Schema documentation for xml-actions-1.3.xsd

20 october 2013

Table of Contents

Namespace: ""

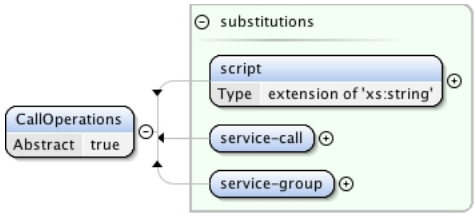
Schemas

Main schema xml-actions-1.3.xsd

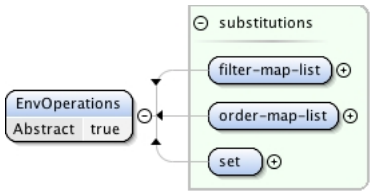
Namespace	No namespace

Elements

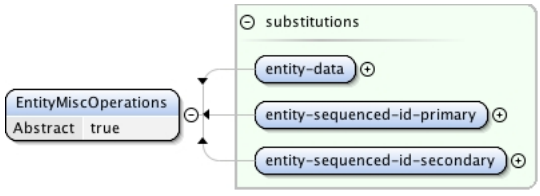
Element CallOperations

Namespace	No namespace
Diagram	
Used by	Element Group AllOperations

Element EnvOperations

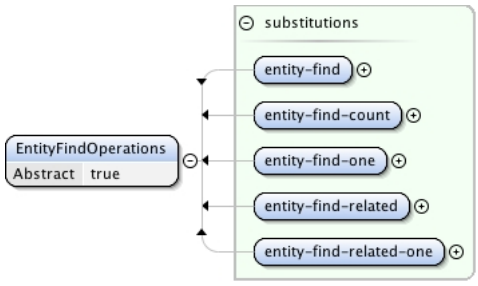
Namespace	No namespace
Diagram	
Used by	Element Group AllOperations

Element EntityMiscOperations

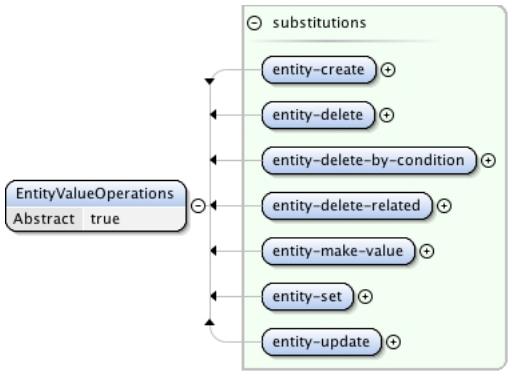
Namespace	No namespace
Diagram	
Used by	Element Group AllOperations

Element EntityFindOperations

Namespace	No namespace

Diagram	
Used by	Element Group AllOperations

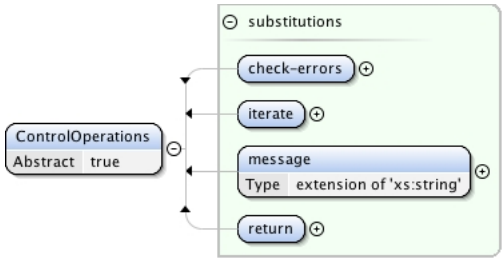
Element `EntityValueOperations`

Namespace	No namespace
Diagram	
Used by	Element Group AllOperations

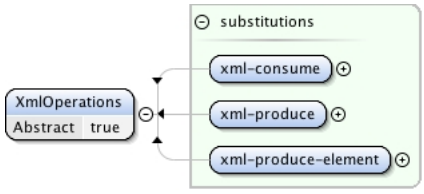
Element `EntityListOperations`

Namespace	No namespace
Diagram	
Used by	Element Group AllOperations

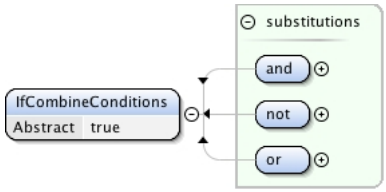
Element `ControlOperations`

Namespace	No namespace
Diagram	
Used by	Element Group AllOperations

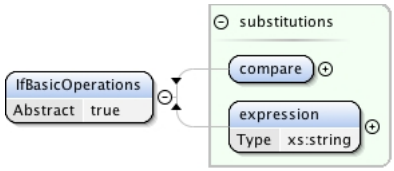
Element `XmlOperations`

Namespace	No namespace
Diagram	

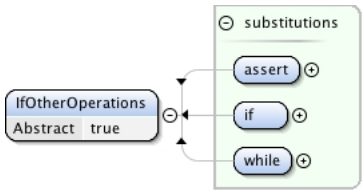
Element `IfCombineConditions`

Namespace	No namespace
Diagram	
Used by	Element Group IfConditions

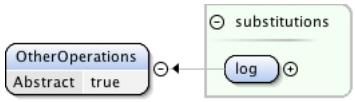
Element `IfBasicOperations`

Namespace	No namespace
Diagram	
Used by	Element Groups AllOperations, IfConditions

Element `IfOtherOperations`

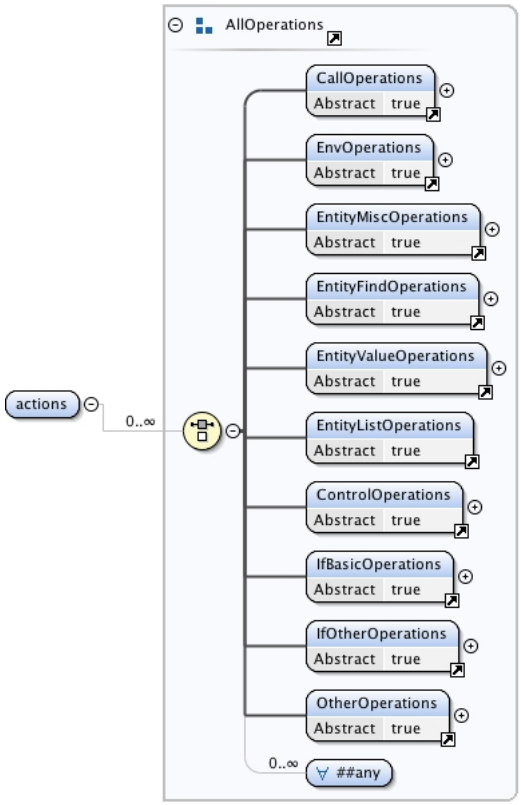
Namespace	No namespace
Diagram	
Used by	Element Group AllOperations

Element `OtherOperations`

Namespace	No namespace
Diagram	
Used by	Element Group AllOperations

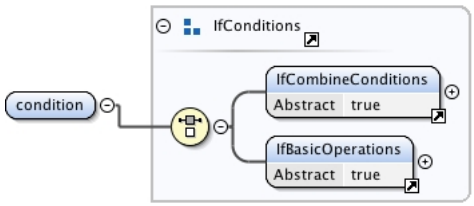
Element `actions`

Namespace	No namespace
Annotations	`XML Actions can be embedded in various files, or put in a file of their own and run like a script. Like a script the parameters passed into the XML Actions will already be defined in the context.`

Diagram	

Instance	`<actions>`
	`  <CallOperations>{1,1}</CallOperations>`
	`  <EnvOperations>{1,1}</EnvOperations>`
	`  <EntityMiscOperations>{1,1}</EntityMiscOperations>`
	`  <EntityFindOperations>{1,1}</EntityFindOperations>`
	`  <EntityValueOperations>{1,1}</EntityValueOperations>`
	`  <EntityListOperations>{1,1}</EntityListOperations>`
	`  <ControlOperations>{1,1}</ControlOperations>`
	`  <IfBasicOperations>{1,1}</IfBasicOperations>`
	`  <IfOtherOperations>{1,1}</IfOtherOperations>`
	`  <OtherOperations>{1,1}</OtherOperations>`
	`</actions>`

Element `condition`

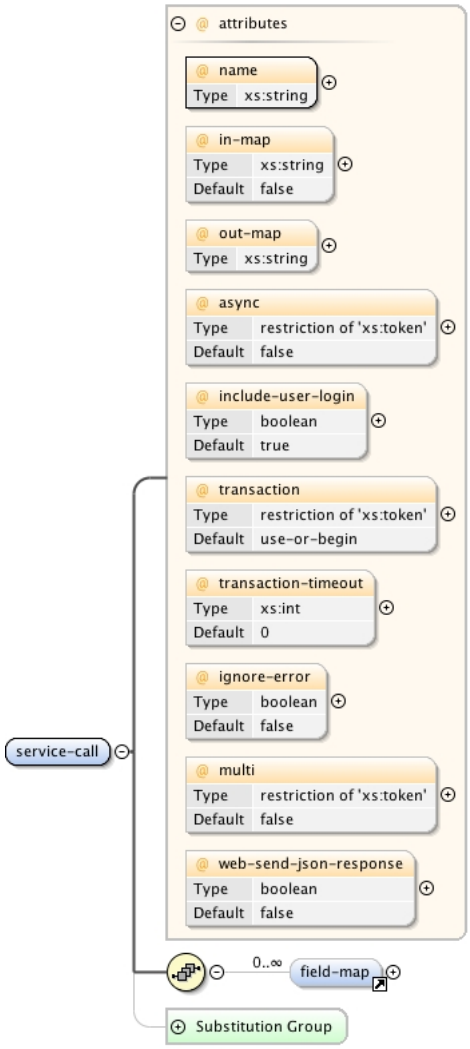
Namespace	No namespace
Annotations	`Contains a single condition of any sort and evaluates to a boolean value. To combine the other if operations the and, or, and xor elements can be used.`
Diagram	
Used by	Elements else-if, if, while

Instance	`<condition>`
	`  <IfCombineConditions>{1,1}</IfCombineConditions>`
	`  <IfBasicOperations>{1,1}</IfBasicOperations>`
	`</condition>`

Element `service-call`

Namespace	No namespace
Annotations	`Call a service.`

Diagram



Instance

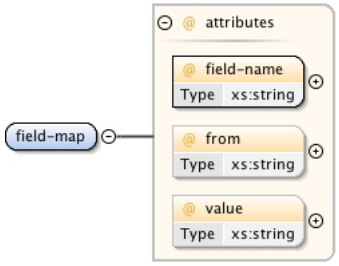
```
<service-call async="false" ignore-error="false" include-user-login="true" in-map="false" multi="false" name="" out-map="" transaction="use-or-begin" transaction-timeout="0" web-send-json-response="false">
  <field-map field-name="" from="" value="">{0,unbounded}</field-map>
</service-call>
```

Attributes

QName	Type	Fixed	Default	Use
async	restriction of xs:token		false	optional
	If true runs the service asynchronously. Use persist to run async through a database record.			
ignore-error	boolean		false	optional
in-map	xs:string		false	optional
	Creates an in parameters with variables matching the names of service in-parameters elements, doing type conversions as needed. If false (default) does nothing. If true constructs an in-map from the context. Otherwise is the name of a Map in the context uses it as the source Map for the service context.			
include-user-login	boolean		true	optional
	Include the current user in the service call. If you don't want to pass that in set to false. Defaults to true.			
multi	restriction of xs:token		false	optional
name	xs:string			required
	The combined service name, like: "${path}.${verb}${noun}". To explicitly separate the verb and noun put a hash (#) between them, like: "${path}.${verb}#${noun}".			

