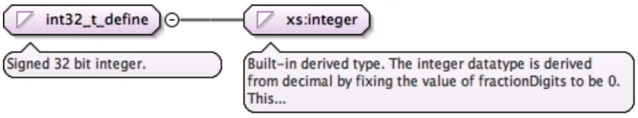
Source	`<xs:simpleType name="int16_t_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Signed 16 bit integer.</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="xs:int">`
	`    <xs:minInclusive value="-32768"/>`
	`    <xs:maxInclusive value="32767"/>`
	`  </xs:restriction>`
	`</xs:simpleType>`

Simple Type `uint16_t_define`

Namespace	No namespace
Annotations	`Unsigned 16 bit integer`
Diagram	 uint16_t_define Unsigned 16 bit integer Built-in derived type. The int datatype i from long by setting the value of maxIn 2147483647 and... s derived clusive to be
Type	restriction of xs:int
Facets	maxInclusive `65535` minInclusive `0`
Used by	Simple Type U16_define

Source	`<xs:simpleType name="uint16_t_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Unsigned 16 bit integer</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="xs:int">`
	`    <xs:minInclusive value="0"/>`
	`    <xs:maxInclusive value="65535"/>`
	`  </xs:restriction>`
	`</xs:simpleType>`

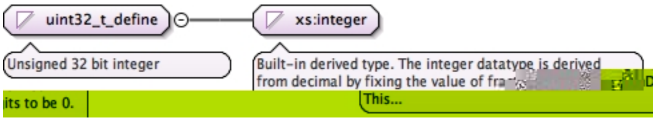
Simple Type `int32_t_define`

Namespace	No namespace
Annotations	`Signed 32 bit integer.`
Diagram	 int32_t_define xs:integer Signed 32 bit integer. Built-in derived type. The integer datatype is derived from decimal by fixing the value of fractionDigits to be 0. This...
Type	restriction of xs:integer
Facets	maxInclusive `2147483647` minInclusive `-2147483648`
Used by	Simple Types I32_define, NATIVE_INT_TYPE_define

Source	`<xs:simpleType name="int32_t_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Signed 32 bit integer.</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="xs:integer">`
	`    <xs:minInclusive value="-2147483648"/>`
	`    <xs:maxInclusive value="2147483647"/>`

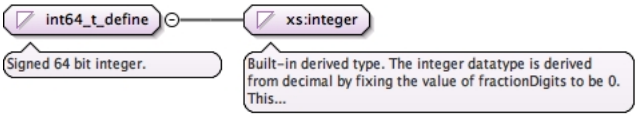
	</xs:restriction> </xs:simpleType>

Simple Type `uint32_t_define`

Namespace	No namespace
Annotations	Unsigned 32 bit integer
Diagram	 Unsigned 32 bit integer Built-in derived type. The integer datatype is derived from decimal by fixing the value of fra its to be 0. This...
Type	restriction of xs:integer
Facets	maxInclusive 4294967295 minInclusive 0
Used by	Simple Types NATIVE_UINT_TYPE_define, U32_define

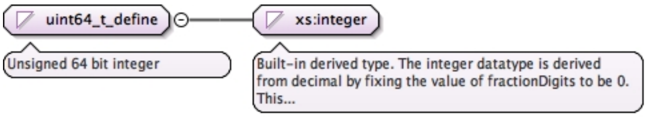
Source	<xs:simpleType name="uint32_t_define">
	  <xs:annotation>
	    <xs:documentation>Unsigned 32 bit integer</xs:documentation>
	  </xs:annotation>
	  <xs:restriction base="xs:integer">
	    <xs:minInclusive value="0"/>
	    <xs:maxInclusive value="4294967295"/>
	  </xs:restriction>
	</xs:simpleType>

Simple Type `int64_t_define`

Namespace	No namespace
Annotations	Signed 64 bit integer.
Diagram	 Signed 64 bit integer. Built-in derived type. The integer datatype is derived from decimal by fixing the value of fractionDigits to be 0. This...
Type	restriction of xs:integer
Facets	maxInclusive 9223372036854775807 minInclusive -9223372036854775808
Used by	Simple Type I64_define

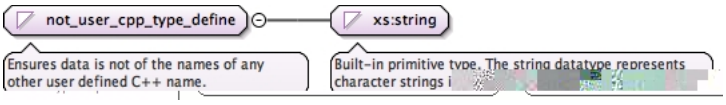
Source	<xs:simpleType name="int64_t_define">
	  <xs:annotation>
	    <xs:documentation>Signed 64 bit integer.</xs:documentation>
	  </xs:annotation>
	  <xs:restriction base="xs:integer">
	    <xs:minInclusive value="-9223372036854775808"/>
	    <xs:maxInclusive value="9223372036854775807"/>
	  </xs:restriction>
	</xs:simpleType>

