**Manuale delle regole**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Protocollo: Asi modello standard relazioni**

|  |  |
| --- | --- |
| Autore: | **Nicolò Fassina** |
| Versione: | **1.0** |
| Data di Rilascio: | **07/08/2020** |
| N° di pagine totali: | **9** |
| Rilasciato da: | **Nicolò Fassina** |
| Distribuito a: | **Asi** |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ai sensi degli Artt. 13 e 14 del GDPR 2016/679, vi informiamo che i vostri dati, acquisiti e/o acquisibili nel corso del rapporto giuridico tra noi intercorrente, sono o potranno essere oggetto di trattamento da parte della nostra società per fini d’adempimento contrattuale, amministrativo-contabile e di comunicazione commerciale ai soli prodotti e servizi da noi proposti e, più in generale, nel rispetto della normativa sopra citata e degli obblighi di sicurezza e riservatezza previsti. Informativa completa è disponibile presso il sito: www.plain.it

**Sommario**

[1 Scopo del documento 3](#_Toc47685832)

[2 Tipologie di regole 3](#_Toc47685833)

[2.1 Regole di trasformazione 3](#_Toc47685834)

[2.1.1 Add 3](#_Toc47685835)

[2.1.2 Change 4](#_Toc47685836)

[2.1.3 Copy 4](#_Toc47685837)

[2.1.4 Delete 5](#_Toc47685838)

[2.1.5 Move 5](#_Toc47685839)

[2.1.6 Rename 6](#_Toc47685840)

[2.2 Regole di utilità 6](#_Toc47685841)

[2.2.1 Path 6](#_Toc47685842)

[2.2.2 Repeat 6](#_Toc47685843)

[2.3 Struttura di un file delle regole 7](#_Toc47685844)

[2.3.1 Schema XSD di un file delle regole 7](#_Toc47685845)

# Scopo del documento

Il presente documento ha l’obiettivo di definire le tipologie di regole attualmente supportate dall’Editor delle regole e dal Modulo di trasformazione. In questo documento verrà inoltre riportata la struttura di un file di regole affinché quest’ultimo risulti valido.

# Tipologie di regole

Sia l’Editor delle regole che il Modulo di trasformazione supportano 7 tipologie di regole suddivise in 2 categorie. Esse sono suddivise in:

* Regole di trasformazione
* Regole di utilità

Ogni regola corrisponde ad un elemento XML. Nella [sezione 2.4](#_Schema_dei_tipi) viene riportato lo schema XSD di entrambe le categorie.

## Regole di trasformazione

Le regole di trasformazione permettono di manipolare direttamente elementi, attributi e namespace contenuti nel tracciato XML a cui fanno riferimento.

### Add

La regola *add* permette di aggiungere un elemento, un attributo o un namespace ad un elemento del tracciato XML. A questa regola possono essere associati gli attributi:

* **type**: contiene il tipo di nodo da aggiungere. I valori ammessi sono:
  + element (impostato di default): per aggiungere un elemento
  + attribute: per aggiungere un attributo
  + namespace: per aggiungere un namespace
* **name**: contiene il nome del nodo da inserire.
* **in**: contiene il percorso XPath nel quale aggiungere il nodo.
* **where**: contiene un’espressione XPath che funge da filtro di selezione per l’attributo **in**.
* **before**: permette di inserire il nuovo nodo prima del nodo specificato da questo attributo.
* **after**: permette di inserire il nuovo nodo dopo il nodo specificato da questo attributo.
* **if**: contiene un’espressione XPath booleana. La regola viene applicata solo in caso di esito positivo.

Gli attributi obbligatori sono: **name** e **in**. Il valore associato a questa regola, se presente, sarà il valore del nodo inserito.

### Change

La regola *change* permette di modificare il valore di un elemento, un attributo o un namespace contenuto nel tracciato XML. A questa regola possono essere associati gli attributi:

* **type**: contiene il tipo di nodo da modificare. I valori ammessi sono:
  + element (impostato di default): per modificare il valore di un elemento
  + attribute: per modificare il valore di un attributo
  + namespace: per modificare il valore di un namespace
* **name**: contiene il nome del nodo da modificare.
* **in**: contiene il percorso XPath del nodo da modificare.
* **where**: contiene un’espressione XPath che funge da filtro di selezione per l’attributo **in**.
* **if**: contiene un’espressione XPath booleana. La regola viene applicata solo in caso di esito positivo.

Gli attributi obbligatori sono: **name** e **in**. Il valore associato a questa regola è obbligatorio e sarà il nuovo valore del nodo.