QName	Type	Fixed	Default	Use
out-map	xs:string			optional
	Optional name in the method environment to use for the output (results) map. If empty then the output map will be ignored.			
transaction	restriction of xs:token		use-or-begin	optional
transaction-timeout	xs:int		0	optional
	Defines the timeout for the transaction, in seconds. This value is only used if this service begins a transaction (either require-new, or use-or-begin and there is no other transaction already in place).			
web-send-json-response	boolean		false	optional

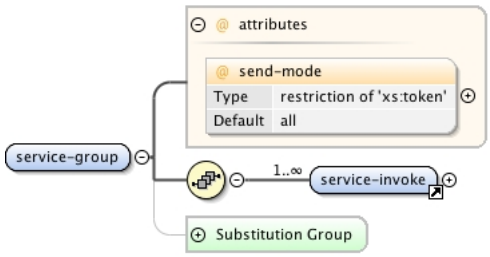
Element `field-map`

Namespace	No namespace
Annotations	A name/value pair. If from and value are empty will look in the context for a field matching the field-name.
Diagram	
Used by	Elements entity-find-one, filter-map-list, service-call

Attributes

QName	Type	Fixed	Default	Use
field-name	xs:string			required
	Name of the entity field.			
from	xs:string			optional
	Name of the field (variable) in the context.			
value	xs:string			optional
	Literal string or use ${} syntax to expand variables.			

Element `service-group`

Namespace	No namespace
Diagram	

Instance

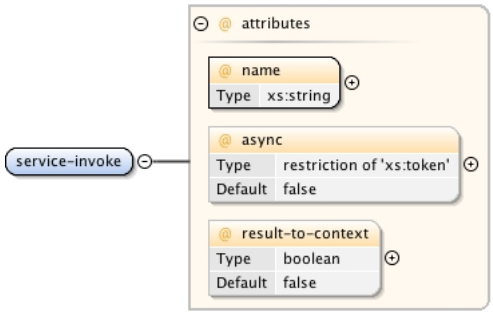
```
<service-group send-mode="all">
  <service-invoke async="false" name="" result-to-context="false">{1,unbounded}</service-invoke>
</service-group>
```

Attributes

QName	Type	Fixed	Default	Use
send-mode	restriction of xs:token		all	optional

Element `service-invoke`

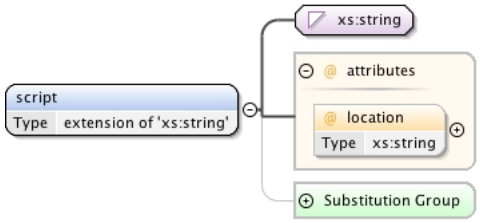
Namespace	No namespace

Diagram	
Used by	Element service-group

Attributes

QName	Type	Fixed	Default	Use
async	restriction of xs:token		false	optional
	`If true runs the service asynchronously. Use persist to run async through a database record.`			
name	xs:string			required
	`The combined service name, like: "${path}.${verb}${noun}". To explicitly separate the verb and noun put a hash (#) between them, like: "${path}.${verb}#${noun}".`			
result-to-context	boolean		false	optional

Element `script`

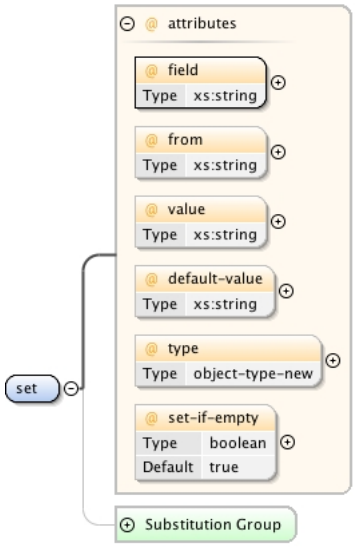
Namespace	No namespace
Annotations	`Runs the script at the specified location. You can also put a Groovy script inline under this element. If a location is specified the file can be a Groovy script or an xml-actions script. The script will run in the same context as the current operation.`
Diagram	
Type	extension of xs:string

Attributes

QName	Type	Fixed	Default	Use
location	xs:string			optional

Element `set`

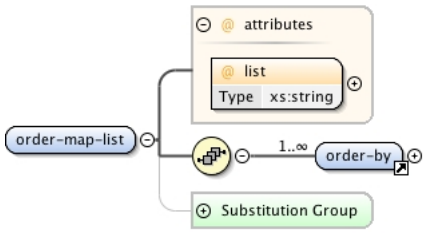
Namespace	No namespace
Annotations	`Set a field from another field (from) or an inline value, or a default-value.`

Diagram	

Attributes

QName	Type	Fixed	Default	Use
default-value	xs:string			optional
	Default value to set in field if an empty String or null value is found.			
field	xs:string			required
	Name of the field to set a value in.			
from	xs:string			optional
	Name of a field to copy from. Can also be an expression that evaluates to something to put into the field.			
set-if-empty	boolean		true	optional
	If an empty String or null value is found, set that in the field. Defaults to true, set to false to do nothing to the field.			
type	object-type-new			optional
	Type to convert to. NewList will create a new List, NewMap will create a new Map.			
value	xs:string			optional
	Inline value to copy in field. May include variables using the ${} syntax.			

Element `order-map-list`

Namespace	No namespace
Annotations	Sort a List of Maps by field names given in order-by sub-element.
Diagram	

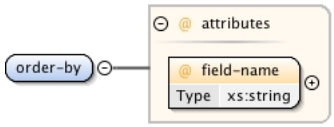
Instance	`<order-map-list list="">`
	`  <order-by field-name="">{1,unbounded}</order-by>`
	`</order-map-list>`

Attributes

QName	Type	Fixed	Default	Use
list	xs:string			required
	Name of the list to be sorted.			

Element `order-by`

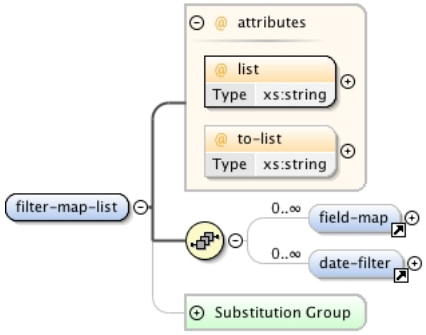
Namespace	No namespace

Annotations	Defines a field to order the results by.
Diagram	
Used by	Elements entity-find, order-map-list

Attributes

QName	Type	Fixed	Default	Use
field-name	xs:string			required
	Name of field to order list by.			

Element `filter-map-list`

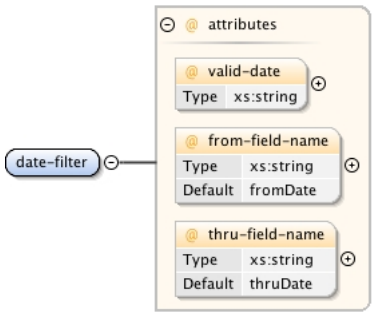
Namespace	No namespace
Annotations	Filters the given List of Maps by the field-maps specified.
Diagram	

Instance	`<filter-map-list list="" to-list="">`
	`  <field-map field-name="" from="" value="">{0,unbounded}</field-map>`
	`  <date-filter from-field-name="fromDate" thru-field-name="thruDate" valid-date="">{0,unbounded}</date-filter>`
	`</filter-map-list>`

Attributes

QName	Type	Fixed	Default	Use
list	xs:string			required
	The name of the field that contains a List of Map objects.			
to-list	xs:string			optional
	Optional name of the output list. If empty filter the input list in-place.			

Element `date-filter`

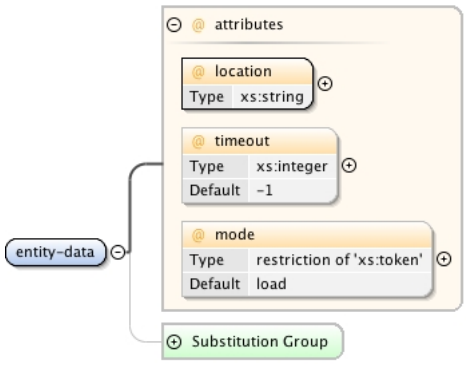
Namespace	No namespace
Annotations	Adds a constraint to find to filter by the from and thru dates in each record, comparing them to the valid-date value.
Diagram	
Used by	Elements econditions, entity-delete-by-condition, entity-find, entity-find-count, filter-map-list, having-econditions

Attributes

QName	Type	Fixed	Default	Use
from-field-name	xs:string		fromDate	optional

QName	Type	Fixed	Default	Use
	The name of the entity field to use as the from/beginning effective date. Defaults to fromDate.			
thru-field-name	xs:string		thruDate	optional
	The name of the entity field to use as the thru/ending effective date. Defaults to thruDate.			
valid-date	xs:string			optional
	The name of a field in the context to compare each value to. Defaults to now.			

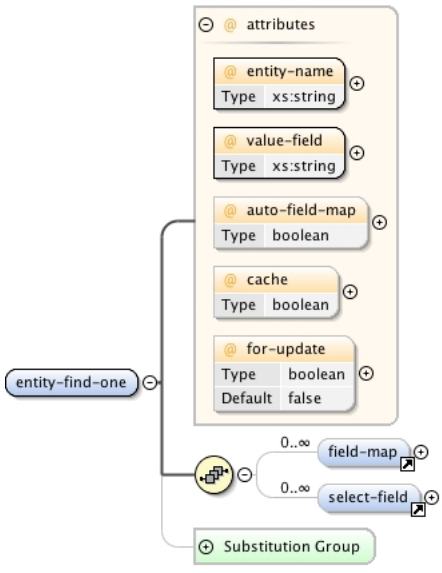
Element `entity-data`

Namespace	No namespace
Annotations	Load or assert each record in an entity-facade-xml file.
Diagram	

Attributes

QName	Type	Fixed	Default	Use
location	xs:string			required
	Location of an XML file to load in database or verify in assert mode.			
mode	restriction of xs:token		load	optional
timeout	xs:integer		-1	optional
	Start a new transaction and load the data with a longer timeout.			