Simple Type `uint64_t_define`

Namespace	No namespace
Annotations	Unsigned 64 bit integer
Diagram	 Unsigned 64 bit integer Built-in derived type. The integer datatype is derived from decimal by fixing the value of fractionDigits to be 0. This...
Type	restriction of xs:integer
Facets	maxInclusive 18446744073709551615 minInclusive 0
Used by	Simple Type U64_define

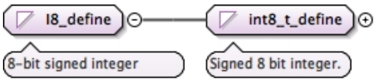
Source	`<xs:simpleType name="uint64_t_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Unsigned 64 bit integer</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="xs:integer">`
	`    <xs:minInclusive value="0"/>`
	`    <xs:maxInclusive value="18446744073709551615"/>`
	`  </xs:restriction>`
	`</xs:simpleType>`

Simple Type `not_user_cpp_type_define`

Namespace	No namespace
Annotations	Ensures data is not of the names of any other user defined C++ name.
Diagram	 not_user_cpp_type_define xs:string Ensures data is not of the names of any other user defined C++ name. Built-in primitive type. The string datatype represents character strings i
Type	xs:string
Source	

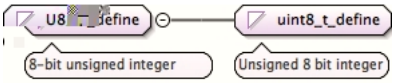
Source	`<xs:simpleType name="NATIVE_UINT_TYPE_define">`
	`  <xs:annotation>`
	`    <xs:documentation>native unsigned integer type declaration</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="uint32_t_define"/>`
	`</xs:simpleType>`

Simple Type `I8_define`

Namespace	No namespace
Annotations	8-bit signed integer
Diagram	 I8_define int8_t_define 8-bit signed integer Signed 8 bit integer.
Type	int8_t_define
Type hierarchy	- xs:int - int8_t_define - I8_define
Facets	maxInclusive 127 minInclusive -128

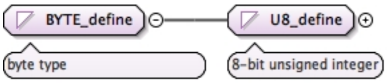
Source	`<xs:simpleType name="I8_define">`
	`  <xs:annotation>`
	`    <xs:documentation>8-bit signed integer</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="int8_t_define"/>`
	`</xs:simpleType>`

Simple Type `U8_define`

Namespace	No namespace
Annotations	8-bit unsigned integer
Diagram	 U8_define uint8_t_define 8-bit unsigned integer Unsigned 8 bit integer
Type	uint8_t_define
Type hierarchy	- xs:unsignedByte - uint8_t_define - U8_define
Facets	maxInclusive 255 minInclusive 0
Used by	Simple Type BYTE_define

Source	`<xs:simpleType name="U8_define">`
	`  <xs:annotation>`
	`    <xs:documentation>8-bit unsigned integer</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="uint8_t_define"/>`
	`</xs:simpleType>`

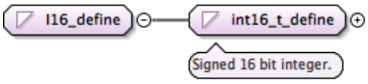
Simple Type `BYTE_define`

Namespace	No namespace
Annotations	byte type
Diagram	 BYTE_define U8_define byte type 8-bit unsigned integer
Type	U8_define
Type hierarchy	- xs:unsignedByte

	• uint8_t_define • U8_define • BYTE_define
Facets	maxInclusive 255 minInclusive 0

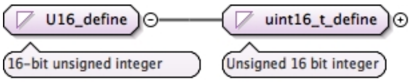
Source	`<xs:simpleType name="BYTE_define">`
	`  <xs:annotation>`
	`    <xs:documentation>byte type</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="U8_define"/>`
	`</xs:simpleType>`

Simple Type `I16_define`

Namespace	No namespace
Diagram	
Type	int16_t_define
Type hierarchy	• xs:int • int16_t_define • I16_define
Facets	maxInclusive 32767 minInclusive -32768

Source	`<xs:simpleType name="I16_define">`
	`  <xs:restriction base="int16_t_define"/>`
	`</xs:simpleType>`

Simple Type `U16_define`

Namespace	No namespace
Annotations	`16-bit unsigned integer`
Diagram	 U16_define uint16_t_define 16-bit unsigned integer Unsigned 16 bit integer
Type	uint16_t_define
Type hierarchy	• xs:int • uint16_t_define • U16_define
Facets	maxInclusive 65535 minInclusive 0

Source	`<xs:simpleType name="U16_define">`
	`  <xs:annotation>`
	`    <xs:documentation>16-bit unsigned integer</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="uint16_t_define"/>`
	`</xs:simpleType>`

