Multi-language package

# Overview

The multi-language package describes the support to replace text in XML based resource such as feature catalogues or portrayal catalogues. It is a generic concept that will work for any resource if it is XML. Instead of having the support inside the XML resources here the language support will be achieved by separate files that supply the language support for one language. That makes the maintenance much easier since there is not one central resource to maintain but several language package. Each of those language packages can be independently maintained by experts with the knowledge both on the domain and on the language.

The main idea behind this concept is to create and use so called language packs that

1. identify all elements in the resource to be translated
2. contains the original text in the language of the resource (usually English but not limited to it)
3. contains the translated test
4. hold some status information on each item

# Model

The model of a language pack is described by the following diagram.

Diagram

Description automatically generated

The ***TranslationPackageType*** consists of a minimal set of header information and a sequence of source files elements. Each of the ***SourceFileType*** elements has a header to identify the resource and a set of items that hold the information for one text element in the resource to be translated.

The details are in the following tables. Note that most classes exist in a namespace **S100\_LA**. This prefix is not part of the class names but logically belongs to them. Exceptions are marked in the tables.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Class | TranslationPackageType | The main element of a language package. | - | - |  |
| Role | language | The language in that the translations are made. |  | LanguageType | The type contains an attribute ***language*** that hold language codes according to ISO639-2/T (e.g deu, fra) |
| Attribute | issueDate | The date when the language pack is issued | 1 | Date |  |
| Attribute | issueTime | The time when the language pack is issued | 0..1 | Time |  |
| Attribute | responsibleParty | Meta information about the responsible organisation or individual | 0..\* | CI\_ResponsibleParty | This type conforms to ISO 19115 and is not defined in the namespace ***S100\_LA.***  There may be more than one parties responsible which then have different roles. (e.g, custodian, translator, or publisher) |
| Role | sourceFile | Holding all translation items for one source file | 1..\* | SourceFileType |  |

**TranslationPackageType**

**SourceFileType**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Class | SourceFileType | An element holding all information on elements to be translated of a single source file. | - | - | The source file must be in XML format |
| Role | header | Information to identify the source file. | 1 | SourceHeaderType |  |
| Role | translationItem | A list of items each describes an element of the source file the is subject to translation. | 0..\* | TranslationItemType |  |

**SourceHeaderType**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Class | SourceHeaderType | The information to identify the source file for that this translation package contains the translations | - | - |  |
| Attribute | resourceIdentification | The identifier of the resource | 1 | URI | This can be either a file URI or a hash URN.  Example:  file:///somefile.xml |
| Role | identification | Information on version information inside the source file | 0..\* | ResourceIdentification |  |

**ResourceIdentification**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Class | ResourceIdentification | Information where a version information can be found in the source (XML) file and what value is stored there. | - | - |  |
| Attribute | path | An XPath expression to the element or attribute that contains the identification information | 1 | CharacterString | e.g. /S100FC:S100\_FC\_FeatureCatalogue/S100FC:versionNumber |
| Attribute | value | The value that must be in the specified element to match the source file. | 1 | CharacterString | e.g.  1.0.0 |

**TranslationItemType**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Mult** | **Type** | **Remarks** |
| Class | TranslationItemType | Information on a single translatable item in the source file. | - | - |  |
| Attribute | path | An XPath expression to the element or attribute that contains the text to be translated | 1 | CharacterString |  |
| Attribute | original | The text as it appears in the source file | 0..1 | CharacterString |  |
| Attribute | status | The status of the item | 0..1 | Status |  |
| Attribute | translation | The translated text. | 1 | CharacterString |  |

**Status**

|  |  |  |  |
| --- | --- | --- | --- |
| **Role Name** | **Name** | **Description** | **Remarks** |
| Enumeration | Status | The status of the translation item. | The purpose is to support the life cycle of each translation item. |
| Literal | New | The item is not yet translated. |  |
| Literal | Modified | The content of the item has been changed in the source file. | The translation must be revisited because it may not be valid anymore. |
| Literal | Deleted | The item does not exist in the source file anymore. |  |
| Literal | Translated | The item is translated and ready to use. |  |

**How it works:**

The ***path*** attribute in an instance of a ***TranslationItemType*** will point to either an element or an attribute in the source file. Technically an XPath expression is used for that purpose. The expression must uniquely define one element, hence a query on the XML DOM tree must return exactly one node.

**Example:**

<S100FC:S100\_FC\_FeatureCatalogue xmlns:S100FC=<http://www.iho.int/S100FC> …  
…  
 <S100FC:S100\_FC\_SimpleAttributes>  
 <S100FC:S100\_FC\_SimpleAttribute>  
 <S100FC:name>Category of Topping</S100FC:name>  
 <S100FC:definition>Topping on a pizza (a maximum of four)</S100FC:definition>  
 <S100FC:code>categoryOfTopping</S100FC:code>

A simplified tree structure of the XML file would look like:

The XPath expression:

/S100FC:S100\_FC\_FeatureCatalogue/S100FC:S100\_FC\_SimpleAttributes/S100FC:S100\_FC\_SimpleAttribute[./S100FC:code/text()='categoryOfTopping']/S100FC:name

would specify the marked element. It is looking for an elements with the name ***S100FC:S100\_FC\_Simple\_Attribute*** with parent ***S100FC:S100\_FCSimpleAttributes*** and then ***S100FC:S100\_FC\_FeatureCatalog*** that has a child ***S100FC:code*** with a text of “*categoryOfTopping*”. Then the child **S100FC:name** is selected.