Element `entity-find-one`

Namespace	No namespace
Annotations	Does a find by primary key. If no value is found does nothing to the value-field.
Diagram	

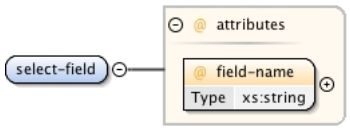
Instance	`<entity-find-one auto-field-map="" cache="" entity-name="" for-update="false" value-field="">`

```
  <field-map field-name="" from="" value="">{0,unbounded}</field-map>
  <select-field field-name="">{0,unbounded}</select-field>
</entity-find-one>
```

Attributes

QName	Type	Fixed	Default	Use
auto-field-map	boolean			optional
	Looks for all primary key fields by name in the context, and in the parameters map if not found in the context. Defaults to true. If a field-map sub-element is found this will default to false instead.			
cache	boolean			optional
	Look in the cache before finding in the datasource. The default for this comes from the cache attribute on the entity definition.			
entity-name	xs:string			required
	Name of the entity to find an instance of.			
for-update	boolean		false	optional
	Lock the selected record so only this transaction can change it until it is ended (committed or rolled back). This does not have to be set to true in order to update the record, it just keeps other transactions from updating it. In SQL this does a select for update. If this is true the cache will not be used, regardless of the cache attribute here and on the entity definition.			
value-field	xs:string			required
	Field to put result resulting EntityValue object in.			

Element `select-field`

Namespace	No namespace
Annotations	Used to specify fields to select. If there are none of these elements all fields will be selected.
Diagram	
Used by	Elements entity-find, entity-find-count, entity-find-one

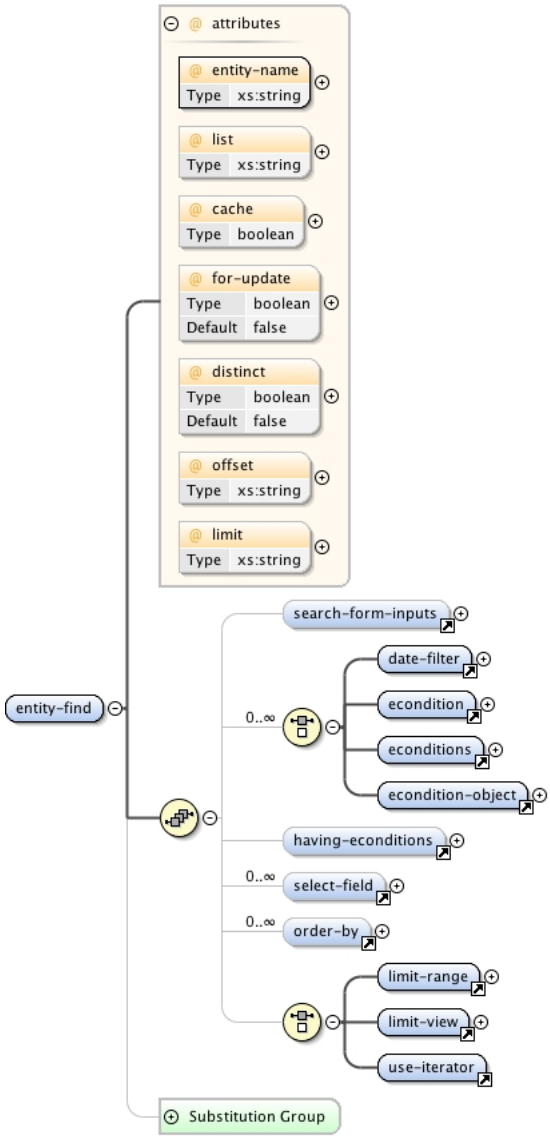
Attributes

QName	Type	Fixed	Default	Use
field-name	xs:string			required
	Name of a field to select.			

Element `entity-find`

Namespace	No namespace
Annotations	Like entity-and returns a list of entity values if any are found, otherwise returns an empty list. Use any combination of constraint, constraints and constraint-object.

Diagram



Instance

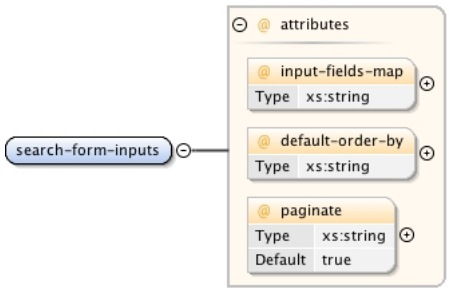
```
<entity-find cache="" distinct="false" entity-name="" for-
update="false" limit="" list="" offset="">
  <search-form-inputs default-order-by="" input-fields-map="" paginate="true">{0,1}</
search-form-inputs>
  <having-econditions combine="and">{0,1}</having-econditions>
  <select-field field-name="">{0,unbounded}</select-field>
  <order-by field-name="">{0,unbounded}</order-by>
</entity-find>
```

Attributes

QName	Type	Fixed	Default	Use
cache	boolean			optional
	Look in the cache before finding in the datasource. The default for this comes from the cache attribute on the entity definition.			
distinct	boolean		false	optional
	Get only distinct results, based on the combination of all fields selected. Defaults to false.			
entity-name	xs:string			required
	Name of entity to find instances of.			
for-update	boolean		false	optional
	Lock the selected record so only this transaction can change it until it is ended (committed or rolled back). This does not have to be set to true in order to update the record, it just keeps other transactions from updating it. In SQL this does a select for update.			

QName	Type	Fixed	Default	Use
	If this is true the cache will not be used, regardless of the cache attribute here and on the entity definition.			
limit	xs:string			optional
	Get back only this many results.			
list	xs:string			optional
	Name of the list to put results in. Required unless the entity-find is used under the entity-options element in a XML Screen Form.			
offset	xs:string			optional
	Get back results starting at this offset.			

Element `search-form-inputs`

Namespace	No namespace
Annotations	Adds econditions for the fields found in the input-fields-map. The fields and special fields with suffixes supported are the same as the *-find fields in the XML Forms. This means that you can use this to process the data from the various inputs generated by XML Forms. The suffixes include things like *_op for operators and *_ic for ignore case. For historical reference, this does basically what the Apache OFBiz prepareFind service does.
Diagram	
Used by	Element entity-find	

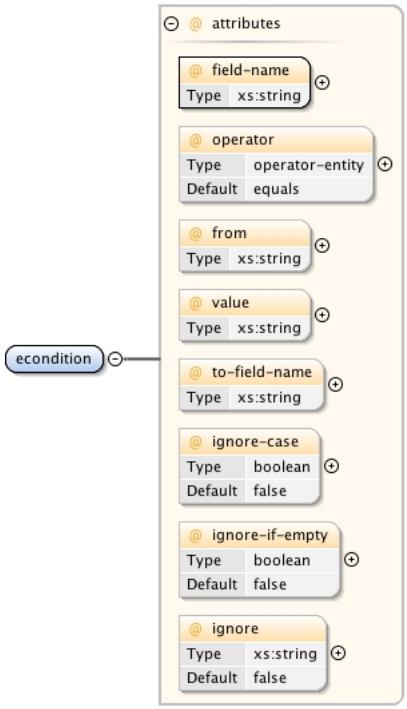
Attributes

QName	Type	Fixed	Default	Use
default-order-by	xs:string			optional
	If no orderByField parameter, order by this.			
input-fields-map	xs:string			optional
	The map to get form fields from. If empty will look at the ec.web.parameters map if the web facade is available, otherwise the current context (ec.context).			
paginate	xs:string		true	optional
	Indicate if this find should set pagination options even if there are no pageSize and pageIndex parameters. Also adds a context field called "${entity-find.@list}Count" with a count of the total possible results (ie without the offset/limit). Defaults to true.			

Element `econdition`

Namespace	No namespace
Annotations	Adds a econdition to the query to compare the field-name field to a context field, a String value, or another field on the entity.

Diagram



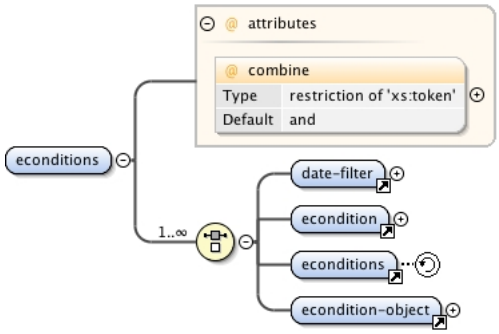
Used by

Elements econditions, entity-delete-by-condition, entity-find, entity-find-count, having-econditions

Attributes

QName	Type	Fixed	Default	Use
field-name	xs:string			required
	The field on the entity to constrain on. If from, value and to-field-name are all empty this is also used as the name of the context field to compare to.			
from	xs:string			optional
	Field expression in the context to compare the entity field to.			
ignore	xs:string		false	optional
	Ignore the econdition (leave out of the find) if set to true. Defaults to false.			
ignore-case	boolean		false	optional
	Ignore case when doing the compare. Defaults to false.			
ignore-if-empty	boolean		false	optional
	Leave out the constraint if the comparison value is empty or null. Defaults to false.			
operator	operator-entity		equals	optional
	Operator to apply to field-name on one side, and from, value, or to-field-name on the other side. For the between operator the from should be a Collection with exactly 2 values in it. For the in operator the from should be a Collection with 1 to many values in it. For the like operator use the standard SQL wildcards, including "%" for any number of characters (like *) and "_" for a single character (like ?), and escape them with a "!" in from of each character to escape). Defaults to equals.			
to-field-name	xs:string			optional
	Compare the field-name field to another field on the entity.			
value	xs:string			optional
	Comparison value, use ${} syntax to expand variables.			

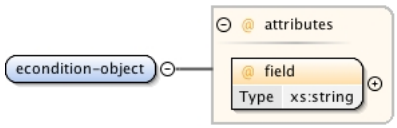
Element econditions

Namespace	No namespace
Annotations	The econditions element contains a list of econditions that are combined with either and or or. The default is and. You can have econditions under econditions, for building fairly complex econdition trees, and you can also drop in econdition-objects at any point.
Diagram	
Used by	Elements econditions, entity-delete-by-condition, entity-find, entity-find-count, having-econditions
Instance	<econditions combine="and"> <date-filter from-field-name="fromDate" thru-field-name="thruDate" valid-date="">{1,1}</date-filter> <econdition field-name="" from="" ignore="false" ignore-case="false" ignore-if-empty="false" operator="equals" to-field-name="" value="">{1,1}</econdition> <econditions combine="and">{1,1}</econditions> <econdition-object field="">{1,1}</econdition-object> </econditions>

Attributes

QName	Type	Fixed	Default	Use
combine	restriction of xs:token		and	optional
	Operator to use to combine econditions in the list.			

Element econdition-object

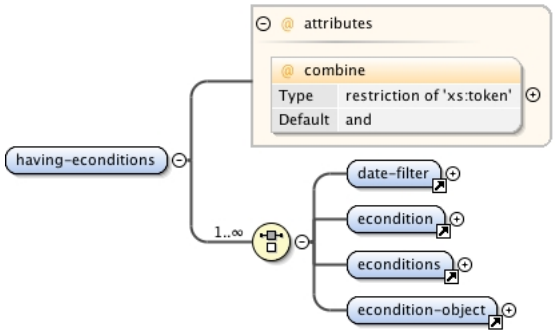
Namespace	No namespace
Annotations	Add a condition that has been defined elsewhere and is available in the current context. Can also be a Map and it will add conditions where the entries are ANDed together and each key/value are compared with equal.
Diagram	
Used by	Elements econditions, entity-delete-by-condition, entity-find, entity-find-count, having-econditions

Attributes

QName	Type	Fixed	Default	Use
field	xs:string			required
	Field in the current context that implements the EntityCondition interface or the Map interface.			