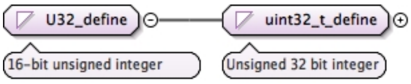
Simple Type `I32_define`

Namespace	No namespace
Annotations	`32-bit signed integer`
Diagram	I32_define int: t_define 32-bit signed integer Signed 32

Type	int32_t_define
Type hierarchy	• xs:integer • int32_t_define • I32_define
Facets	maxInclusive 2147483647 minInclusive -2147483648

Source	`<xs:simpleType name="I32_define">`
	`  <xs:annotation>`
	`    <xs:documentation>32-bit signed integer</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="int32_t_define"/>`
	`</xs:simpleType>`

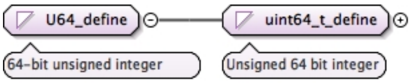
Simple Type `U32_define`

Namespace	No namespace
Annotations	16-bit unsigned integer
Diagram	 U32_define uint32_t_define 16-bit unsigned integer Unsigned 32 bit integer
Type	uint32_t_define
Type hierarchy	• xs:integer • uint32_t_define • U32_define
Facets	maxInclusive 4294967295 minInclusive 0

Source	`<xs:simpleType name="U32_define">`
	`  <xs:annotation>`
	`    <xs:documentation>16-bit unsigned integer</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="uint32_t_define"/>`
	`</xs:simpleType>`

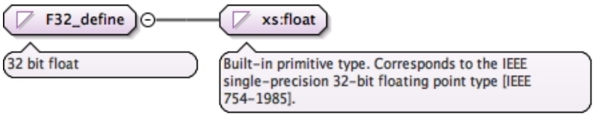
Simple Type `I64_define`

Namespace	No namespace
Annotations	64-bit unsigned integer
Diagram	

Diagram	 U64_define uint64_t_define 64-bit unsigned integer Unsigned 64 bit integer	
Type	uint64_t_define	
Type hierarchy	• xs:integer • uint64_t_define • U64_define	
Facets	maxInclusive	18446744073709551615
	minInclusive	0

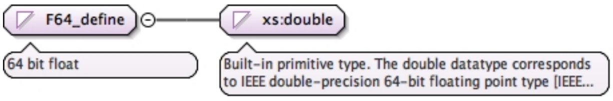
Source	`<xs:simpleType name="U64_define">`
	`  <xs:annotation>`
	`    <xs:documentation>64-bit unsigned integer</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="uint64_t_define"/>`
	`</xs:simpleType>`	

Simple Type `F32_define`

Namespace	No namespace
Annotations	32 bit float
Diagram	 F32_define xs:float 32 bit float Built-in primitive type. Corresponds to the IEEE single-precision 32-bit floating point type [IEEE 754-1985].
Type	xs:float

Source	`<xs:simpleType name="F32_define">`
	`  <xs:annotation>`
	`    <xs:documentation>32 bit float</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="xs:float"/>`
	`</xs:simpleType>`

Simple Type `F64_define`

Namespace	No namespace
Annotations	64 bit float
Diagram	 F64_define xs:double 64 bit float Built-in primitive type. The double datatype corresponds to IEEE double-precision 64-bit floating point type [IEEE...
Type	xs:double

Source	`<xs:simpleType name="F64_define">`
	`  <xs:annotation>`
	`    <xs:documentation>64 bit float</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:restriction base="xs:double"/>`
	`</xs:simpleType>`

Attribute(s)

Attribute `port / @name`

Namespace	No namespace	
Annotations	Name of the port.	
Properties	use:	required
Used by	Element	port

Source	`<xs:attribute name="name" use="required">`
	`  <xs:annotation>`
	`    <xs:documentation>Name of the port.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`	

Attribute port / @data_type

Namespace	No namespace	
Annotations	Type of data that is being accessed/sent from the port.	
Properties	use:	required
Used by	Element	port

Source	`<xs:attribute name="data_type" use="required">`
	`  <xs:annotation>`
	`    <xs:documentation>Type of data that is being accessed/sent from the port.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`	

Attribute port / @kind

Namespace	No namespace	
Annotations	Defines if port is an input or an output port.	
Type	port_types_define	
Properties	use:	required
Facets	enumeration	input
	enumeration	sync_input
	enumeration	guarded_input
	enumeration	async_input
	enumeration	model_input
	enumeration	output
Used by	Element	port

Source	`<xs:attribute name="kind" use="required" type="port_types_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Defines if port is an input or an output port.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`	

Attribute port / @max_number

Namespace	No namespace	
Annotations	Defines how many connections can be established to this port.	
Used by	Element	port

