eHealth Framework

How to migrate the codesystem repository to the redesigned structure

Imprint

InterComponentWare
Altrottstraße 31
69190 Walldorf
Tel.: +49 (0) 6227 385 0
Fax.: +49 (0) 6227 385 199

Document version: preliminary
Document Language: en (US)
Product Name: eHealth Framework
Product Version: 2.10.3
Last Change: 23.03.2010
Editorial Staff: BAS Technology

Notice

The wording in this document applies equally to women and men. The masculine form was selected to ease the comprehensibility and legibility of the text.

Note:

The current version of this document has a draft status and various chapters are still in review.

The document is collaboratively built with the use of the Darwin-Information-Typing-Architecture (DITA) and has therefore a draft status concerning styles and layout. The necessary adaptations are currently also in a developmental stage.

Contents

1 Overview

Purpose

In eHF 2.10, the import processor was completely redesigned, in order to fulfill several requirements like:

- enable incremental updates
- consideration of terminology changes at runtime
- import of terms in additional languages
- support of a default value like a concept value for each code
- logging of import results and messages
- schema definition ensures more robust error checking
- schema definition ensures separation of the various aspects of terminology definiton and configuration

In addition, the import processor is prepared for additional features like: loading of terminology from several locations and support of several format.

The migration is supported by a migration tool that provides a smooth migration from the previous repository data to the new format.

This HOW-TO explains the steps required for migrating the codesystem and codeset definition and the configuration of the import processor.

2 Migration of the CodeSystem Repository

2.1 Overview of what has changed

In the context of the redesign of the import processor, the code system and code set definition as well as the import configuration was changed. The main changes are listed below grouped by the concepts. Be aware that the concepts "Catalog", "Codesystem Contribution" and "CodeTerms" are new. For detailled information please see the reference documentation."

CodeSystem

- The codesystem definition is now divided in the codesystem contribution and the code terms definition. So, the translation is separated from the concept definition.
- Each code has a concept value that represents a human understandable engineering text of the related concept.
- The codesystem definition could be broken down into multiple files, each being called a `contribution`.
- The previous `MASTER` and `EXTENSION` codesystem definition could be achieved using the `contribution`.

CodeTerms

- A single file could contain the translated terms for the concept codes of a code system for several locales, or the translated terms for each locale could be in a separate file.

CodeSet

- A codeset does not require a version. The version is generated by the import processor
- A codeset could define an explicit list of codes, or use regular expressions to define the collection of codes that it should be included.
- To define a semantic order, the codes could be listed in the correct order within the `ordered` block.

Catalog

- The terminology catalog represents the published code concepts (code system contribution, code set, code terms) of a codesystem repository for each terminology revision. It is possible to define more than one terminology catalog for a codesystem repository.

Import configuration

- The import configuration lists all the catalogs that are required.
- The import configuration lists all the concepts (codesystems, codesets, and codeterms) that should be imported
- The import configuration lists all the code categories and the assigned code sets.
- The import configuration supports an entire or incremental import of the code concepts. So, it is possible to import addtional languages or code system extensions in a separate import step.

2.2 Preparation

To perform the migration, please use the repository migration tool.

This task shows you the steps required to setup the migration tool.

This requires the repository-migration tool, which you could download from TODO:location.

Since the repository-migration tool is written in Java, it requires a Java Runtime Environment.

1. Extract the ehf-repository-migration-tool.zip to a local directory.

   The content of the folder looks as shown in the figure below:

   The `lib` folder contains all the required `jar` files.

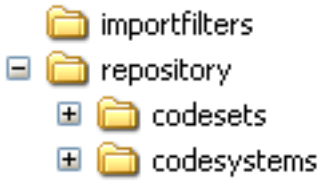
   The `config.properties` is a text file containing the configurable parameters and their values for the migration tool.

   The `migrate.bat` is the MS-Windows batch file that triggers the migration process for the MS-Windows environment. This should be executed from the MS-Windows command prompt.