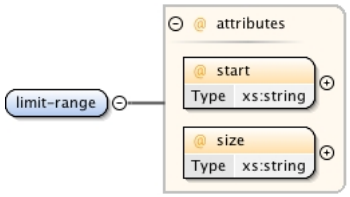
Element having-econditions

Namespace	No namespace
Annotations	Similar to econditions but runs after the grouping and functions are done.

Diagram	
Used by	Elements entity-find, entity-find-count
Instance	`<having-econditions combine="and"> <date-filter from-field-name="fromDate" thru-field-name="thruDate" valid-date="">{1,1}</date-filter> <econdition field-name="" from="" ignore="false" ignore-case="false" ignore-if-empty="false" operator="equals" to-field-name="" value="">{1,1}</econdition> <econditions combine="and">{1,1}</econditions> <econdition-object field="">{1,1}</econdition-object> </having-econditions>`

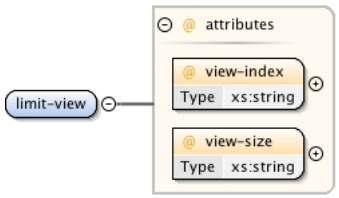
Attributes	QName	Type	Fixed	Default	Use
	combine	restriction of xs:token		and	optional
		Operator to use to combine econditions in the list.			

Element `limit-range`

Namespace	No namespace
Annotations	Limit the results by a start index and a size.
Diagram	
Used by	Element entity-find

Attributes	QName	Type	Fixed	Default	Use
	size	xs:string			required
		The number of results to include beyond the start.			
	start	xs:string			required
		The start/beginning index of results to include.			

Element `limit-view`

Namespace	No namespace
Annotations	Limit the results using parameters like those used to paginate results in a user interface.
Diagram	
Used by	Element entity-find

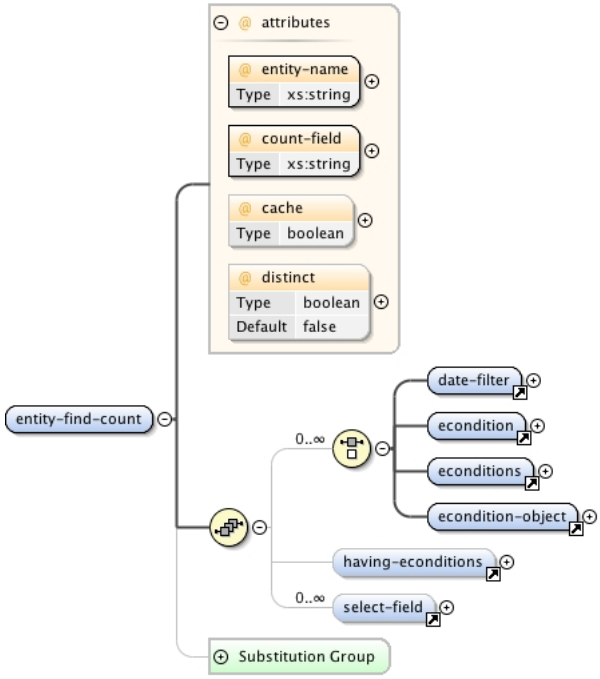
Attributes	QName	Type	Fixed	Default	Use
	view-index	xs:string			required

QName	Type	Fixed	Default	Use
	Index of records to view, depends on view-size.			
view-size	xs:string			required
	Number of records to view, like the number of results per-screen.			

Element `use-iterator`

Namespace	No namespace
Annotations	Specifies whether or not to use the EntityListIterator when doing the query. This is much more efficient for large data sets because the results are read incrementally instead of all at once. Note that when using this the use-cache setting will be ignored. Also note that an EntityListIterator must be closed when you are finished, but this is done automatically by the iterate operation. Must be true or false, defaults to false.
Diagram	
Used by	Element entity-find

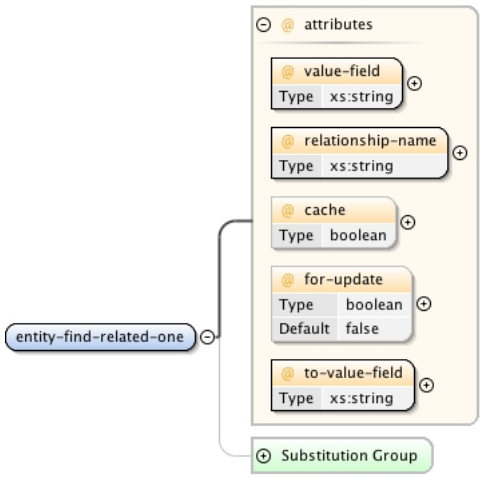
Element `entity-find-count`

Namespace	No namespace
Annotations	Find the count of the number of records that match the given conditions. Conditions follow the same structure as in the entity-find operation.
Diagram	
Instance	<entity-find-count cache="" count-field="" distinct="false" entity-name=""> <date-filter from-field-name="fromDate" thru-field-name="thruDate" valid-date="">{1,1}</date-filter> <econdition field-name="" from="" ignore="false" ignore-case="false" ignore-if-empty="false" operator="equals" to-field-name="" value="">{1,1}</econdition> <econditions combine="and">{1,1}</econditions> <econdition-object field="">{1,1}</econdition-object> <having-econditions combine="and">{0,1}</having-econditions> <select-field field-name="">{0,unbounded}</select-field> </entity-find-count>

Attributes	QName	Type	Fixed	Default	Use
	cache	boolean			optional
		Look in the cache before finding in the datasource. The default for this comes from the cache attribute on the entity definition.			

QName	Type	Fixed	Default	Use
count-field	xs:string			required
	Name of the field (variable) to put result of the count in.			
distinct	boolean		false	optional
	Get only distinct results, based on the combination of all fields selected. Defaults to false.			
entity-name	xs:string			required
	Name of entity to search in.			

Element `entity-find-related-one`

Namespace	No namespace
Annotations	Find a single value related to an existing value.
Diagram	

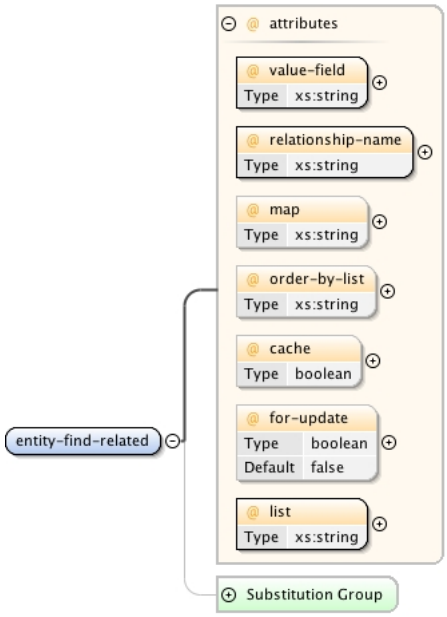
Attributes

QName	Type	Fixed	Default	Use
cache	boolean			optional
	Look in the cache before finding in the datasource. The default for this comes from the cache attribute on the entity definition.			
for-update	boolean		false	optional
	Lock the selected record so only this transaction can change it until it is ended (committed or rolled back). This does not have to be set to true in order to update the record, it just keeps other transactions from updating it. In SQL this does a select for update. If this is true the cache will not be used, regardless of the cache attribute here and on the entity definition.			
relationship-name	xs:string			required
	Name of the relationship to use, consists of the relationship title and the related entity name, like: ${title}${related-entity-name}.			
to-value-field	xs:string			required
	Name of field to put the entity value result in.			
value-field	xs:string			required
	Name of the existing entity value in the context.			

Element `entity-find-related`

Namespace	No namespace
Annotations	Find a list of values related to a specific value.

Diagram

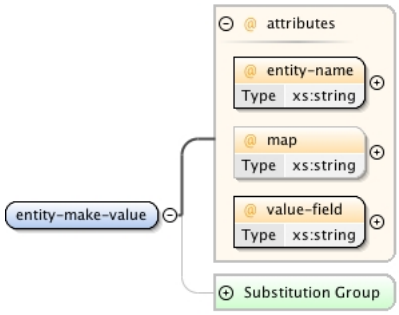


Attributes

QName	Type	Fixed	Default	Use
cache	boolean			optional
	Look in the cache before finding in the datasource. The default for this comes from the cache attribute on the entity definition.			
for-update	boolean		false	optional
	Lock the selected record so only this transaction can change it until it is ended (committed or rolled back). This does not have to be set to true in order to update the record, it just keeps other transactions from updating it. In SQL this does a select for update. If this is true the cache will not be used, regardless of the cache attribute here and on the entity definition.			
list	xs:string			required
	Name of the list to put the entity list result in.			
map	xs:string			optional
	A map containing extra constraints for the find.			
order-by-list	xs:string			optional
	A list of field names to order the results by.			
relationship-name	xs:string			required
	Name of the relationship to use, consists of the relationship title and the related entity name,like: ${title}${related-entity-name}.			
value-field	xs:string			required
	Name of the existing entity value in the context.			

Element `entity-make-value`

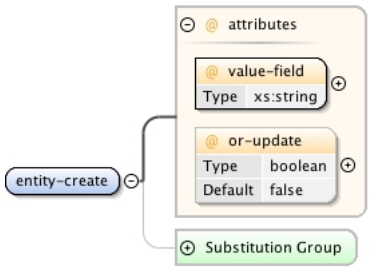
Namespace	No namespace
Annotations	The make-value tag uses the delegator to construct an entity value. The resulting value will not exist in the database, but will simply be assembled using the entity-name and fields map. The resulting EntityValue object will be placed in the method environment using the specified value-field.

Diagram	

Attributes

QName	Type	Fixed	Default	Use
entity-name	xs:string			required
	The name of the entity to construct an instance of.			
map	xs:string			optional
	The name of a map in the method environment that will be used for the entity fields.If the map is an EntityValue object then this will clone the value.			
value-field	xs:string			required
	The name of the field where the EntityValue object will be put.			

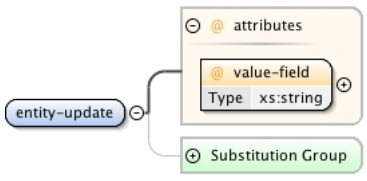
Element `entity-create`

Namespace	No namespace
Annotations	The create-value tag persists the specified EntityValue object by creating a new instance of the entity in the datasource. An error will result if an instance of the entity exists in the datasource with the same primary key.
Diagram	

Attributes

QName	Type	Fixed	Default	Use
or-update	boolean		false	optional
	Update value if already exists instead of returning an error, defaults to false.			
value-field	xs:string			required
	The name of the field that contains the EntityValue object.			