Source	`<xs:attribute name="max_number">`
	`  <xs:annotation>`
	`    <xs:documentation>Defines how many connections can be established to this port.</`
	`xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`	

Attribute port / @role

Namespace	No namespace	
Annotations	Specifies what role this port plays or what this port is connected to.	
Type	component_role_define	
Properties	content:	simple
Facets	enumeration	LogEvent
	enumeration	LogTextEvent
	enumeration	TimeGet
	enumeration	ParamSet
	enumeration	ParamGet
	enumeration	Telemetry
	enumeration	CmdRegistration

	enumeration Cmd enumeration CmdResponse
Used by	Element port

Source	`<xs:attribute name="role" type="component_role_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Specifies what role this port plays or what this port is connected to.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute port / @priority

Namespace	No namespace
Annotations	Priority of port.
Type	xs:integer
Properties	content: simple
Used by	Element port

Source	`<xs:attribute name="priority" type="xs:integer">`
	`  <xs:annotation>`
	`    <xs:documentation>Priority of port.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute port / @full

Namespace	No namespace
Annotations	Describes what to do with incoming items if full.
Used by	Element port

Source	`<xs:attribute name="full">`
	`  <xs:annotation>`
	`    <xs:documentation>Describes what to do with incoming items if full.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute item / @name

Namespace	No namespace
Annotations	Name of the enum item.
Properties	use: required
Used by	Element item

Source	`<xs:attribute name="name" use="required">`
	`  <xs:annotation>`
	`    <xs:documentation>Name of the enum item.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute item / @value

Namespace	No namespace
Annotations	The value being sent through the enum item.
Used by	Element item

Source	`<xs:attribute name="value">`
	`  <xs:annotation>`
	`    <xs:documentation>The value being sent through the enum item.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute item / @comment

Namespace	No namespace

Annotations	Comment about the enum item.	
Used by	Element	item
Source	<xs:attribute name="comment"> <xs:annotation> <xs:documentation>Comment about the enum item.</xs:documentation> </xs:annotation> </xs:attribute>	

Attribute enum / @name

Namespace	No namespace	
Annotations	Enum Name.	
Properties	use:	required
Used by	Element	enum

Source	<xs:attribute name="name" use="required">
	  <xs:annotation>
	    <xs:documentation>Enum Name.</xs:documentation>
	  </xs:annotation>
	</xs:attribute>	

Attribute channel / @id

Namespace	No namespace	
Annotations	ID for the telemetry channel.	
Type	id_define	
Properties	use:	required
Facets	pattern	((0?x\d+)|\d+)
Used by	Element	channel
Source	<xs:attribute name="id" use="required" type="id_define"> <xs:annotation> <xs:documentation>ID for the telemetry channel.</xs:documentation> </xs:annotation> </xs:attribute>	

Attribute channel / @name

Namespace	No namespace	
Annotations	Name of the telemetry channel.	
Properties	use:	required
Used by	Element	channel
Source	<xs:attribute name="name" use="required"> <xs:annotation> <xs:documentation>Name of the telemetry channel.</xs:documentation> </xs:annotation> </xs:attribute>	

Attribute channel / @update

Namespace	No namespace	
Annotations	Defines when the channel updates.	
Type	channel_update_define	
Properties	content:	simple
Facets	enumeration	always
	enumeration	on_change
Used by	Element	channel
Source		

	`</xs:annotation>`
	`</xs:attribute>`

Attribute channel / @abbrev

Namespace	No namespace
Annotations	Required for AMPCS dictionary generation.
Used by	Element channel

Source	`<xs:attribute name="abbrev">`
	  `<xs:annotation>`
	    `<xs:documentation>Required for AMPCS dictionary generation.</xs:documentation>`
	  `</xs:annotation>`
	`</xs:attribute>`

Attribute channel / @format_string

Namespace	No namespace
Annotations	Used to format data into readable content on the ground system software.
Used by	Element channel

Source	`<xs:attribute name="format_string">`
	  `<xs:annotation>`
	    `<xs:documentation>Used to format data into readable content on the ground system software.</`
	`xs:documentation>`
	  `</xs:annotation>`
	`</xs:attribute>`

Attribute channel / @high_yellow

Namespace	No namespace
Type	xs:decimal
Properties	content: simple
Used by	Element channel
Source	`<xs:attribute name="high_yellow" type="xs:decimal"/>`

Attribute channel / @high_red

Namespace	No namespace
Type	xs:decimal
Properties	content: simple
Used by	Element channel
Source	`<xs:attribute name="high_red" type="xs:decimal"/>`