   The `migrate.sh` is the Unix script file for starting the migration process for the Unix environment. This should be executed from the Unix or equivalent shell prompt.

2. Prepare the old repository

   The tool expects the old repository to have the following directory structure (by default the tool uses the directory names as shown in the figure.):



   The folder `codesystems` could have subfolders and should contain all the codesystem definitions, and the folder `codesets` could also have subfolders and should contain all the codeset definitions.

   The `importfilters` folder should not be a parent folder of `codesystems` or `codesets`. It should only contain the import filter XML files.

   The tool could be configured to specify a different location for the `importfilters` and `repository`. The default names of the folders could also be changed.

   Please refer to the section configuration on page 3 for more information.

2.3 Configuring the tool

The tool looks for a config file with the name `config.properties` in the current directory. This file provides the tool with information such as the location of the repository, the import filters, output folder and few other configurable parameters.

Sample config file

A sample config file would look like the one below:

```
repositoryPath=C:/repository/repository
importFilterPath=C:/repository/importfilters
outputFolder=C:/new-repository
conceptLocales=en,de
```

Attention:

Please note that all the paths MUST NOT have any trailing path separator character. The path separator character could be Unix style ('/') or if Windows style is preferred, please use ('\\').

The mandatory configuration parameters are as follows.

Name	Description
repositoryPath	Specifies the location of the repositories.
importFilterPath	Specifies the location of the import filters.
outputFolder	Specifies the location for the output. The tool would create the same directory structure as the old one.
conceptLocales	A sequence of comma separated locales, that should be used as the concept value. If for some codes, there is no value for the first locale, it would use the next locale and so on. If none of the locales are available, it would print a warning. The concept value would be set to `FIXME` string for such codes.

Table1. Mandatory configuration parameters

Other configurable parameters are as follows.

Name	Default	Description
includeLocales	*	A sequence of comma separated locales, that should be included in the output. This is in addition to the locales specified by the conceptLocales. A `'*'` means include all locales that are available.
catalogFile	catalog.xml	The name for the catalog file.
codesystemsFold	codesystems	The name of the folder within the `repositoryPath` that contains all the codesystems.
codesetsFolder	codesets	The name of the folder within the `repositoryPath` that contains all the codesets.
codeTermSuffix	TERM	All the translations would be moved to a separate file with the same name as the codesystem file, and suffixed with this word. (eg. for codesystem `loinc.xml` , the translation file would be called `loinc_TERM.xml`)

Table2. Other configuration parameters

2.4 Executing the tool

Once the config file is set up correctly, executing the tool is really easy.

To execute the migration tool, open a command prompt, and change to the base directory of the tool and execute the batch file (`migrate.bat`).

This command loads the config file, and recursively go through each xml file in the specified repository folder, and convert them to the new format. The new files are stored in the folder specified by the `outputFolder` config parameter.

The migration tool might generate various error and warning messages (to the console) as it processes the files. These errors or warning messages should be reviewed and the problems fixed, to avoid import failures later.

The following sections lists the possible errors/warnings that you might get, and the measures needed to fix them.

Possible error/warning messages:

Message	Cause	Solution
ERROR: status is 'draft' for 2.16.840.1.113883.3.37.1.9.23.3 (CoveredPartyCH), would not be imported.	The importer would only import the codesystems that are marked as `final`.	Change the status to `final` or exclude this codesystem from the terminology config file.
ERROR: No concept values found for oid: 2.16.840.1.113883.3.37.1.9.17.2 (RegionCodesRU)	The new importer requires a default value (also known as the concept value) for every Code.	Add the concept value for this codesystem in the generated file, or add an additional (existing) locale to the config parameter `conceptLocales`.
WARN: '5.7' key missing for 2.16.840.1.113883.3.37.1.9.10.1 (DocumentCode):de	The key `5.7` does not have a value for the given locale, but it has value for some other locales.	Add a value for the locale in generated file. If the value is not provided, the new codesystem module would fallback to the concept value for this locale. If the concept value is missing for a key, the word `FIXME` would be added as the concept value for the key.