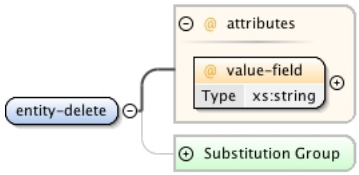
Element `entity-update`

Namespace	No namespace
Annotations	Updates the specified EntityValue object in the datasource. An error will result if the record is not found in the datasource.
Diagram	

Attributes

QName	Type	Fixed	Default	Use
value-field	xs:string			required
	The name of the field that contains the EntityValue object.			

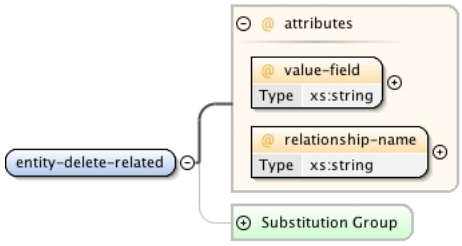
Element entity-delete

Namespace	No namespace
Annotations	Deletes the specified EntityValue object from the datasource. An error will result if the record is not found in the datasource.
Diagram	

Attributes

QName	Type	Fixed	Default	Use
value-field	xs:string			required
	The name of the field that contains the EntityValue object.			

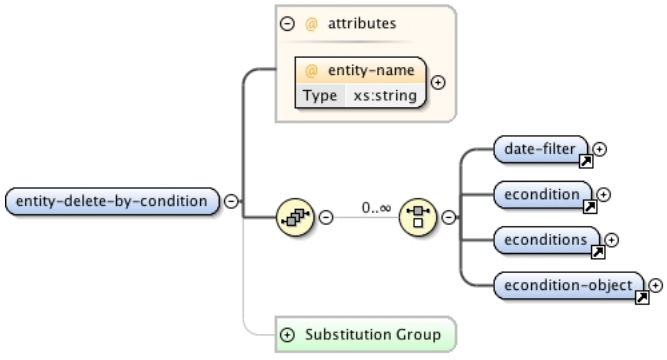
Element entity-delete-related

Namespace	No namespace
Annotations	Given a value-field and a relationship-name, follows the relationship and deletes all related records. For a type one relationship it will remove a single record if it exists, and for a type many relationship it will remove all the records that are related to it. Instead of using cascading deletes you should have your code delete all related data with foreign keys pointing the the value-field record, and then delete the value-field.
Diagram	

Attributes

QName	Type	Fixed	Default	Use
relationship-name	xs:string			required
	Name of a relationship to use to delete related records.			
value-field	xs:string			required
	Field that contains an EntityValue object to delete related records from.			

Element entity-delete-by-condition

Namespace	No namespace
Annotations	Deletes entity values that match the econditions.
Diagram	

Instance

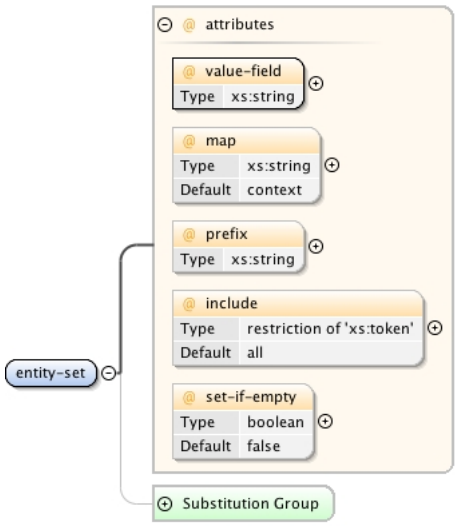
```
<entity-delete-by-condition entity-name="">
  <date-filter from-field-name="fromDate" thru-field-name="thruDate" valid-date="">{1,1}</
date-filter>
```

```
  <econdition field-name="" from="" ignore="false" ignore-case="false" ignore-if-
empty="false" operator="equals" to-field-name="" value="">{1,1}</econdition>
  <econditions combine="and">{1,1}</econditions>
  <econdition-object field="">{1,1}</econdition-object>
</entity-delete-by-condition>
```

Attributes

QName	Type	Fixed	Default	Use
entity-name	xs:string			required
	The name of the entity to remove instances of.			

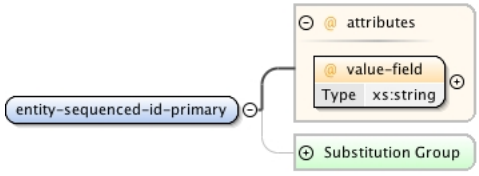
Element `entity-set`

Namespace	No namespace
Annotations	Looks for each field (pk, nonpk, or all) in the named map and if it exists there it will copy it into the named value object.
Diagram	

Attributes

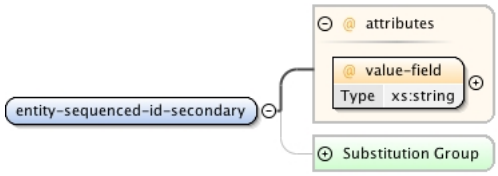
QName	Type	Fixed	Default	Use
include	restriction of xs:token		all	optional
map	xs:string		context	optional
	The name of a map in the method environment that will be used for the entity fields. Defaults to the context root, which is where incoming parameters go by default.			
prefix	xs:string			optional
	If not null or empty will be pre-pended to each field name (upper-casing the first letter of the field name first), and that will be used as the fields Map lookup name instead of the field-name.			
set-if-empty	boolean		false	optional
	Specifies whether or not to set fields that are null or empty. Defaults to false.			
value-field	xs:string			required
	Field that contains an EntityValue object.			

Element `entity-sequenced-id-primary`

Namespace	No namespace
Annotations	Get the next guaranteed unique seq id for this entity, and set it in the primary key field. This will set it in the first primary key field in the entity definition, but it really should be used for entities with only one primary key field.
Diagram	

Attributes	QName	Type	Fixed	Default	Use
	value-field	xs:string			required
		The EntityValue object to work on.			

Element `entity-sequenced-id-secondary`

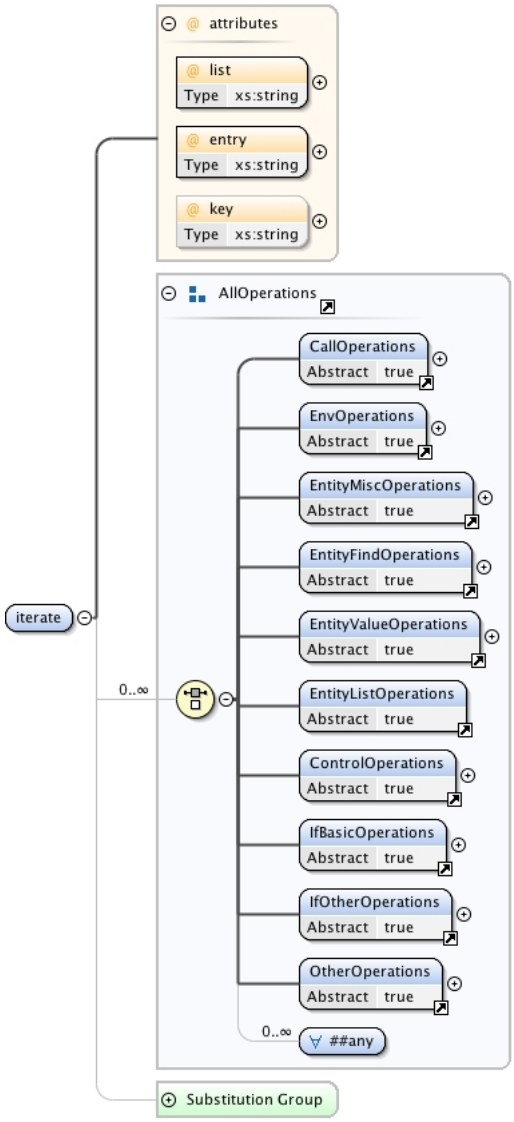
Namespace	No namespace
Annotations	Given an entity value object with all primary key fields except one already set will generate an ID for the remaining primary key field by looking at all records with the partial primary key and then adding increment-by to the highest value.
Diagram	

Attributes	QName	Type	Fixed	Default	Use
	value-field	xs:string			required
		The EntityValue object to work on.			

Element `iterate`

Namespace	No namespace
Annotations	The operations contained by the iterate tag will be executed for each of the entries in the list, and will make the current entry available in the method environment by the entry-name specified. This tag can contain any of the xml-action operations, including the conditional/if operations. Any xml-action operation can be nested under the iterate tag.

Diagram



Instance

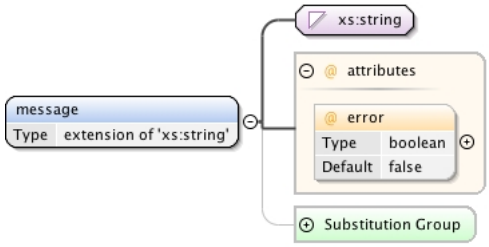
```
<iterate entry="" key="" list="">
  <CallOperations>{1,1}</CallOperations>
  <EnvOperations>{1,1}</EnvOperations>
  <EntityMiscOperations>{1,1}</EntityMiscOperations>
  <EntityFindOperations>{1,1}</EntityFindOperations>
  <EntityValueOperations>{1,1}</EntityValueOperations>
  <EntityListOperations>{1,1}</EntityListOperations>
  <ControlOperations>{1,1}</ControlOperations>
  <IfBasicOperations>{1,1}</IfBasicOperations>
  <IfOtherOperations>{1,1}</IfOtherOperations>
  <OtherOperations>{1,1}</OtherOperations>
</iterate>
```

Attributes

QName	Type	Fixed	Default	Use
entry	xs:string			required
	The name of the field that will contain each entry as we iterate through the list.			
key	xs:string			optional
	If list points to a Map or Collection of Map.Entry the key will be put where this refers to, the value where the entry attribute refers to.			
list	xs:string			required
	The name of the field that contains the list to iterate over.			

Element message

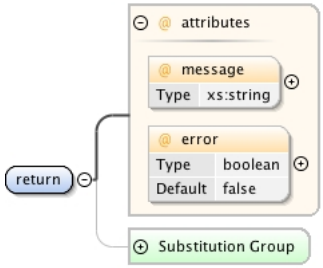
Namespace	No namespace

Annotations	Adds the message (sub-element text) to the ExecutionContext MessageFacade, either the errors list if error=true or the messages list otherwise.
Diagram	
Type	extension of xs:string
Attributes	**QName** / **Type** / **Fixed** / **Default** / **Use** **error** / boolean / / false / optional If true will be considered caused by an error, meaning transaction will be rolled back, etc.