Attribute channel / @high_orange

Namespace	No namespace
Type	xs:decimal
Properties	content: simple
Used by	Element channel
Source	`<xs:attribute name="high_orange" type="xs:decimal"/>`

Attribute channel / @low_yellow

Namespace	No namespace
Type	xs:decimal
Properties	content: simple
Used by	Element channel

Source	`<xs:attribute name="low_yellow" type="xs:decimal"/>`

Attribute channel / @low_red

Namespace	No namespace	
Type	xs:decimal	
Properties	content:	simple
Used by	Element	channel
Source	`<xs:attribute name="low_red" type="xs:decimal"/>`	

Attribute channel / @low_orange

Namespace	No namespace	
Type	xs:decimal	
Properties	content:	simple
Used by		

Attribute type_size_choice_define / @size

Namespace	No namespace
Annotations	The size of the argument.
Type	xs:nonNegativeInteger
Properties	content: simple
Used by	Attribute Group type_size_choice_define

Source	`<xs:attribute name="size" type="xs:nonNegativeInteger">`
	`  <xs:annotation>`
	`    <xs:documentation>The size of the argument.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute telemetry / @telemetry_base

Namespace	No namespace
Annotations	Base at which IDs start from.
Type	base_code_define
Properties	content: simple
Used by	Element telemetry

	`<xs:documentation>Name of the event.</xs:documentation>`
	`</xs:annotation>`
	`</xs:attribute>`

Attribute event / @id

Namespace	No namespace
Annotations	Event ID.
Type	id_define
Properties	use: required
Facets	pattern ((0?x\d+)|\d+)
Used by	Element event

Source	`<xs:attribute name="id" use="required" type="id_define">`
	`<xs:annotation>`
	`<xs:documentation>Event ID.</xs:documentation>`
	`</xs:annotation>`
	`</xs:attribute>`

Attribute event / @severity

Namespace	No namespace
Annotations	Severity of event.
Type	severity_define
Properties	use: required
Facets	enumeration COMMAND enumeration ACTIVITY_LO enumeration ACTIVITY_HI enumeration WARNING_LO enumeration WARNING_HI enumeration DIAGNOSTIC enumeration FATAL
Used by	Element event

Source	`<xs:attribute name="severity" use="required" type="severity_define">`
	`<xs:annotation>`
	`<xs:documentation>Severity of event.</xs:documentation>`
	`</xs:annotation>`
	`</xs:attribute>`

Attribute event / @format_string

Namespace	No namespace
Annotations	Output string that will be shown in ground system software.
Properties	use: required
Used by	Element event

Source	`<xs:attribute name="format_string" use="required">`
	`<xs:annotation>`
	`<xs:documentation>Output string that will be shown in ground system software.</xs:documentation>`
	`</xs:annotation>`
	`</xs:attribute>`

Attribute event / @throttle

Namespace	No namespace
Annotations	Determines how many of the events are generated before the program stops them.
Type	xs:nonNegativeInteger
Properties	content: simple
Used by	Element event

Source	`<xs:attribute name="throttle" type="xs:nonNegativeInteger">`
	`  <xs:annotation>`
	`    <xs:documentation>Determines how many of the events are generated before the program stops`
	` them.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute events / @event_base

Namespace	No namespace
Annotations	Base at which ids start from.
Type	base_code_define
Properties	content: simple
Used by	Element events

Source	`<xs:attribute name="event_base" type="base_code_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Base at which ids start from.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute command / @kind

Namespace	No namespace
Annotations	Command kind.
Type	command_kind_define
Properties	use: required
Facets	enumeration async enumeration sync enumeration guarded
Used by	Element command

Source	`<xs:attribute name="kind" use="required" type="command_kind_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Command kind.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute command / @opcode

Namespace	No namespace
Annotations	Command opcode.
Type	id_define
Properties	use: required
Facets	pattern ((0?x\d+)|\d+)

	</xs:annotation> </xs:attribute>

Attribute command / @priority

Namespace	No namespace
Annotations	Priority of the command.
Type	xs:integer
Properties	content: simple
Used by	Element command
Source	<xs:attribute name="priority" type="xs:integer"> <xs:annotation> <xs:documentation>Priority of the command.</xs:documentation> </xs:annotation> </xs:attribute>

Attribute command / @full

Namespace	No namespace
Annotations	Describes what to do with incoming items if full.
Type	full_items_define
Properties	content: simple
Facets	enumeration drop enumeration assert enumeration block
Used by	Element command
Source	<xs:attribute name="full" type="full_items_define"> <xs:annotation> <xs:documentation>Describes what to do with incoming items if full.</xs:documentation> </xs:annotation> </xs:attribute>