### Copy

La regola *copy* permette di copiare un elemento, un attributo o un namespace contenuto nel tracciato XML, in un elemento dello stesso tracciato. A questa regola possono essere associati gli attributi:

* **type**: contiene il tipo di nodo da copiare. I valori ammessi sono:
  + element (impostato di default): per copiare un elemento
  + attribute: per copiare un attributo
  + namespace: per copiare un namespace
* **name**: contiene il nome del nodo da copiare.
* **from**: contiene il percorso XPath del nodo da copiare.
* **fromWhere**: contiene un’espressione XPath che funge da filtro di selezione per l’attributo **from**.
* **to**: contiene il percorso XPath in cui andrà inserita la copia del nodo.
* **toWhere**: contiene un’espressione XPath che funge da filtro di selezione per l’attributo **to**.
* **before**: permette di inserire la copia del nodo prima del nodo specificato da questo attributo.
* **after**: permette di inserire copia del nodo dopo il nodo specificato da questo attributo.
* **if**: contiene un’espressione XPath booleana. La regola viene applicata solo in caso di esito positivo.

Gli attributi obbligatori sono: **name**, **from** e **to**. La regola non può avere un valore.

### Delete

La regola *delete* permette di eliminare un elemento, un attributo o un namespace contenuto nel tracciato XML. A questa regola possono essere associati gli attributi:

* **type**: contiene il tipo di nodo da eliminare. I valori ammessi sono:
  + element (impostato di default): per eliminare un elemento
  + attribute: per eliminare un attributo
  + namespace: per eliminare un namespace
* **name**: contiene il nome del nodo da eliminare.
* **in**: contiene il percorso XPath del nodo da eliminare.
* **where**: contiene un’espressione XPath che funge da filtro di selezione per l’attributo **in**.
* **if**: contiene un’espressione XPath booleana. La regola viene applicata solo in caso di esito positivo.

Gli attributi obbligatori sono: **name** e **in**. La regola non può avere un valore.

### Move

La regola *move* permette di spostare un elemento, un attributo o un namespace contenuto nel tracciato XML, in un elemento dello stesso tracciato. A questa regola possono essere associati gli attributi:

* **type**: contiene il tipo di nodo da spostare. I valori ammessi sono:
  + element (impostato di default): per spostare un elemento
  + attribute: per spostare un attributo
  + namespace: per spostare un namespace
* **name**: contiene il nome del nodo da spostare.
* **from**: contiene il percorso XPath del nodo da spostare.
* **fromWhere**: contiene un’espressione XPath che funge da filtro di selezione per l’attributo **from**.
* **to**: contiene il nuovo percorso XPath in cui andrà inserita il nodo.
* **toWhere**: contiene un’espressione XPath che funge da filtro di selezione per l’attributo **to**.
* **before**: permette di inserire il nodo prima del nodo specificato da questo attributo.
* **after**: permette di inserire il nodo dopo il nodo specificato da questo attributo.
* **if**: contiene un’espressione XPath booleana. La regola viene applicata solo in caso di esito positivo.

Gli attributi obbligatori sono: **name**, **from** e **to**. La regola non può avere un valore.

### Rename

La regola *rename* permette di rinominare un elemento, un attributo o un namespace contenuto nel tracciato XML. A questa regola possono essere associati gli attributi:

* **type**: contiene il tipo di nodo da rinominare. I valori ammessi sono:
  + element (impostato di default): per rinominare un elemento
  + attribute: per rinominare un attributo
  + namespace: per rinominare un namespace
* **name**: contiene il nome del nodo da rinominare.
* **in**: contiene il percorso XPath del nodo da rinominare.
* **where**: contiene un’espressione XPath che funge da filtro di selezione per l’attributo **in**.
* **if**: contiene un’espressione XPath booleana. La regola viene applicata solo in caso di esito positivo.

Gli attributi obbligatori sono: **name** e **in**. Il valore associato a questa regola è obbligatorio e sarà il nuovo nome del nodo.

## Regole di utilità

Le regole di utilità fungono da supporto alle regole di trasformazione permettendo di semplificarne la definizione. Non apportano alcuna modifica diretta al tracciato XML a cui si riferiscono.

### Path

La regola *path* permette di definire il percorso XPath di partenza a cui faranno riferimento tutte le regole definite successivamente. Questa regola non può avere valore e deve contenere unicamente l’attributo **current** in cui sarà riportato il percorso XPath di riferimento.

### Repeat

La regola *repeat* permette di ripetere più volte una sequenza di regole. Questa regola può contenere un qualsiasi numero (diverso da 0) di regole di qualsiasi tipo diverso da *repeat* e deve avere un unico attributo **times** contenente il numero di ripetizioni desiderate. Il numero di ripetizioni può essere esplicito o in forma di espressione XPath.

## Struttura di un file delle regole

Un file di regole può essere composto sia da regole di trasformazione che da regole di utilità racchiuse in un elemento *rules*. Tale elemento può avere l’attributo **defaultPrefix** che specifica un prefisso da utilizzare per riferirsi ad elementi appartenenti al namespace xmlns senza prefisso.

