Validating DDEX Messages

DDEX develops standards for the electronic communication of music-related metadata using, to the most part, messages. One step in verifying that a message has been created in accordance with the standard is “validation”.
For those DDEX standards based on XML – i.e. ERN, MLC, RIN, MWL, MWN, LoD and others – off-the-shelf XML tools can be used to help with validation. For the flat file standards (DSR and, soon, CDM) DDEX members have created tools.

This article provides some guidance on how to validate DDEX messages.

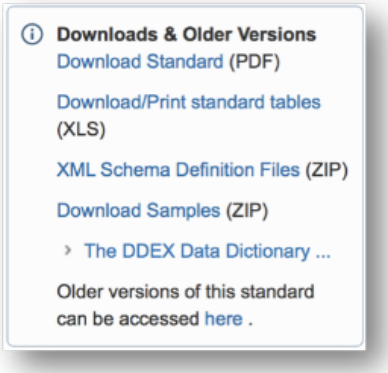
Validating XML Messages

DDEX has published two XML Schema Definition (XSD) files for each standard: one that defines the structure of the messages that can be created for that standard and a second one that defines all the allowed-values for the DDEX standards. These two files are sometimes referred to as the “message XSD” and “AVS XSD”.

The message XSD loads the AVS XSD to making AVS validation an integral part of the message validation.

The XSDs are published in two versions and locations: The “online versions” are published on a web server at the URL indicated by the namespace of the message. The XSD for Version 3.8.2 of the ERN standard, which has a namespace of http://ddex.net/xml/ern/382, is published on the DDEX website at http://ddex.net/xml/ern/382/release-notification.xsd. The online version of the message XSD then imports the online version of the AVS XSD, which is located at http://ddex.net/xml/avs/avs_20161006.xsd for ERN 3.8.2. Implementers and users can use the online versions of the XSDs for their applications as well as for their production system.

The XSDs are also published on the knowledge base for downloading and for local, offline, use. All XML standards’ web pages display a box such as the one shown overleaf, which allows to download the XSD files in a zip archive.



The offline message XSD loads the AVS XSD from the same zip archive (and not the online version discussed above), thus allowing to implement and use the XSDs without the need for an internet connection.

Other than referring to the offline AVS XSD there is no difference between the two sets of XSDs[1]. While the examples below use the online XSD files, there is no material difference for users who wish to use the offline XSDs.

For the ERN standard, DDEX has also developed profiles that define constraints beyond those defined by the message and AVS XSDs. To support an automated checking whether an ERN message meets these profile constraints, DDEX has developed – but not published beyond its membership – ISO/Schematron rules.

A complete validation of an XML message created in accordance with a DDEX standard may therefore require two steps: XSD Validation and Schematron Validation.

Step 1 – XSD Validation

Click here to expand...

The first step to validate a DDEX message is to verify that the structure of the XML file conforms to the XML Schema Definition file for that standard.

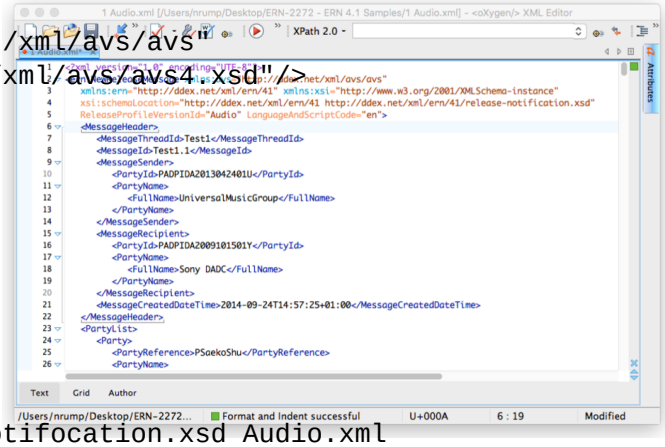
IDE

This can be done in an XML editor such as Syncro Soft’s Oxygen or Altova’s XmlSpy. Other equivalent tools are available. The screenshot on the right shows a valid ERN message in accordance with Version 4.1. Please note:

- The namespace of the message is `http://ddex.net/xml/ern/41`;
- The location for the XSD for this namespace is `http://ddex.net/xml/ern/41/release-notification.xsd`;
- The button to trigger the validation – if not done automatically – is the document icon with the large red check mark; and
- The green square in the top right corner indicating that the file is valid.