Attribute commands / @opcode_base

Namespace	No namespace
Annotations	Base at which the opcodes start from.
Type	base_code_define
Properties	content: simple
Used by	Element commands
Source	<xs:attribute name="opcode_base" type="base_code_define"> <xs:annotation> <xs:documentation>Base at which the opcodes start from.</xs:documentation> </xs:annotation> </xs:attribute>

Attribute arg_define / arg / @name

Namespace	No namespace
Annotations	Name of the argument.
Properties	use: required
Used by	Element arg_define/arg
Source	<xs:attribute name="name" use="required"> <xs:annotation> <xs:documentation>Name of the argument.</xs:documentation> </xs:annotation> </xs:attribute>

Attribute arg_define / arg / @pass_by

Namespace	No namespace

Annotations	Defines how the arguments are passed.
Type	pass_by_define
Properties	content: simple
Facets	enumeration reference enumeration value enumeration pointer
Used by	Element arg_define/arg

Source	`<xs:attribute name="pass_by" type="pass_by_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Defines how the arguments are passed.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute arg_define / arg / @comment

Namespace	No namespace
Annotations	Comments about the argument.
Type	xs:string
Properties	content: simple
Used by	Element arg_define/arg

Source	`<xs:attribute name="comment" type="xs:string">`
	`  <xs:annotation>`
	`    <xs:documentation>Comments about the argument.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute interface_define / @name

Namespace	No namespace
Annotations	Interface name.
Properties	use: required
Used by	Complex Type interface_define

Source	`<xs:attribute name="name" use="required">`
	`  <xs:annotation>`
	`    <xs:documentation>Interface name.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute interface_define / @priority

Namespace	No namespace
Type	xs:integer
Properties	content: simple
Used by	Complex Type interface_define
Source	`<xs:attribute name="priority" type="xs:integer"/>`

Attribute data_type_and_default_define / @data_type

Namespace	No namespace
Type	union of(not_user_cpp_type_define, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token, restriction of xs:token)
Properties	use: required
Used by	Complex Type data_type_and_default_define

Source	`<xs:attribute name="data_type" use="required">`
	`  <xs:simpleType>`

```
    <xs:union memberTypes="not_user_cpp_type_define">
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="ENUM"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="string"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="I8"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="U8"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="I16"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="U16"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="I32"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="U32"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="I64"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="U64"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="F32"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="F64"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="NATIVE_INT_TYPE"/>
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="NATIVE_UINT_TYPE"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:union>
  </xs:simpleType>
</xs:attribute>
```

Attribute `data_type_and_default_define / @default`

Namespace	No namespace
Used by	Complex Type data_type_and_default_define
Source	`<xs:attribute name="default"/>`

Attribute data_type_and_default_define / @size

Namespace	No namespace
Type	positive_integer_define
Properties	content: simple
Facets	minInclusive 1
Used by	Complex Type data_type_and_default_define
Source	`<xs:attribute name="size" type="positive_integer_define"/>`

Attribute parameter / @id

Namespace	No namespace
Annotations	ID of the attribute.
Type	id_define
Properties	use: required
Facets	pattern ((0?x\d+)|\d+)
Used by	Element parameter

Source	`<xs:attribute name="id" use="required" type="id_define">`
	`  <xs:annotation>`
	`    <xs:documentation>ID of the attribute.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute parameter / @set_opcode

Namespace	No namespace
Annotations	Opcode for setting the parameter.
Type	id_define
Properties	use: required
Facets	pattern ((0?x\d+)|\d+)
Used by	Element parameter

Source	`<xs:attribute name="set_opcode" use="required" type="id_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Opcode for setting the parameter.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute parameter / @save_opcode

Namespace	No namespace
Annotations	Opcode for saving the parameter.
Type	id_define
Properties	use: required
Facets	pattern ((0?x\d+)|\d+)
Used by	Element parameter

Source	`<xs:attribute name="save_opcode" use="required" type="id_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Opcode for saving the parameter.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute parameter / @name

Namespace	No namespace
Annotations	Parameter name

Properties	use:	required
Used by	Element	parameter

Source	<xs:attribute name="name" use="required">
	  <xs:annotation>
	    <xs:documentation>Parameter name</xs:documentation>
	  </xs:annotation>
	</xs:attribute>	

Attribute parameters / @parameter_base

Namespace	No namespace	
Annotations		
Type	base_code_define	
Properties	content:	simple
Used by	Element	parameters