Element `check-errors`

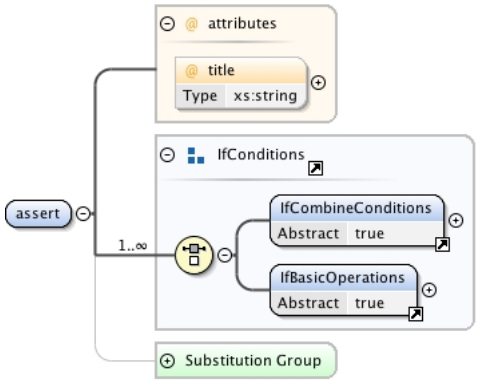
Namespace	No namespace
Annotations	Check the ExecutionContext error message list (ec.message.errors) and if it is not empty return as an error immediately.
Diagram	

Element `return`

Namespace	No namespace
Annotations	Returns immediately.
Diagram	
Attributes	**QName** / **Type** / **Fixed** / **Default** / **Use** **error** / boolean / / false / optional If true will be considered caused by an error, meaning transaction will be rolled back, etc. **message** / xs:string / / / optional Adds a message to the errors list (ec.message.errors) if error=true, the messages list (ec.message.messages) otherwise.

Element `assert`

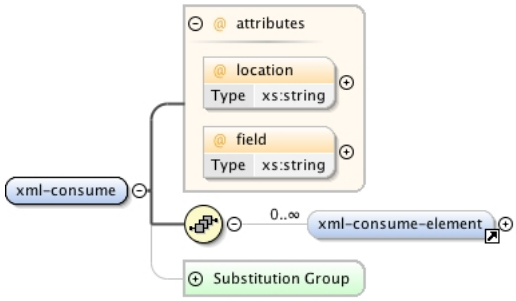
Namespace	No namespace
Annotations	Each condition under the assert element will be checked and if it fails an error will be added to the given error list. Note that while the definitions for the if-* operations are used, the tags should be empty because of the differing semantics. This is mainly used for testing, and for writing xml-actions that are meant to be used as part of a test suite. This is mostly useful for testing because the messages are targeted at a programmer, and not really at an end user.

Diagram	

Instance	`<assert title="">`
	`  <IfCombineConditions>{1,1}</IfCombineConditions>`
	`  <IfBasicOperations>{1,1}</IfBasicOperations>`
	`</assert>`

Attributes	**QName** | **Type** | **Fixed** | **Default** | **Use** **title** | xs:string | | | optional A title that can be used in reports for testing.

Element `xml-consume`

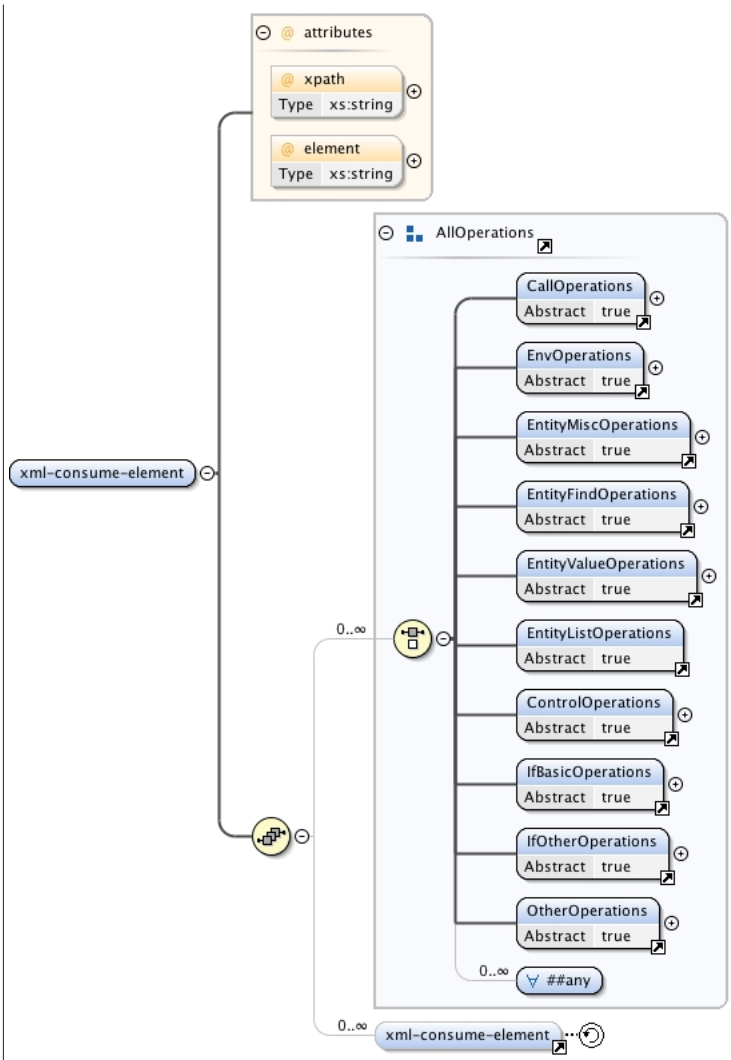
Namespace	No namespace
Annotations	Used to process/consume an XML document. The document can be either a text file at a location or can be in a field in the current context that is either an XML text document or a org.w3c.dom.Document object or even a org.w3c.dom.Element object.
Diagram	

Instance	`<xml-consume field="" location="">`
	`  <xml-consume-element element="" xpath="">{0,unbounded}</xml-consume-element>`
	`</xml-consume>`

Attributes	**QName** | **Type** | **Fixed** | **Default** | **Use** **field** | xs:string | | | optional **location** | xs:string | | | optional

Element `xml-consume-element`

Namespace	No namespace
Annotations	Process a single or list of XML elements looked up using an xpath expression relative to the current element (or root element if right under the xml-consume element). The sub-operations and xml-consume-element tags will be run for each element matching the xpath expression.

Diagram	
Used by	Elements xml-consume, xml-consume-element

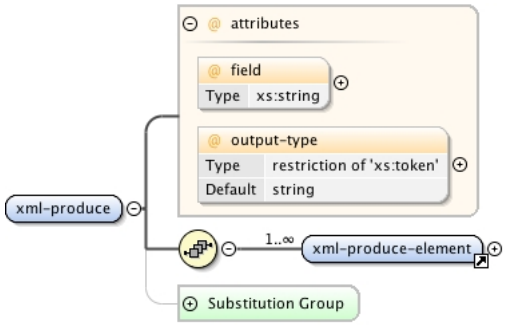
Instance	`<xml-consume-element element="" xpath="">`
	`  <CallOperations>{1,1}</CallOperations>`
	`  <EnvOperations>{1,1}</EnvOperations>`
	`  <EntityMiscOperations>{1,1}</EntityMiscOperations>`
	`  <EntityFindOperations>{1,1}</EntityFindOperations>`
	`  <EntityValueOperations>{1,1}</EntityValueOperations>`
	`  <EntityListOperations>{1,1}</EntityListOperations>`
	`  <ControlOperations>{1,1}</ControlOperations>`
	`  <IfBasicOperations>{1,1}</IfBasicOperations>`
	`  <IfOtherOperations>{1,1}</IfOtherOperations>`
	`  <OtherOperations>{1,1}</OtherOperations>`
	`  <xml-consume-element element="" xpath="">{0,unbounded}</xml-consume-element>`
	`</xml-consume-element>`

Attributes

QName	Type	Fixed	Default	Use
element	xs:string			optional
	The name of the field the element/node object will go into for the operations under this tag.This object can be treated either as a org.w3c.dom.Element or as a Map.			
xpath	xs:string			optional

Element `xml-produce`

Namespace	No namespace
Annotations	Used to produce/create an XML document object. Must have at least one xml-produce-element operation under it and that becomes the root element.

Diagram	

Instance	`<xml-produce field="" output-type="string">`
	`  <xml-produce-element element="" name="" parent-element="">{1,unbounded}</xml-produce-element>`
	`</xml-produce>`

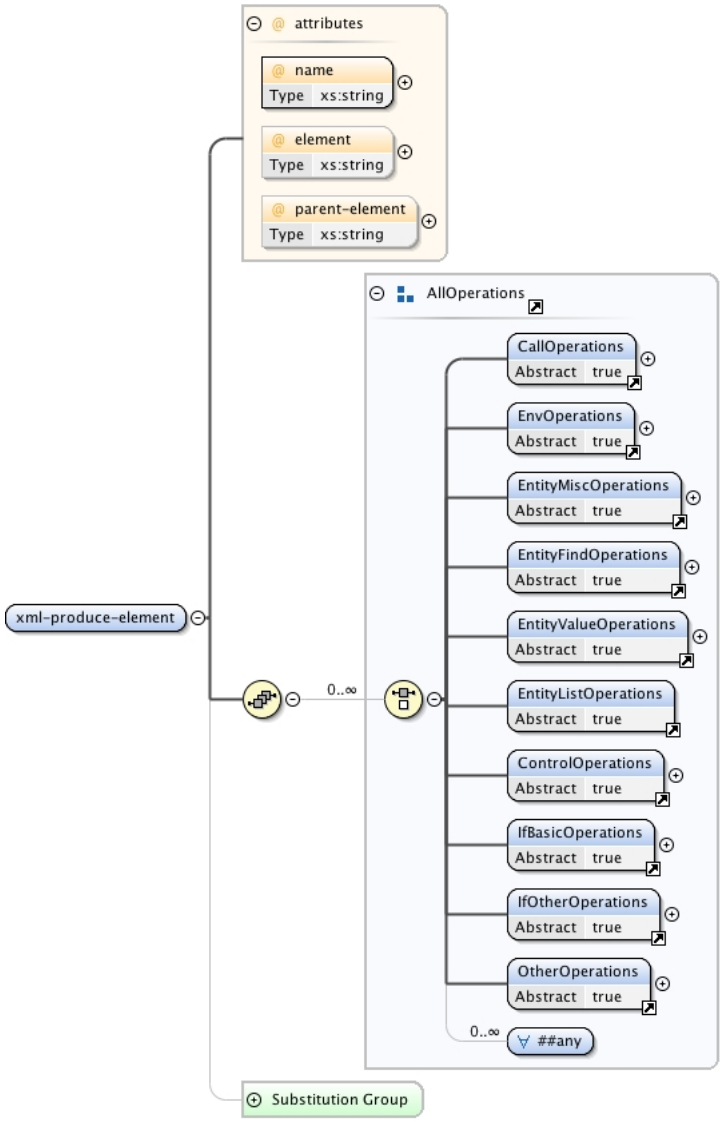
Attributes

QName	Type	Fixed	Default	Use
field	xs:string			optional
output-type	restriction of xs:token		string	optional
	The type of output to produce. Either the document object created while adding elements, or that object converted to a String.			

Element `xml-produce-element`

Namespace	No namespace
Annotations	Create a single XML element, added to a document under the named parent element field. The element created becomes the parent element for any child xml-produce-element operations encountered.