This validation also includes the verification of the allowed values. This can be verified by opening the ERN XSD and finding a HTTP based AVS schema location:

```
<xs:importnamespace="http://ddex.net/xml/avs/avs"
    schemaLocation="http://ddex.net/xml/avs/avs4.xsd"/>
```



Any errors found by the tool will be highlighted and will allow the user to go to the offending location.

Command Line (UNIX)

It is also possible to verify the XML file with a command line tool. For UNIX systems this can be done using xmllint:

```
> xmllint --noout –schema release-notifocation.xsd Audio.xml
```

xmllint prints out a status (e.g. “Audio.xml is valid”) and returns an error code to enable it to be used as part of a larger application.

Command Line (Windows)

For Windows systems this can be done as follows using a script for the Microsoft PowerShell as discussed on https://gallery.technet.microsoft.com/scriptcenter/2f6f0541-d152-4474-a8c1-b441d7424454 or by using AltovaXML Community Edition 2013. While Altova does no longer support the 2013 edition of its XML Community Edition, it is still available to download from http://www.softpedia.com/get/Internet/Other-Internet-Related/AltovaXML.shtml.

Online tool

DDEX is in the process of developing and deploying an online tool that can be used to test messages.

Step 2 – Profile Validation (ERN only)

Click here to expand...

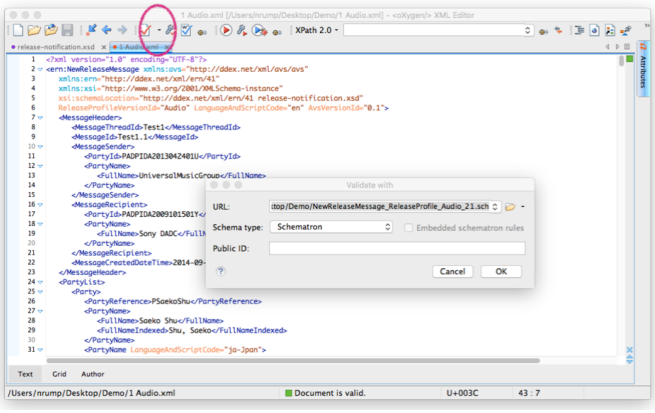
As indicated above, DDEX has published ISO/Schematron rules for recent ERN versions to its members. These Schematron files can be used to verify whether an ERN message conforms to the relevant profile.

They can be used in an XML editor in the same way as an XSD as long as the editor is able to handle XSLT2 and Xpath2 expressions. Most modern XML editors – such as Oxygen and XmlSpy[2] – are XSLT2/Xpath2 compatible.

IDE

In Oxygen this can be done by clicking the downward triangle next to the XSD validation button (encircled in red in the diagram below), selecting “Validate with…” and entering the appropriate information into the relevant window:

- The URL field needs to point to the Schematron file provided by DDEX; and
- The Schema Type field needs to have “Schematron” selected.



Any Schematron error will, as with the XSD validation, allow the user to navigate to the offending location. And, as with XSD validation, a green box in the top right corner indicates that no Schematron errors were found.

Command Line (UNIX)

The Schematron validation can also be done from a command line or within an application. However, the process is a bit more complex and involves having a working XSLT processor capable of handling style sheets in version XSLT2 installed.