### Schema XSD di un file delle regole

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

<xs:element name="rules" type="RulesType"/>

<xs:complexType name="RulesType">

<xs:sequence>

<xs:choice minOccurs="0" maxOccurs="unbounded">

<xs:element name="path" type="PathType"/>

<xs:element name="repeat" type="RepeatType"/>

<xs:element name="add" type="AddType"/>

<xs:element name="change" type="ChangeType"/>

<xs:element name="copy" type="CopyType"/>

<xs:element name="delete" type="DeleteType"/>

<xs:element name="move" type="MoveType"/>

<xs:element name="rename" type="RenameType"/>

</xs:choice>

</xs:sequence>

<xs:attribute name="defaultPrefix" type="NotEmptyString"/>

</xs:complexType>

<xs:complexType name="PathType">

<xs:attribute name="current" type="NotEmptyString" use="required"/>

</xs:complexType>

<xs:complexType name="RepeatType">

<xs:sequence>

<xs:choice maxOccurs="unbounded">

<xs:element name="path" type="PathType"/>

<xs:element name="add" type="AddType"/>

<xs:element name="change" type="ChangeType"/>

<xs:element name="copy" type="CopyType"/>

<xs:element name="delete" type="DeleteType"/>

<xs:element name="move" type="MoveType"/>

<xs:element name="rename" type="RenameType"/>

</xs:choice>

</xs:sequence>

<xs:attribute name="times" type="NotEmptyString" use="required"/>

</xs:complexType>

<xs:complexType name="AddType">

<xs:simpleContent>

<xs:extension base="xs:string">

<xs:attribute name="type" type="TypeEnum" default="element"/>

<xs:attribute name="name" type="NotEmptyString" use="required"/>

<xs:attribute name="in" type="NotEmptyString" use="required"/>

<xs:attribute name="where" type="NotEmptyString"/>

<xs:attribute name="before" type="NotEmptyString"/>

<xs:attribute name="after" type="NotEmptyString"/>

<xs:attribute name="if" type="NotEmptyString"/>

</xs:extension>

</xs:simpleContent>

</xs:complexType>

<xs:complexType name="ChangeType">

<xs:simpleContent>

<xs:extension base="NotEmptyString">

<xs:attribute name="type" type="TypeEnum" default="element"/>

<xs:attribute name="name" type="NotEmptyString" use="required"/>

<xs:attribute name="in" type="NotEmptyString" use="required"/>

<xs:attribute name="where" type="NotEmptyString"/>

<xs:attribute name="if" type="NotEmptyString"/>

</xs:extension>

</xs:simpleContent>

</xs:complexType>

<xs:complexType name="CopyType">

<xs:attribute name="type" type="TypeEnum" default="element"/>

<xs:attribute name="name" type="NotEmptyString" use="required"/>

<xs:attribute name="from" type="NotEmptyString" use="required"/>

<xs:attribute name="fromWhere" type="NotEmptyString"/>

<xs:attribute name="to" type="NotEmptyString" use="required"/>

<xs:attribute name="toWhere" type="NotEmptyString"/>

<xs:attribute name="before" type="NotEmptyString"/>

<xs:attribute name="after" type="NotEmptyString"/>

<xs:attribute name="if" type="NotEmptyString"/>

</xs:complexType>

<xs:complexType name="DeleteType">

<xs:attribute name="type" type="NotEmptyString" default="element"/>

<xs:attribute name="name" type="NotEmptyString" use="required"/>

<xs:attribute name="in" type="NotEmptyString" use="required"/>

<xs:attribute name="where" type="NotEmptyString"/>

<xs:attribute name="if" type="NotEmptyString"/>

</xs:complexType>

<xs:complexType name="MoveType">

<xs:attribute name="type" type="TypeEnum" default="element"/>

<xs:attribute name="name" type="NotEmptyString" use="required"/>

<xs:attribute name="from" type="NotEmptyString" use="required"/>

<xs:attribute name="fromWhere" type="NotEmptyString"/>

<xs:attribute name="to" type="NotEmptyString" use="required"/>

<xs:attribute name="toWhere" type="NotEmptyString"/>

<xs:attribute name="before" type="NotEmptyString"/>

<xs:attribute name="after" type="NotEmptyString"/>

<xs:attribute name="if" type="NotEmptyString"/>

</xs:complexType>

<xs:complexType name="RenameType">

<xs:simpleContent>

<xs:extension base="NotEmptyString">

<xs:attribute name="type" type="TypeEnum" default="element"/>

<xs:attribute name="name" type="NotEmptyString" use="required"/>

<xs:attribute name="in" type="NotEmptyString" use="required"/>

<xs:attribute name="where" type="NotEmptyString"/>

<xs:attribute name="if" type="NotEmptyString"/>

</xs:extension>

</xs:simpleContent>

</xs:complexType>

<xs:simpleType name="TypeEnum">

<xs:restriction base="xs:string">

<xs:enumeration value="element"/>

<xs:enumeration value="attribute"/>

<xs:enumeration value="namespace"/>

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="NotEmptyString">

<xs:restriction base="xs:string">

<xs:minLength value="1" />

</xs:restriction>

</xs:simpleType>

</xs:schema>