

**Documentation of the Subtitling
Conversion Framework (SCF) - 27/09/2017**

Contents

- About the SCF PDF documentation.....3
- Introduction.....3
- General prerequisites.....4
- General approach.....4
 - Structure of requirements.....4
 - Tests.....5
 - General Test Procedure.....5
 - Module STLXML-XSD.....5
 - Module STLXML2STL.....5

About the SCF PDF documentation

Copyright 2017 Institut für Rundfunktechnik GmbH, Munich, Germany

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License.

You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, the subject work distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and limitations under the License.

Introduction

The subtitle conversion framework (SCF) is a set of different modules that allows a user to convert between different file based subtitle formats. The original driver and still the main purpose of the SCF is to convert a subtitle file that conforms to the binary subtitle format EBU STL (Tech 3264) in one XML document that conforms to one of the EBU-TT profiles (EBU-TT Part 1 - Tech 3350 or EBU-TT-D Tech 3380).

The main design idea is to keep the different conversion modules as separate as possible so that in a processing chain one module implementation can be replaced by an alternative implementation.

In addition the SCF follows clearly defined and documented requirements. The documentation of the requirements are published together with the source code. Where possible the requirements that are implemented are covered by automated software tests.

Although there is no reason to not use the SCF in operation its goal is primarily to provide a reference framework that shows in a transparent way how to implement new subtitle standards (especially EBU-TT) and how to integrate them in existing work flows. The source code can be used by other tool providers to implement their own conversion methods. They can just take it as a reference or they could just reuse some of the SCF code.

The current version of the SCF is version 1.0. Comments from the community that we received since the first publication of SCF have been considered. Any comments regarding the current version are welcome and should be made by opening an issue on GitHub or sending an email to open.source@irt.de

Overview of the processing chain

The main conversion chain follows these processing steps:

- The start point is a binary EBU STL which is decoded and serialized in a custom XML representation of the EBU STL file, called STLXML. This has the following advantages: XML processing tools can be used from that point onwards the XML representation of the EBU STL file is human readable standard XML validation technologies can be used.
- The STLXML representation is then transformed into an EBU-TT file that conforms to EBU-TT Part 1. The guidance for the transformation is the EBU document EBU Tech 3360 (also known as EBU-TT Part 2). Like an EBU STL file the purpose of an EBU-TT Part 1 document that was created from an EBU STL file based on the Tech 3360 guideline is thought to be used in exchange and production but not for distribution.
- To get an EBU-TT file that can be used for distribution over IP based networks a transformation to EBU-TT-D (EBU Tech 3380) is necessary. The module for this conversion step expects an EBU-TT file that follows the Tech 3360 guideline and will create a valid EBU-TT-D file.

<https://github.com/IRT-Open-Source/scf>

General prerequisites

- For the XSLT modules, an XSLT processor that is conformant to XSLT 1.0 is needed. You could use for example a Saxon XSLT processor from version 6.5.5.
- To validate an STLXML with the STLXML W3C XML Schema, an XML Schema 1.0 parser is required. You could use for example the Xerces XML parser and validator.
- For the conversion of an EBU STL file into STLXML (an XML based representation of the EBU STL file), you need the Python script `stl2stlxml.py`. This script requires a version 2.7.x Python runtime environment - it will not run with Python version 3.0.

<http://saxon.sourceforge.net/>

<https://xerces.apache.org/>

<https://www.python.org/downloads/>

General approach

Structure of requirements

The SCF implementation is based on the requirements for the different modules. The requirements are available as documentation of the different modules.

The structure of the requirements is as follows:

- **Title:** a short title, prepended with the internal ID of the requirement
- **Description:** the requirement text. The specified text will be used to test the implementation.
- **Area:** Apart from more general requirements the requirements are categorized by modules (e.g. STLXML2EBU-TT or EBU-TT2EBU-TT-D)
- **Requirement Review Status:** This is the internal review status of the requirement itself (especially of the requirement text).
- **Status Implementation:** This status indicates if the requirement is already met by the implementation. The possible status codes are:
 - *outstanding* - the corresponding code has not yet been written, or the requirement has been implemented but there are no test files for it
 - *waitingReview* - the code to implement the requirement has been written but nobody (except the developer) has reviewed the code yet
 - *underReview* - the corresponding code is under review and has not yet been accepted by the first reviewer
 - *reviewed* - the corresponding code has been reviewed and accepted by the first reviewer
 - *accepted* - the corresponding code has been accepted by the developer team and is ready to be published
- **Priority according to MoSCoW:** the priority that is the base to decide when the feature will be implemented. The possible values are:
 - *M* - must
 - *S* - should
 - *C* - could
 - *W* - won't

For more information see: https://en.wikipedia.org/wiki/MoSCoW_method

Tests

While a general procedure applies to most of the SCF modules, in some cases a different approach has been chosen.

General Test Procedure

This procedure applies to most of the SCF modules. The test files that are used as test input for a module are named according to the following pattern:

```
requirement-[requirement ID]-[number of test for this requirement].[file suffix]
```

Example: The first test file for the requirement 27 in an XML format is named `requirement-0027-001.xml`.

If there are certain assertions written that can be automatically processed (e.g. by an Schematron schema), each assertion file has the corresponding file name of the test file (with file suffix of the assertion format). For example a Schematron file that tests the output of a module that gets the test file `requirement-0027-001.xml` as input would be named as `requirement-0027-001.sch`.

Module STLXML-XSD

As the STLXML-XSD module consists only of an XSD file that describes the STLXML format, the test files can only pass or fail the XSD validation. For this reason no Schematron files are needed to check the test result - instead the expected result is just contained in the file name of each test file in form of the token `PASS` or `FAIL`. Hence all STLXML-XSD test files are named according to the following pattern:

```
requirement-[requirement ID]-[number of test for this requirement]-[exp. result].xml
```

Example: The third test file for the requirement 506 (which shall not validate against the XSD) is named `requirement-0506-003-FAIL.xml`. According to its name, this test file must fail the XSD validation.

Module STLXML2STL

The STLXML2STL module uses a different approach for testing that consists of first creating all required test files from a set of templates, and second processing of the actual test files. Details can be found in the respective test README.