Below is the process for the Saxon processor. Schematron files are, in effect, XML Stylesheets (XSLTs) but they need to be prepared for the

specific XSLT processor at hand. For Saxon this happens in three steps and involves calling a Java runtime environment as follows:

```
> java -jar $saxon -xsl:iso_dsdl_include.xsl -o:tmp1.xml -s:ddex.sch
> java -jar $saxon -xsl:iso_abstract_expand.xsl -o:tmp2.xml -s:tmp1.xml
> java -jar $saxon -xsl:iso_svrl_for_xslt2.xsl -o:ddex.xsl -s:tmp2.xml
```

With `$saxon` pointing to the JAR file from the Saxon package and ddex.sch being the Schematron rules file provided by DDEX. The above three steps can be prepared and only the output from the last command, `ddex.xsl`, needs to be retained. This file can then be used to test a DDEX Message file as follows:

```
> java -jar $saxon -xsl:ddex.xsl -s:test_file.xml > output.xml
```

The output will be an XML file that can be parsed, for example, by grep, and if the string role="Fatal Error" does not occur, the file passes the validation.
The Schematron rules are set up in a way, however, to also provide warnings and conditional errors that can equally be filtered out by role="Error", role="Conditional Error" and role="Conditional Fatal Error". In addition, the documentation of the errors provides information on the rule that has been violated.

Command Line (Windows)

For Windows systems the same procedure as for UNIX can be used and the two commands for generating the XSLT file ddex.xsl from the Schematron file and using it to test the file test_file.xml are as follows:

```
c:/> java -jar %SAXON_PATH% -o:ddex.xsl -s:ddex.sch
iso_svrl_for_xslt2.xsl
c:/> java -jar %SAXON_PATH% test_file.xml ddex.xsl > output.xml
```

Online tool

DDEX is in the process of developing and deploying an online tool that can be used to test messages.