Table3. Possible error/warning messages

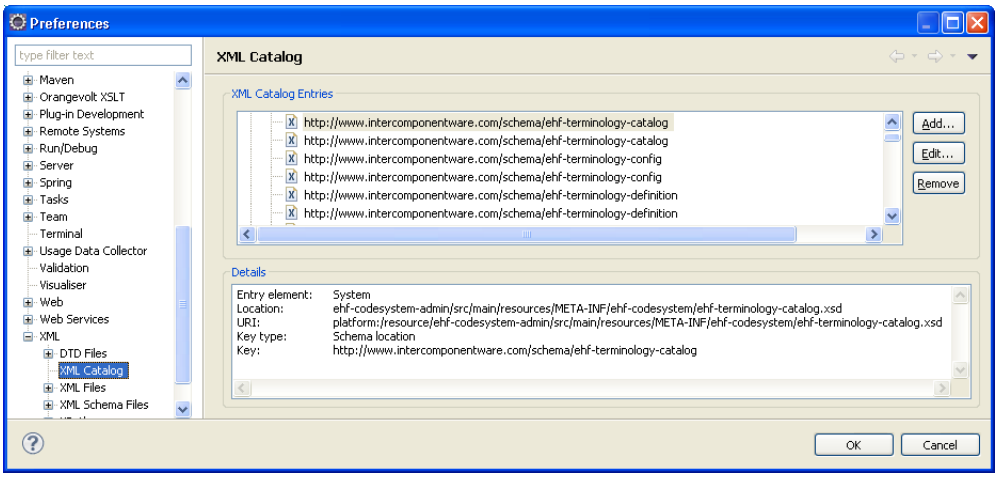
2.5 Validation of the generated repository

This task shows you how to validate the generated codesystem repository, using Eclipse Helios IDE.

This requires Eclipse Helios to be setup to locate the new codesystem XSD files as described below.

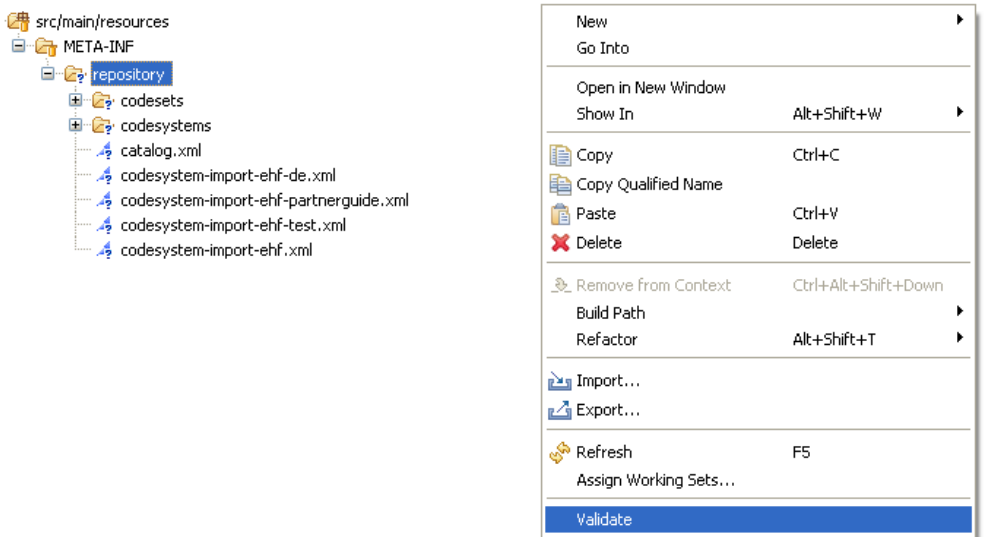
1. Open the Eclipse Helios menu option: *`Window->Preferences->XML->XML Catalog`*.
   This would open the `XML Catalog` dialog box.

2. Add the schema locations for `ehf-terminology-config`, `ehf-terminology-catalog` and `ehf-terminology-definition` as shown in the figure below:



   a. Press `OK` button to return to the editor mode.

