

USING THE EPUBCHECK 4.0 PREVIEW APPLICATION

1. Overview/Summary

Significant effort has gone into enhancements to ePubCheck 3.0. This new release is currently named "ePubCheck 4.0 Preview." The primary goals for these enhancements were:

- To provide enhanced checks to identify notable or potentially problematic ePub 3 constructs.
- To make it easier to integrate ePubCheck into an automated ePub processing system by providing all program output in JSON for easy consumption by ePub processing tools
- Regularize the form and content of ePubCheck console output in order to make it easier to read the program's output.
- To make the checks performed by ePubCheck more understandable, the --listchecks switch was added, which causes the "dictionary" of messages to be written to the screen or to a file.
- Allow user control over ePubCheck message content and severity, again to allow ePubCheck to more easily integrate into a processing workflow.

Contents

1. Overview/Summary	1
2. Command Line Help	2
3. New features.....	3
a) Rationalized messages and message output control.....	3
b) Enhanced Checks	3
c) JSON output of ePubCheck Messages	3
d) ePubCheck Message Dictionary	3
e) Message Customization: --customMessages <messageFile>	4
(1) customMessages messageFile format.....	4
(2) messageFile specification on the command line and via environment variable	5
4. JSON File Structure.....	6
5. See Also "BookReporter.py.docx"	7

2. Command Line Help

Epubcheck Version 4.0.0-SNAPSHOT

When running this tool, the first argument should be the name (with the path) of the file to check.

If checking a non-epub file, the epub version of the file must be specified using `-v` and the type of the file using `-mode`.
The default version is: 3.0.

Modes and versions supported:

```
--mode opf -v 2.0
--mode opf -v 3.0
--mode xhtml -v 2.0
--mode xhtml -v 3.0
--mode svg -v 2.0
--mode svg -v 3.0
--mode nav -v 3.0
--mode mo -v 3.0 // For Media Overlays validation
--mode exp // For expanded EPUB archives
```

This tool also accepts the following options:

```
--save           = saves the epub created from the expanded epub
--out <file>     = output an assessment XML document file.
--json [<file>]   = output an assessment JSON document file
-m <file>        = same as --mode
-o <file>        = same as --out
-j <file>        = same as --json
-f, --fatal      = include only fatal errors in the output
-e, --error      = include only error and fatal severity messages in output
-w, --warn       = include fatal, error, and warn severity messages in output
-u, --usage      = include ePub feature usage information in output
                  (default is OFF); if enabled, usage information will
                  always be included in the output file

-l, --listChecks [<file>] = list message ids and severity levels to the custom message file named <file>
                           or the console
-c, --customMessages [<file>] = override message severity levels as defined in the custom message file named <file>

-h, -? or --help = displays this help message
```

ePubCheck's command line help has been updated to reflect the new features that have been added to the program. The rest of this document explains the new features. As of this pre-release, documentation for the existing features is still provided on the Google Code ePubCheck web site:

<https://code.google.com/p/epubcheck/wiki/Running>.

3. New features

a) Rationalized messages and message output control

ePubCheck's messages have been reviewed for consistency and clarity, and unique identifiers added to uniquely identify each message that can be generated by the program.

In addition, the **--fatal**, **--error**, **--warn**, and **--usage** command line flags have been added to provide more control over the output written to the console.

b) Enhanced Checks

A number of new checks have been added to ePubCheck, including a number of **"USAGE"** checks that document ePub feature usage and the structure of the ePub file itself. To generate the messages included in the current version of the program, use the ePubcheck command switch **--listMessages <filename>** to save them to a tab-delimited file.

c) JSON output of ePubCheck Messages

The **--json [<file>]** command line switch causes ePubCheck to generate a .JSON file that contains all program output in a form that is easily consumable by software tools or scripts. See [JSON File Structure](#) in this document for an overview of the file format.

If **<file>** is omitted, ePubcheck will name the output file **<inputFileName>check.json**.

Thus, if you are checking the ePub **MyEpub.ePub**, the resulting .JSON file will be named **MyEpub.ePubcheck.json**. A full path may be specified in the **<file>** parameter.

d) ePubCheck Message Dictionary

As of this writing, the set of checks in ePubCheck++ and the messages generated by those checks are grouped into several categories:

- **ACC** – issues and usage information related to content accessibility
- **CHK** – errors related to the processing of an ePubCheck custom message file
- **CSS** – issues and usage of CSS
- **HTM** – issues and usage with ePub content documents
- **MED** – issues and usage of media (audio and video)
- **NAV** – ePub navigation document
- **NCX** – legacy ePub 2 NCX navigation/toc document
- **OPF** – publication OPF file
- **PKG** – .ePub container file
- **RSC** – resource files (image files, .css files, etc.) contained in the ePub
- **SCP** – scripts contained in the .ePub file

To generate the messages included in the current version of the program, use the ePubcheck command switch

--listMessages <filename> to save them to a file.

By default, ePubCheck's messages are assigned one of four severities:

- **FATAL** a severe problem with the ePub file itself prevents checking
- **ERROR** a major problem that will almost certainly prevent successful ePub rendering
- **WARNING** an aspect of the book is not compliant with the ePub specification, but the book is *likely* to be renderable by most reading systems
- **USAGE** message that documents the use of specific ePub features or markup patterns

In general, ePub authors should correct problems with severities of **FATAL** or **ERROR** as they are likely to not display correctly in most reading systems.