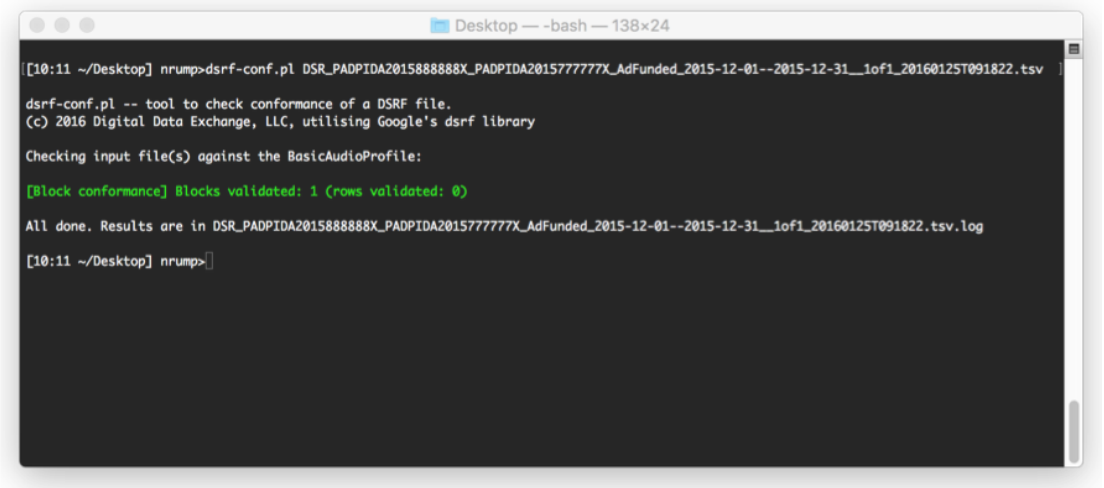
Validating Flat-file Messages

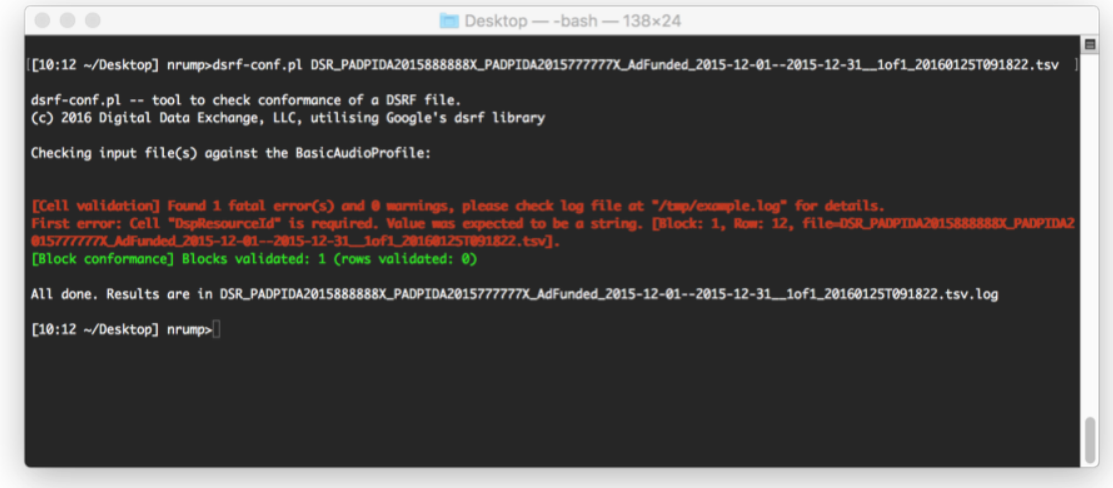
The validation of a flat-file message such as a TSV file generated in accordance with the Digital Sales Reporting Message Suite Standard (DSR) can be conducted by a tool developed by Google. The tool is available to download from GitHub at https://github.com/ddexnet/dsrf.

The tool can be deployed on any computer running Max OS X, LinuX or MS Windows with a Python v2.7 (or later) interpreter installed. The tool comes with a manual and can be used with a small Perl script `dsrf-conf.pl` as follows:

```
> dsrf-conf.pl file.tsv
```

The two screen shots below show a successful and an unsuccessful test. In each case, a log file containing all findings is generated. The script is provided in Annex A.





Perl Script for DSRF Validator

This is a Perl script calling the DSRF Validator discussed in Clause 3 to test whether a DSR file is in accordance with the relevant profile. The script, written for OS X, expects the variables `$validator` to point to the location of the DSRF Validator.

```
#!/usr/bin/perl
use strict;
use warnings;

print "\ndsrf-conf.pl -- tool to check conformance of a DSRF file. \n"
print "(c) 2016 Digital Data Exchange, LLC, utilising Google's dsrf
library\n\n";

#
# set variables (filelist being the alphabetical list of input files)
#
my $validator = "/Users/nrump/bin/dsrf";
my $parser = $validator . "/run_dsrf.py";
my $conformance = $validator . "/conformance/conformance_processor.py";
my $log = "/tmp/example.log";
my $log2 = "/tmp/example2.log";
my $usage = "Please use the script as follows:\n\tdsrfconf.pl FILE
[FILE*]\nProgram aborted";
my @filelist = sort (@ARGV);
my $filecount = $#ARGV + 1;
```

```
my $profile = "";
$/ = "\r";     # to support all kinds of CR/LF/CRLF combos

#
# see if at least one file has been given and that all files exist
#
die "No file to process. $usage" if ( $filecount < 1);
-f or die "File '$_' does not exist. $usage" foreach (@filelist);

#
# open the first file (which is ...1ofx... as the array is sorted) and
obtain the profile
#
open (FILE, "<$filelist[0]");
while(<FILE>){
  s/^\s+//; # make sure there are no leading spaces
  next unless (/^HEAD/);
  my @fields = split "\t";
  $profile=$fields[2];
  last;
}
close FILE;

#
# end validation if there is no profile
# if there is: remove all spaces from the profile name (as this is what
the tool expects)
#
if ( $profile eq "") {
  print "    No profile given in $filelist[0]\n    Conformance cannot be
tested\n";
  open FILE, $filelist[0].log;
  print FILE "No profile given in $filelist[0]\n    Conformance cannot
be tested\n";
  close FILE;
  exit;
}
print "Checking input file(s) against the $profile:\n\n";
$profile =~ s/ //g;

system "python $parser @filelist | python $conformance > $log2";
```

```
system "cat $log $log2 > $filelist[0].log";
system "rm -f $log $log2";
print "\n\nAll done. Results are in $filelist[0].log\n";
```

[1] At least for comparatively recent versions of the DDEX standards. For older versions there may be the possibility that the online AVS XSD has been updated since the offline XSD had been published.

[2] XmlSpy does not support Schematron validation out of the box. Adding Schematron capabilities to XmlSpy requires installing a plug-in.