RSMFValidatorSampleCode

Introduction

The RSMFValidatorSampleCode project contains source code that leverages the Relativity Short Message Format Validator library. The source code demonstrates how to use the Relativity Short Message Format Validator library to verify that generated RSMF files conform to the RSMF specification and that they contain consistent data. This document also has several exercises that a user can run with the sample application to better understand how the Validation library works and how to solve the issues that the library might report.

Included with the sample code in the RSMFManifestSchema folder is rsmf_schema_1_0_0.json. This file describes the schema that is used to validate the data in rsmf_manifest.json files. It is provided to assist developers with creating their own applications that generate RSMF files. All rsmf_manifest.json files with a version of 1.0.0 must adhere to this schema.

For all samples, the RSMFValidatorSampleCode application should be compiled. See readme.md for more instructions. A text editor that supports UTF-8 encoding will also be required.

Sample 1

This sample contains an rsmf_manifest.json file that contains content that violates the RSMF JSON schema.

1. Run RSMFValidatorSampleCode on Documentation\Sample1.
2. The sample application reports that the file has failed validation.
3. Inspect the resultant rsmf.validation.log.txt file.
4. In this file the specific error encountered is listed. In the case of sample 1, there is an invalid participant. The participant is reported as being invalid because it's missing a required property.
5. Locate the participant in rsmf_manifest.json. Hint: The line and position information are given as part of the error message.
    a. To correct the invalid participant, add a display property with a valid string to the participant that is defined on line 4.

```
{
"id": "1",
"display": "participant name"
}
```

6. After the rsmf_manifest.json file has been corrected, rerun the sample application. The file is now reported as passing.

Sample 2

This sample contains an rsmf.zip file that passes with warnings. If you have gone through the Sample 1 exercise, make sure to delete the rsmf.validation.log.txt file that was generated. The sample application appends its output to that file and it may cause confusion. Also, the program 7-Zip or some other zip utility is needed for this exercise.

1. Run RSMFValidatorSampleCode on Documentation\Sample2.
2. The sample application reports that the file rsmf.zip has passed with warnings.
3. Inspect the resultant rsmf.validation.log.txt file.
4. In this file the specific error encountered is listed. In the case of sample 2, there is a participant that references an avatar, but that avatar is not included in the Zip file.
5. Warnings are errors that may not cause issues in Relativity. While this warning may not cause issues in Relativity, for this exercise it is expected that all avatars will be present.
6. There are two ways to address this warning. The first is to remove the reference to the avatar in the rsmf_manifest.json file that is in the rsmf.zip file. As this avatar is expected to be present, this isn't the best solution. Instead the solution is to add the correct avatar to the rsmf.zip file.
7. A file, avatar.png has been included in the Sample2 directory. The warning in the log file indicated that the missing avatar was named AV1. For the issue to be corrected, rename avatar.png to AV1. Notice that no extension is used in this example.
8. Add the AV1 file to rsmf.zip using your choice of application.
9. Rerun the sample application. The file rsmf.zip now passes.

Sample 3

This sample demonstrates how to address issues that occur at the RSMF EML layer. It is handy to have some knowledge of the MIME format (RFC 5322) but it is not necessary to follow the exercise. If you have gone through the Sample 1 or Sample 2 exercise, make sure to delete the rsmf.validation.log.txt file that was generated.

1. Run RSMFValidatorSampleCode on Documentation\Sample3.
2. The sample application reports that the file sample.rsmf has failed.
3. Inspect the resultant rsmf.validation.log.txt file.
4. In this file the specific error encountered is listed. In the case of sample 3, there is an extra attachment at the RSMF EML layer.
5. Examine sample.rsmf in a text editor. Notice at line 43 a new section begins (this is designated by the --RSMFEML indicator). The headers for this section indicate that it is an attachment.
6. RSMF files are only allowed to have rsmf.zip as an attachment. It is necessary to remove the second attachment from the RSMF file. Note that this isn't the same thing as removing an attachment from the rsmf.zip file.
7. To remove the attachment, delete all lines of the file from the --RSMFEML on line 43 up to but not including the next line that begins with --RSMFEML.
8. Rerun the sample application. The file sample.rsmf now passes.

Sample 4

This sample demonstrates how to address issues that take multiple passes of validation. It uses an rsmf_manifest.json file with several errors to demonstrate that some issues cannot be detected until previous issues are addressed. If you have gone through previous exercises, make sure to delete the rsmf.validation.log.txt file that was generated.

1. Run RSMFValidatorSampleCode on Documentation\Sample4.
2. The sample application reports that the file rsmf_manifest.json has failed.
3. Inspect the resultant rsmf.validation.log.txt file.

4. The report indicates there was a ManifestSchemaViolation. The property “id” expects a string value, not an integer.
5. Correct the issue by adding quotes around the id value. Rerun the validation sample application.
6. Three new errors have been logged to rsmf.validation.log.txt file. The first two errors, ManifestInvalidConversationParticipant and ManifestInvalidEventParticipant, are generated because the conversation lists participant “2” as a member, but there is no participant with an id of “2”. The third, ManifestUndefinedProperty, shows that a property “attachment” isn’t a property that is defined by the RSMF specification.
7. To correct the first two errors, it must be determined if there is a missing participant in the participants array or if the id in the conversation’s participants array is incorrect. For this exercise it is assumed that the conversation’s participants array is incorrect. To correct these issues, change the “2” to a “1” in the conversation’s participants array.
8. The undefined property can be addressed in two ways. It’s possible that the creator of this rsmf_manifest.json meant to include an attachment. If this were the case the correct property to use is the “attachments” property with an array of attachment objects. It might also be possible that the creator did intend to use their own property. In this case, the correct way to do this is to use the “custom” property with an array of name/value pair objects. It is left as an exercise to the reader to determine the best way to address this issue.
9. After the final issue has been addressed, rerun the validator sample application. The file is now reported as passing.