In the case of more elements have the same code the Xpath expression would be ambiguous. In feature catalogues it is ensured that the codes are unique.

# Schema

The schema is defined in the namespace:

xmlns="<http://www.iho.int/s100/la>"

It uses types from two ISO 19115 schemas

xmlns:gco="http://standards.iso.org/iso/19115/-3/gco/1.0"

xmlns:cit="<http://standards.iso.org/iso/19115/-3/cit/2.0>"

The following settings are made for this schema

targetNamespace="http://www.iho.int/S100/la"

elementFormDefault="qualified"

attributeFormDefault="unqualified"

In the schema the type **Status** is implemented as a simple type as follows:

<xs:simpleType name="Status">

<xs:annotation>

<xs:documentation>

The status of the translation item. The purpose is manly to support the  
 functionality of an translation tool.

</xs:documentation>

</xs:annotation>

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

<xs:enumeration value="New">

<xs:annotation>

<xs:documentation>

The item is new, there is no translation available yet.

</xs:documentation>

</xs:annotation>

</xs:enumeration>

<xs:enumeration value="Modified">

<xs:annotation>

<xs:documentation>

The original text has been changed in the source

</xs:documentation>

</xs:annotation>

</xs:enumeration>

<xs:enumeration value="Deleted">

<xs:annotation>

<xs:documentation>

The text defined by path is not longer available in the source  
 document.  
 </xs:documentation>

</xs:annotation>

</xs:enumeration>

<xs:enumeration value="Translated">

<xs:annotation>

<xs:documentation>

The item has a valid translation.

</xs:documentation>

</xs:annotation>

</xs:enumeration>

</xs:restriction>

</xs:simpleType>

For the definition of the language a complex type **LanguageType** is defined that is using an element of the type **gco:CodeListValue\_Type** (from ISO 19115)

<xs:complexType name="LanguageType">

<xs:sequence>

<xs:element name="languageCode" type="gco:CodeListValue\_Type"/>

</xs:sequence>

</xs:complexType>

Each translation item is defined by the type **TranslationItemType**:

<xs:complexType name="TranslationItemType">

<xs:annotation>

<xs:documentation>

One item to be translated. This will be a uniquely identifiable element or

attribute in a source (XML) file

</xs:documentation>

</xs:annotation>

<xs:sequence>

<xs:element name="path" type="xs:string">

<xs:annotation>

<xs:documentation>

The XPath that defines the 'source' text.

</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="original" type="xs:string" minOccurs="0">

<xs:annotation>

<xs:documentation>

The original text as exists in the source document.

</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="status" type="Status" minOccurs="0">

<xs:annotation>

<xs:documentation>

The status of the translation item

</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="translation" type="xs:string">

<xs:annotation>

<xs:documentation>

The translated text.

</xs:documentation>

</xs:annotation>

</xs:element>

</xs:sequence>

</xs:complexType>

The type **ResourceIdentification** contains the information to identify a source file e.g. by defining the XPath to the version element and the value that this element in the source file must have.

<xs:complexType name="ResourceIdentification">

<xs:annotation>

<xs:documentation>

Information to identify a specific version of an (XML) file.

</xs:documentation>

</xs:annotation>

<xs:sequence>

<xs:element name="path" type="xs:string">

<xs:annotation>

<xs:documentation>

The XPath to the element or attribute that allows the unique identification

of the source file.

</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="value" type="xs:string">

<xs:annotation>

<xs:documentation>

The value of the element or attribute that describes the identification of

the source file. e.g. the version or issue date

</xs:documentation>

</xs:annotation>

</xs:element>

</xs:sequence>

</xs:complexType>

The type **SourceHeaderType** contains the information on the source file as the filename and an identification mechanism.

<xs:complexType name="SourceHeaderType">

<xs:annotation>

<xs:documentation>

Information to identyfing the source file. It supports mechanism to distinguish

different versions of a source file.

</xs:documentation>

</xs:annotation>

<xs:sequence>

<xs:element name="resourceIdentifier" type="xs:anyURI"/>

<xs:element name="identification" type="ResourceIdentification" minOccurs="0"

maxOccurs="unbounded">

<xs:annotation>

<xs:documentation>

Identification by one or more elements or attributes in the source

file.

</xs:documentation>

</xs:annotation>

</xs:element>

</xs:sequence>

</xs:complexType>

The type **SourceFileType** contains the header information of the file and a list of translation items for that file.

<xs:complexType name="SourceFileType">

<xs:sequence>

<xs:element name="header" type="SourceHeaderType"/>

<xs:element name="translationItem" type="TranslationItemType" minOccurs="0"

maxOccurs="unbounded">

<xs:annotation>

<xs:documentation>

The list of translation items.