3. Right click on the repository folder in the `Package Explorer` on the left hand pane and select `Validate` from the pop-up menu as shown in the figure below:



All the XML schema violations would be highlighted.

2.6 Known issues

Some known limitations of the migration tool and their work-arounds are listed below:

- **`include` tag in previous import filters**

  The previous import filter provided an option to specify a wild card to include all the other import filters. This is not longer supported.

  The migration tool would therefore migrate each import filter into separate config files with the same name as that of the import filter. These config files could then be listed in the import configuration file in the correct order.

- **Migration of codesystems defined in several files**

  The old codesystem provided an option to extend a codesystem with the attribute `type=EXTENSION`.

The creation of the catalog file for such a codesystem results in a duplicate entry, which needs to be fixed by hand.

For example the codesystem `Charset` has the `MASTER` definition in the file named `Charset.xml` and an `EXTENSION` for Spanish language in a file called `Charset_es.xml`

The migration tool would generate the the following files:

- `Charset.xml`: The codesystem definition
- `Charset_TERM.xml`: All the translation available in the master definition file (en, de)
- `Charset_es_TERM.xml`: The spanish translation

But the catalog file would have duplicate entries for the codesystem `Charset` as shown in the code below:

```
   <terminology oid="2.16.840.1.113883.5.21">
      <codesystem>
         <contribution>
            <resource path="codesystems/document" filename="Charset.xml"/>
         </contribution>
      </codesystem>
      <codesets>
         <codeset name="EXT-GEN-DOCUMENT-CHARSET">
            <resource path="codesets/document" filename="EXT-GEN-DOCUMENT-
CHARSET.xml"/>
         </codeset>
      </codesets>
      <codeterms path="codesystems/document" filename="Charset_es_TERM.xml">
         <codeterm locale="de"/>
         <codeterm locale="en"/>
         <codeterm locale="es"/>
      </codeterms>
   </terminology>
   <terminology oid="2.16.840.1.113883.5.21">
      <codesystem>
         <contribution>
            <resource path="codesystems/document" filename="Charset_es.xml"/>
         </contribution>
      </codesystem>
      <codesets>
         <codeset name="EXT-GEN-DOCUMENT-CHARSET">
            <resource path="codesets/document" filename="EXT-GEN-DOCUMENT-
CHARSET.xml"/>
         </codeset>
      </codesets>
      <codeterms path="codesystems/document" filename="Charset_es_TERM.xml">
         <codeterm locale="de"/>
         <codeterm locale="en"/>
         <codeterm locale="es"/>
      </codeterms>
   </terminology>
```

Please note that there is a duplicate `terminology` block. And both blocks point to the `Charset_es_TERM.xml` codeterm file.

It could be fixed as shown below:

```
   <terminology oid="2.16.840.1.113883.5.21">
      <codesystem>
         <contribution>
            <resource path="codesystems/document" filename="Charset.xml"/>
         </contribution>
```

```
      </codesystem>
      <codesets>
         <codeset name="EXT-GEN-DOCUMENT-CHARSET">
            <resource path="codesets/document" filename="EXT-GEN-DOCUMENT-
CHARSET.xml"/>
         </codeset>
      </codesets>
      <codeterms path="codesystems/document" filename="Charset_TERM.xml">
         <codeterm locale="de"/>
         <codeterm locale="en"/>
         <codeterm locale="es">
            <resource path="codesystems/document" filename="Charset_es_TERM.
xml"/>
         </codeterm>
      </codeterms>
   </terminology>
```

Please note that the duplicate `terminology` block was removed. The `codeterms` points to the file `Charset_TERM.xml`, and for the locale `es`, points to a different codeterm file `Charset_es_TERM.xml`.

- **Codesystems marked as `draft` are not imported.**

  The new import process imports only the codesystems that are marked with `status` as `final`. Trying to import a non-`final` codesystem would result in an Exception (with the message: `No code for xxx given`).

  The solution for this problem is to change the `status` of the codesystem in the generated XML file to `final`.