Diagram



Used by

Element xml-produce

`Instance`

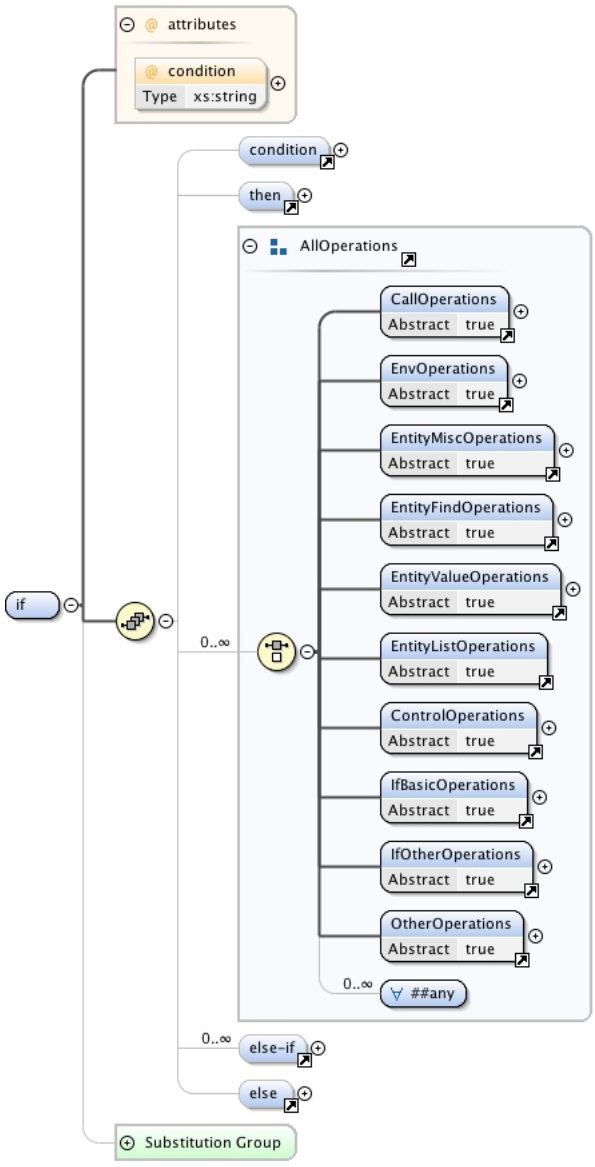
```
<xml-produce-element element="" name="" parent-element="">
  <CallOperations>{1,1}</CallOperations>
  <EnvOperations>{1,1}</EnvOperations>
  <EntityMiscOperations>{1,1}</EntityMiscOperations>
  <EntityFindOperations>{1,1}</EntityFindOperations>
  <EntityValueOperations>{1,1}</EntityValueOperations>
  <EntityListOperations>{1,1}</EntityListOperations>
  <ControlOperations>{1,1}</ControlOperations>
  <IfBasicOperations>{1,1}</IfBasicOperations>
  <IfOtherOperations>{1,1}</IfOtherOperations>
  <OtherOperations>{1,1}</OtherOperations>
</xml-produce-element>
```

Attributes

QName	Type	Fixed	Default	Use
element	xs:string			optional
	The name of the field the element/node object will go into for the operations under this tag. This object can be treated either as a org.w3c.dom.Element or as a Map to set attributes on the element.			
name	xs:string			required
parent-element	xs:string			optional
	This operation is usually nested somewhere under another xml-produce-element operation and under an xml-produce operation. In those cases there will be a default parent element field that will be automatically used if this is not specified, otherwise an			

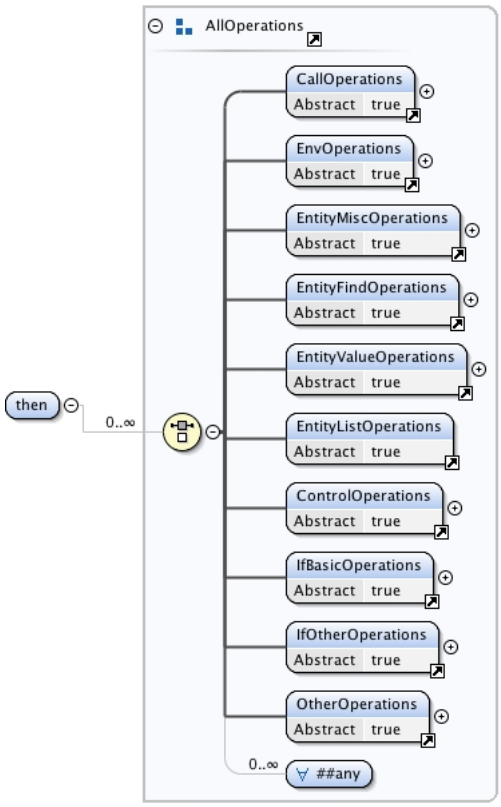
	QName	Type	Fixed	Default	Use
		element field must be specified here (with an object type org.w3c.dom.Element).			

Element `if`

Namespace	No namespace
Annotations	The if operation offers a flexible way of specifying combinations of conditions, alternate conditions, and operations to run on true evaluation of the conditions or to run otherwise. The other if operations are meant for a specific, simple condition when used outside of the condition sub-element of this operation. The attributes of the other if operations are the same when used inside this operation. Note that while the definitions for the if-* operations are used, the tags should be empty because of the differing semantics.
Diagram	
Instance	<if condition=""> <condition>{0,1}</condition> <then>{0,1}</then> <else-if condition="">{0,unbounded}</else-if> <else>{0,1}</else> </if>

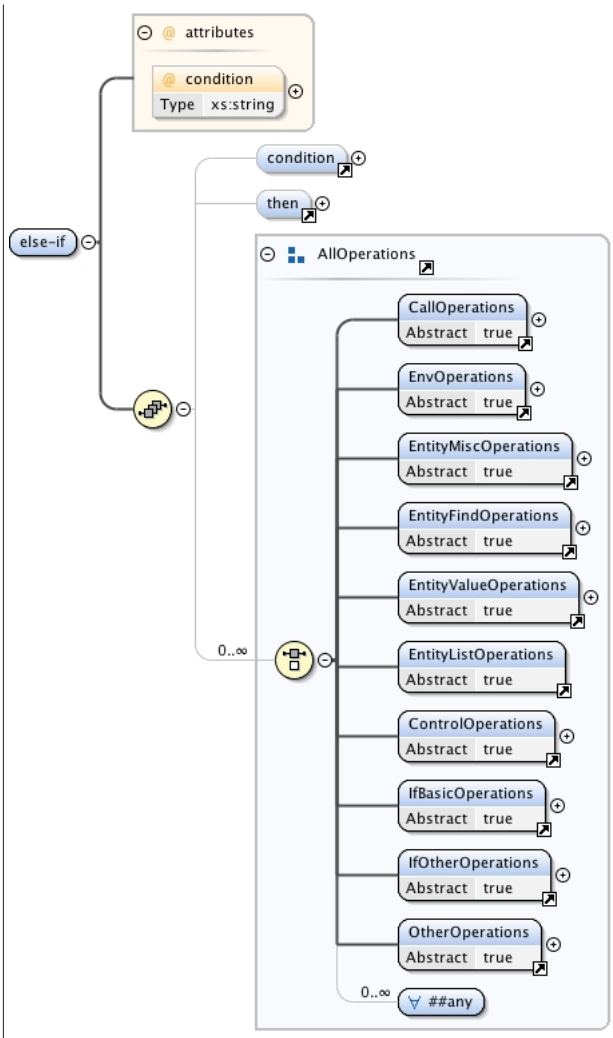
Attributes	QName	Type	Fixed	Default	Use
	condition	xs:string			optional
		A boolean expression in Groovy. Will be AND combined with other conditions if present.			

Element `then`

Namespace	No namespace
Annotations	Operations to run if the corresponding condition evaluate to true.
Diagram	
Used by	Elements else-if, if, while	
`Instance`	<then> <CallOperations>{1,1}</CallOperations> <EnvOperations>{1,1}</EnvOperations> <EntityMiscOperations>{1,1}</EntityMiscOperations> <EntityFindOperations>{1,1}</EntityFindOperations> <EntityValueOperations>{1,1}</EntityValueOperations> <EntityListOperations>{1,1}</EntityListOperations> <ControlOperations>{1,1}</ControlOperations> <IfBasicOperations>{1,1}</IfBasicOperations> <IfOtherOperations>{1,1}</IfOtherOperations> <OtherOperations>{1,1}</OtherOperations> </then>	

Element `else-if`

Namespace	No namespace
Annotations	The else-if element can be used to specify alternate conditional execution blocks. Each else-if element must contain two sub-elements: condition and then. If the condition of the parent is evaluated to false, each condition of the else-if sub-elements will be evaluated, and the operations under the element corresponding to the first condition that evaluates to true will be run.

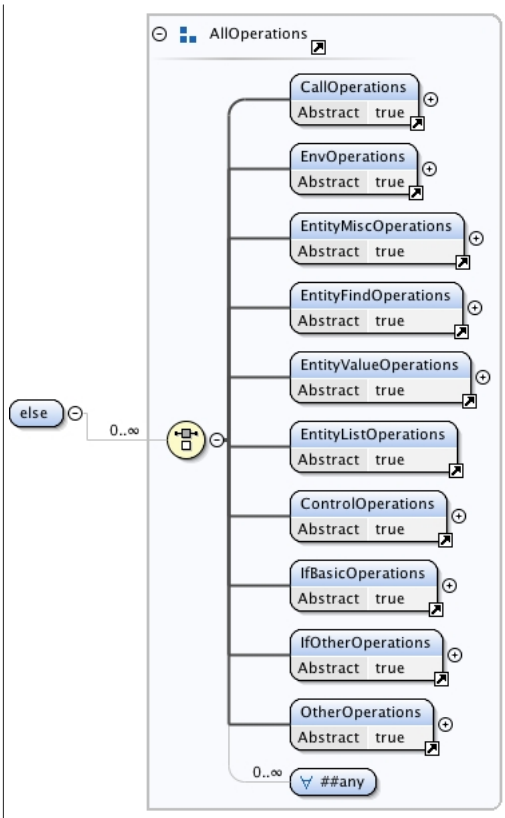
Diagram	
Used by	Element if

Instance	`<else-if condition="">`
	`  <condition>{0,1}</condition>`
	`  <then>{0,1}</then>`
	`</else-if>`

Attributes	**QName** | **Type** | **Fixed** | **Default** | **Use** **condition** | xs:string | | | optional A boolean expression in Groovy. Will be AND combined with other conditions if present.