Source	<xs:attribute name="parameter_base" type="base_code_define">
	  <xs:annotation>
	    <xs:documentation/>
	  </xs:annotation>
	</xs:attribute>	

Attribute parameters / @opcode_base

Namespace	No namespace	
Annotations		
Type	base_code_define	
Properties	content:	simple
Used by	Element	parameters

Source	<xs:attribute name="opcode_base" type="base_code_define">
	  <xs:annotation>
	    <xs:documentation/>
	  </xs:annotation>
	</xs:attribute>	

Attribute component_define / @name

Namespace	No namespace	
Properties	use:	required
Used by	Complex Type	component_define
Source	<xs:attribute name="name" use="required"/>	

Attribute component_define / @kind

Namespace	No namespace	
Annotations	Choice between active, passive, and queued.	
Type	component_types_define	
Properties	use:	required
Facets	enumeration	active
	enumeration	passive
	enumeration	queued
Used by	Complex Type	component_define
Source	<xs:attribute name="kind" use="required" type="component_types_define"> <xs:annotation> <xs:documentation>Choice between active, passive, and queued.</xs:documentation> </xs:annotation> </xs:attribute>	

Attribute component_define / @namespace

Namespace	No namespace
Annotations	The namespace in which the component is located in.
Used by	Complex Type component_define

Source	`<xs:attribute name="namespace">`
	`  <xs:annotation>`
	`    <xs:documentation>The namespace in which the component is located in.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute component_define / @modeler

Namespace	No namespace
Type	xs:boolean
Properties	content: simple
Used by	Complex Type component_define
Source	`<xs:attribute name="modeler" type="xs:boolean"/>`

Attribute return / @name

Namespace	No namespace
Annotations	Name of the argument.
Used by	Element return

Source	`<xs:attribute name="name">`
	`  <xs:annotation>`
	`    <xs:documentation>Name of the argument.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

Attribute return / @pass_by

Namespace	No namespace
Annotations	Defines how the arguments are passed.
Type	pass_by_define
Properties	content: simple
Facets	enumeration reference enumeration value enumeration pointer
Used by	Element return

Source	`<xs:attribute name="pass_by" type="pass_by_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Defines how the arguments are passed.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

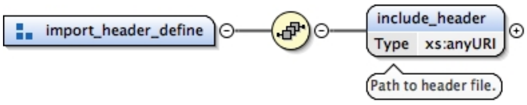
Attribute return / @comment

Namespace	No namespace
Annotations	Comments about the argument.
Type	xs:string
Properties	content: simple
Used by	Element return

Source	`<xs:attribute name="comment" type="xs:string">`
	`  <xs:annotation>`
	`    <xs:documentation>Comments about the argument.</xs:documentation>`
	`  </xs:annotation>`
	`</xs:attribute>`

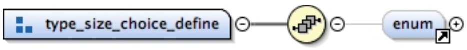
Element Group(s)

Element Group import_header_define

Namespace	No namespace
Diagram	 import_header_define include_header Type xs:anyURI Path to header file.
Used by	Complex Type component_define
Model	include_header
Children	include_header

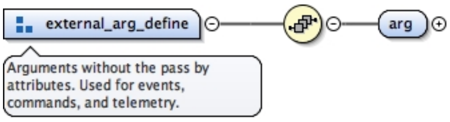
Source	<xs:group name="import_header_define">
	  <xs:sequence>
	    <xs:element name="include_header" type="xs:anyURI">
	      <xs:annotation>
	        <xs:documentation>Path to header file.</xs:documentation>
	      </xs:annotation>
	    </xs:element>
	  </xs:sequence>
	</xs:group>

Element Group type_size_choice_define

Namespace	No namespace
Diagram	
Used by	Elements arg_define/arg, channel, external_arg_define/arg, return
Model	enum{0,1}
Children	enum

Source	<xs:group name="type_size_choice_define">
	  <xs:sequence>
	    <xs:element minOccurs="0" ref="enum"/>
	  </xs:sequence>
	</xs:group>

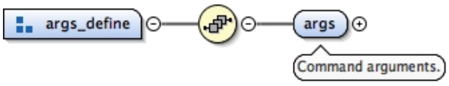
Element Group external_arg_define

Namespace	No namespace
Annotations	Arguments without the pass by attributes. Used for events, commands, and telemetry.
Diagram	 external_arg_define arg Arguments without the pass by attributes. Used for events, commands, and telemetry.
Used by	Elements args_define/args, event/args
Model	arg
Children	arg