</xs:documentation>

</xs:annotation>

</xs:element>

</xs:sequence>

</xs:complexType>

The last type defined by the schema is the type **TranslationPackageFile**

<xs:complexType name="TranslationPackageType">

<xs:sequence>

<xs:element name="language" type="LanguageType">

<xs:annotation>

<xs:documentation>

The language that will be supported by this translation

file.

</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element name="issueDate" type="xs:date"/>

<xs:element name="issueTime" type="xs:time" minOccurs="0"/>

<xs:element name="responsibleParty" type="cit:CI\_Responsibility\_PropertyType"   
 minOccurs="0" maxOccurs="unbounded"/>

<xs:element name="sourceFile" type="SourceFileType" maxOccurs="unbounded">

<xs:annotation>

<xs:documentation>

The list of source files for that this translation file provides

translations.

</xs:documentation>

</xs:annotation>

</xs:element>

</xs:sequence>

</xs:complexType>

this type defines some elements for the metadata of the translation file as

* language
* issue date
* issue time
* and responsible parties

and a list of source files with their translations.

Note that for the responsible party the type **cit:CI\_Responsibility\_PropertyType** is used.

A possible encoding would look like:

<S100LA:responsibleParty>

<cit:CI\_Responsibility>

<cit:role>

<cit:CI\_RoleCode codeList="codeListLocation#CI\_RoleCode"

codeListValue="custodian">custodian</cit:CI\_RoleCode>

</cit:role>

<cit:party>

<cit:CI\_Individual>

<cit:name>

<gco:CharacterString>Max Mustermann</gco:CharacterString>

</cit:name>

</cit:CI\_Individual>

</cit:party>

</cit:CI\_Responsibility>

</S100LA:responsibleParty>

Alternative to the element <cit:CI\_Individual> the element <cit:CI\_Organisation> can be used in case the producer is an organisation rather than an individual.

Note that the predefined list of roles for a responsible party are:

* *resourceProvider*
* *custodian*
* *owner*
* *user*
* *distributor*
* *originator*
* *pointOfContact*
* *principalInvestigator*
* *processor*
* *publisher*
* *author*
* *sponsor*
* *coAuthor*
* *collaborator*
* *editor*
* *mediator*
* *rightsHolder*
* *contributor*
* *funder*
* *stakeholder*

There is no value for ‘*translator’*. ‘*contributor’* could be used for this role if required.

Finally, the schema defines the root element of a translation file.

<xs:element name="translationPackage" type="TranslationPackageType">

<xs:annotation>

<xs:documentation>

The root element of a translation file.

</xs:documentation>

</xs:annotation>

</xs:element>

# Processes

In this chapter some processes are described for the life cycle of a translation file. The process includes:

* Creation
* Maintenance
* Deployment
* Usage of the information

Without loss of generality here only one source file is considered.

## Creating of a translation file

To create a translation file for a given source file at first the items in the source file that are subject to translation must be defined. Then the source file must be investigated and for each item (either an element or an attribute) a *translationItem* in the translation file will be created. It will contain the XPath expression to the item in the source file. The original text is copied into this *translationItem*, the status is set to ‘*New’*, and an empty element for the translated text is provided.

In addition to this main task all the meta information that are required should be created including the file identification.

Technically an XSLT script can be used for this purpose, but the process can be implemented in different ways.

## Language Translation

This is the process to translate the text of each item. The Status information can be used to iterate only over items that are either ***New*** or ***Modified***. This can save a lot of time when the translation is done again after the source has been changed e.g. because of a new version. For each item that is translated the Status will be set to ***Translated.***

## Updating of a translation file

This process will update a translation file. When the source file has been changed the following algorithm can be used to create an updated translation file.

1. Given a Source S and an existing translation file T
2. Create a new translation file NT from S as described in the ‘Creation of a translation file’ section
3. Comparing T and NT as follows:
   1. For all items that are in NT but not in T add an item in T with Status=***New***
   2. For all items that are in T but not in NT set the status of the item in T to ***Deleted***.
   3. For all items that exists both in NT and T but with different original text copy the text from NT to T and set the status in T to ***Modified***.
4. NT can be deleted
5. If at least one item in T has the Status set to ***New*** or ***Modified*** the ‘Language Translation’ process must be repeated

## Deployment

If the translation file has no items with Status set to ***New*** or ***Modified*** it is ready to be used.

Since some information are not required by the final use process this information can be removed in the deployed version.

This information includes

* the original text
* the status
* all items with the Status=Deleted

Note the removal is only done in a copy that should be removed. The removal should never be made on the ‘Master’ translation file because after the removal the file can not be used for the ‘Updating of a translation file’ process.

## Use of the translation file

The use of the translation files is possible in different ways.

1. Create a translated copy of the resource e.g. a Feature catalogue for the supported language and use this resource in the final application.
2. Store the translation information separate to the resource and implement an API that returns the appropriate text for a selected language or if the information is not available the text from the original resource.