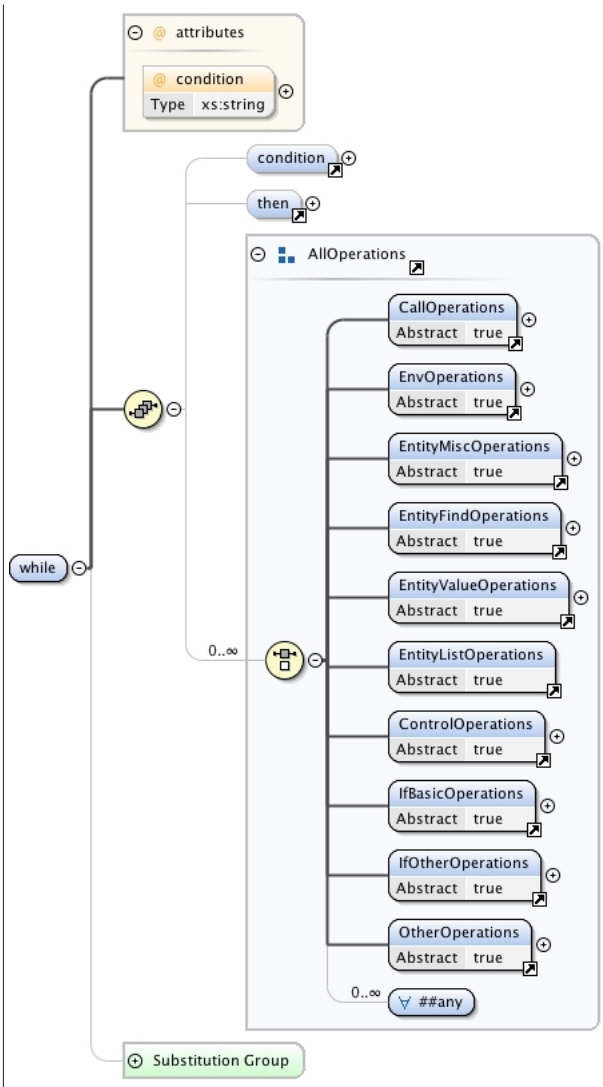
Element `else`

Namespace	No namespace
Annotations	The else element can be used to contain operations that will run if the condition evaluates to false, and when under an if element when no else-if sub-conditions evaluate to true. It can contain any xml-actions operation.

Diagram	
Used by	Elements compare, if
Instance	<else> <CallOperations>{1,1}</CallOperations> <EnvOperations>{1,1}</EnvOperations> <EntityMiscOperations>{1,1}</EntityMiscOperations> <EntityFindOperations>{1,1}</EntityFindOperations> <EntityValueOperations>{1,1}</EntityValueOperations> <EntityListOperations>{1,1}</EntityListOperations> <ControlOperations>{1,1}</ControlOperations> <IfBasicOperations>{1,1}</IfBasicOperations> <IfOtherOperations>{1,1}</IfOtherOperations> <OtherOperations>{1,1}</OtherOperations> </else>

Element `while`

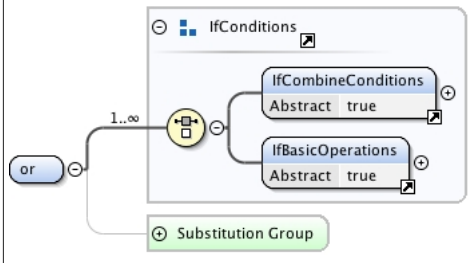
Namespace	No namespace
Annotations	`While loop operation, uses the same condition element as the if operation.`

Diagram	

Instance	<pre><while condition="">
	  <condition>{0,1}</condition>
	  <then>{0,1}</then>
	</while></pre>

Attributes	**QName** / **Type** / **Fixed** / **Default** / **Use** **condition** / xs:string / / / optional A boolean expression in Groovy. Will be AND combined with other conditions if present.

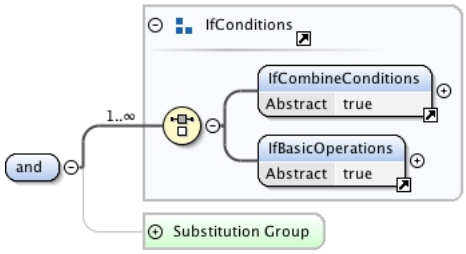
Element or

Namespace	No namespace
Annotations	To be true just one of the conditions underneath needs to be true. Will return true as soon as a condition is true, not evaluating remaining conditions.
Diagram	

Instance	<pre><or>
	  <IfCombineConditions>{1,1}</IfCombineConditions>
	  <IfBasicOperations>{1,1}</IfBasicOperations></pre>

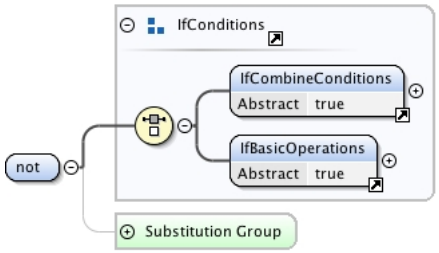
	`</or>`

Element and

Namespace	No namespace
Annotations	To be true all of the conditions underneath need to be true. Will return false as soon as a condition evaluates to false, not evaluating remaining conditions. If no conditions evaluate to false will return true.
Diagram	

Instance	`<and>`
	`  <IfCombineConditions>{1,1}</IfCombineConditions>`
	`  <IfBasicOperations>{1,1}</IfBasicOperations>`
	`</and>`

Element not

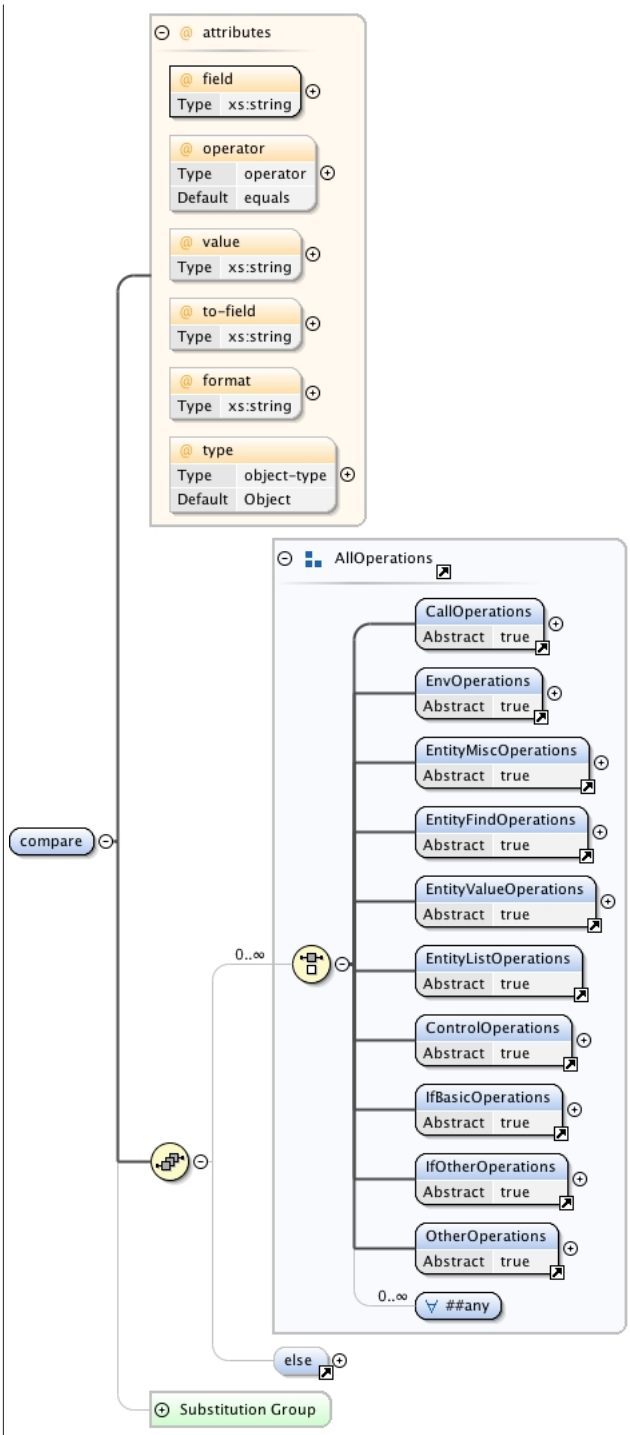
Namespace	No namespace
Annotations	Can only have one condition underneath and simply reverse the boolean value of this condition.
Diagram	

Instance	`<not>`
	`  <IfCombineConditions>{1,1}</IfCombineConditions>`
	`  <IfBasicOperations>{1,1}</IfBasicOperations>`
	`</not>`

Element compare

Namespace	No namespace
Annotations	The operations contained by the if-compare tag will only be executed if the comparison returns true. This tag can contain any of the xml-action operations, including the conditional/if operations.

Diagram



Instance

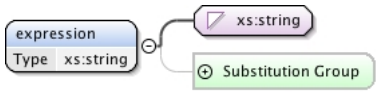
```
<compare field="" format="" operator="equals" to-field="" type="Object" value="">
  <CallOperations>{1,1}</CallOperations>
  <EnvOperations>{1,1}</EnvOperations>
  <EntityMiscOperations>{1,1}</EntityMiscOperations>
  <EntityFindOperations>{1,1}</EntityFindOperations>
  <EntityValueOperations>{1,1}</EntityValueOperations>
  <EntityListOperations>{1,1}</EntityListOperations>
  <ControlOperations>{1,1}</ControlOperations>
  <IfBasicOperations>{1,1}</IfBasicOperations>
  <IfOtherOperations>{1,1}</IfOtherOperations>
  <OtherOperations>{1,1}</OtherOperations>
  <else>{0,1}</else>
</compare>
```

Attributes

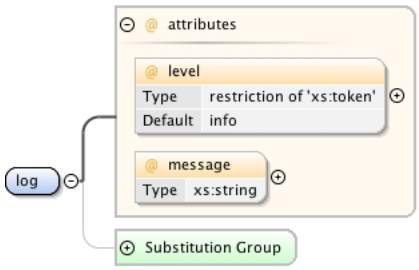
QName	Type	Fixed	Default	Use
field	xs:string			required
	The name of the field in the context (environment) that will be			

QName	Type	Fixed	Default	Use
	compared.			
format	xs:string			optional
	Format string based on the type of the object (date, number, etc).			
operator	operator		equals	optional
to-field	xs:string			optional
	The name of the context field that the main field will be compared to. If left empty will default to the field attribute's value.			
type	object-type		Object	optional
value	xs:string			optional
	The value that the field will compared to. Will evaluate to a String but can be converted to other types.			

Element `expression`

Namespace	No namespace
Annotations	A boolean expression should be inline under this element (to avoid problems with character encoding, etc). When not under a condition element any xml-action operation can be nested under this tag, and will only be run if it evaluates to true.
Diagram	
Type	xs:string

Element `log`

Namespace	No namespace
Annotations	Logs a message using Log4J.
Diagram	

Attributes	QName	Type	Fixed	Default	Use
	level	restriction of xs:token		info	optional
		The logging/debug level to use.			
	message	xs:string			optional
		The message to log. Can insert variables using the ${} syntax.			