Source	<xs:group name="external_arg_define">
	  <xs:annotation>
	    <xs:documentation>Arguments without the pass by attributes. Used for events, commands, and telemetry.</xs:documentation>
	  </xs:annotation>
	  <xs:sequence>
	    <xs:element name="arg">
	      <xs:complexType>
	        <xs:choice minOccurs="0" maxOccurs="unbounded">
	          <xs:group ref="type_size_choice_define"/>
	          <xs:element ref="comment"/>
	        </xs:choice>
	        <xs:attribute name="name" use="required">
	          <xs:annotation>
	            <xs:documentation>Name of the argument.</xs:documentation>
	          </xs:annotation>
	        </xs:attribute>
	        <xs:attribute name="comment" type="xs:string">

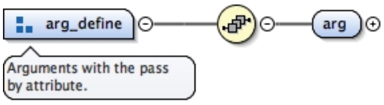
	`          <xs:annotation>`
	`            <xs:documentation>Comments about the argument.</xs:documentation>`
	`          </xs:annotation>`
	`        </xs:attribute>`
	`        <xs:attributeGroup ref="type_size_choice_define"/>`
	`      </xs:complexType>`
	`    </xs:element>`
	`  </xs:sequence>`
	`</xs:group>`

Element Group args_define

Namespace	No namespace
Diagram	
Used by	Element command
Model	args
Children	args

Source	`<xs:group name="args_define">`
	`  <xs:sequence>`
	`    <xs:element name="args">`
	`      <xs:annotation>`
	`        <xs:documentation>Command arguments.</xs:documentation>`
	`      </xs:annotation>`
	`      <xs:complexType>`
	`        <xs:group minOccurs="0" maxOccurs="unbounded" ref="external_arg_define"/>`
	`      </xs:complexType>`
	`    </xs:element>`
	`  </xs:sequence>`
	`</xs:group>`

Element Group arg_define

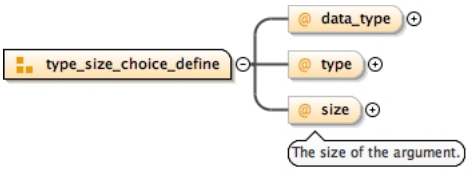
Namespace	No namespace
Annotations	Arguments with the pass by attribute.
Diagram	
Used by	Element interface_define/args
Model	arg
Children	arg

Source	`<xs:group name="arg_define">`
	`  <xs:annotation>`
	`    <xs:documentation>Arguments with the pass by attribute.</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:sequence>`
	`    <xs:element name="arg">`
	`      <xs:complexType>`
	`        <xs:choice minOccurs="0" maxOccurs="unbounded">`
	`          <xs:group ref="type_size_choice_define"/>`
	`          <xs:element ref="comment"/>`
	`        </xs:choice>`
	`        <xs:attribute name="name" use="required">`
	`          <xs:annotation>`
	`            <xs:documentation>Name of the argument.</xs:documentation>`
	`          </xs:annotation>`
	`        </xs:attribute>`
	`        <xs:attribute name="pass_by" type="pass_by_define">`
	`          <xs:annotation>`
	`            <xs:documentation>Defines how the arguments are passed.</xs:documentation>`
	`          </xs:annotation>`
	`        </xs:attribute>`
	`        <xs:attribute name="comment" type="xs:string">`
	`          <xs:annotation>`
	`            <xs:documentation>Comments about the argument.</xs:documentation>`
	`          </xs:annotation>`
	`        </xs:attribute>`
	`        <xs:attributeGroup ref="type_size_choice_define"/>`
	`      </xs:complexType>`

```
    </xs:element>
  </xs:sequence>
</xs:group>
```

Attribute Group(s)

Attribute Group `type_size_choice_define`

Namespace	No namespace
Diagram	 type_size_choice_define data_type type size The size of the argument.
Used by	Elements arg_define/arg, channel, external_arg_define/arg, return

Attributes

QName	Type	Use	
data_type	union of(xs:string, restriction of xs:token, restriction of xs:token)	optional	
size	xs:nonNegativeInteger	optional	
	`The size of the argument.`		
type	union of(xs:string, restriction of xs:token, restriction of xs:token)	optional	

Source

```
<xs:attributeGroup name="type_size_choice_define">
  <xs:attribute name="data_type">
    <xs:simpleType>
      <xs:union memberTypes="xs:string">
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="string"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="ENUM"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:union>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="type">
    <xs:simpleType>
      <xs:union memberTypes="xs:string">
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="string"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:token">
            <xs:enumeration value="ENUM"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:union>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="size" type="xs:nonNegativeInteger">
    <xs:annotation>
      <xs:documentation>The size of the argument.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:attributeGroup>
```