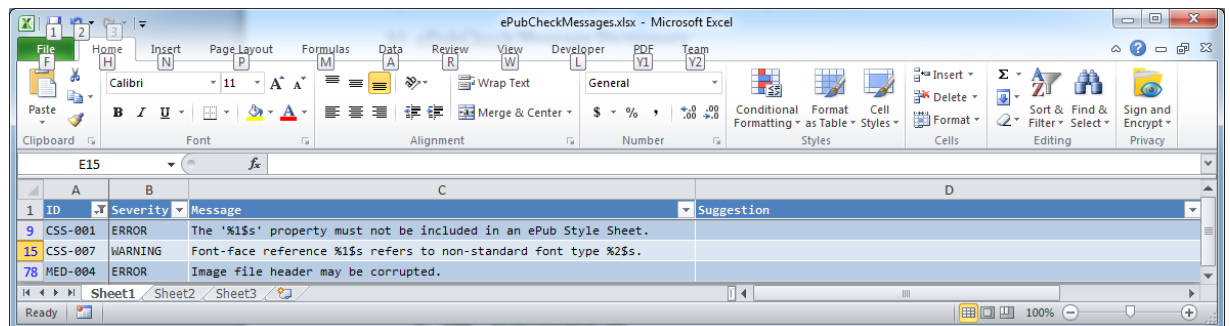
e) Message Customization: `--customMessages <messageFile>`

The Message Customization feature allows users to override the default severity and text of messages reported by ePubCheck. This is handy when automated systems or procedures depend on ePubCheck results to control an ePub-processing workflow. As an example, an organization could require correction of any ePub that produces **ERROR** messages. If that organization regards accessible content as an absolute requirement, the accessibility messages could all be overridden to have a severity of **ERROR**, thus stopping further processing until the accessibility issues have been corrected.

(1) customMessages messageFile format

To use the custom messages feature, a tab-delimited text file must be created that contains the override severity and or message and suggestion texts for each custom message. This file is specified when ePubCheck is run (see [section 3.e.2](#), below)

The file named by the `<messageFile>` parameter of the `--customMessages` command line option contains the override messages and severities for each `messageID` included in the file. The format for the message file matches that of the file created by the output of the `--listChecks` option. The file contains an optional header row, plus one record per customized message, each field in the record delimited by a tab character. These files can easily be created in a spreadsheet application, such as Microsoft Excel:



ID	Severity	Message	Suggestion
9 CSS-001	ERROR	The '%1\$s' property must not be included in an ePub Style Sheet.	
15 CSS-007	WARNING	Font-face reference %1\$s refers to non-standard font type %2\$s.	
78 MED-004	ERROR	Image file header may be corrupted.	

and saved in Excel's "tab-delimited text file" format (using the Save As... command), or in any word processor that preserves tab characters.

The header row may be omitted from the `messageFile`.

When a message's severity is overridden or suppressed, the original severity is shown in the message text. This feature should be used with care, especially the "down-grading" of error severity.

There are five valid **messageSeverity** values:

- **FATAL**
- **ERROR**
- **WARNING**
- **USAGE**
- **SUPPRESS**

If a severity value other than one of these 5 is encountered in a **messageFile**, ePubCheck 4.0 Preview will display an error and exit abnormally.

The special severity value **SUPPRESS** causes ePubCheck 4.0 Preview to output the message using the special severity value of **SUPPRESS**, either to the console or to the .json output file. This allows a process or procedure to ignore those messages. For example, if an organization did not choose to release accessible ePubs, a **messageFile** could be created that overrides all accessibility check severities to **SUPPRESS**; their process or procedure would see no accessibility **ERROR** or **WARNING** errors, and further processing could proceed.

messageIDs in the **messageFile** must exactly match a **messageID** known to ePubCheck 4.0 Preview. If ePubCheck 4.0 Preview encounters a **messageID** that does not match one of its standard messages, ePubCheck 4.0 Preview will display an error and exit abnormally.

The recommended pattern of usage is to start with the output of the **--listChecks** option, remove any messages that you don't wish to customize, and then edit the **messageText** and **messageSeverity** of the messages you do want to customize.

(2) **messageFile specification on the command line and via environment variable**

There are two ways to use the customMessages feature:

- Use the **--customMessages <messageFile>** switch when starting ePubCheck
- Define the environment variable **ePubCheckCustomMessageFile** to be the path to a **messageFile**

If the environment variable **ePubCheckCustomMessageFile** is defined as the name, and perhaps absolute path to, a **messageFile**, then ePubCheck 4.0 Preview will automatically use the custom messages in the **messageFile** referred to by the environment variable, *even if the **--customMessages** switch is not included in the command line*. See below for a method to override the **messageFile** referred to by **ePubCheckCustomMessageFile**.

If the environment variable is defined and the **--customMessages** switch is used with a **messageFile** parameter, the **messageFile** named in the command line will be used instead of the **messageFile** named by the environment variable.

If the environment variable is defined and the **--customMessages** switch is used with a **messageFile** parameter value of **"none"**, ePubCheck 4.0 Preview will run without using a custom message file; the default messages and severities will be used.

When a **messageFile** is used in a given check, the name of the message file is included in the **checker** element of the output .json file as an property pair:

messageFile : <messageFilePath>

If the **messageFile** can't be found or successfully loaded, EPubCheck 4.0 Preview will display an error and exit abnormally.

4. JSON File Structure

The .JSON file produced consists of 5 sections, each of which contains:

- **customMessageFileName**

The name of the custom message file, if any, in use during the ePUBCheck session that produced the .json file.

- **checker**

The checker section contains a number of name-value pairs that describes the ePUBCheck run. The information about the check includes metadata like the path to and file name of the ePUB checked, the date and time of the check session, the number of **FATAL**, **WARNING**, **ERROR** and **SUPPRESS** messages, etc.

- **publication**

The publication section of the file contains metadata about the publication, primarily data extracted from the ePUB's .OPF file. The specific metadata included is:

publisher, title, creator, date, subject, description, rights, identifier, language, nSpines, checksum, renditionLayout, renditionOrientation, renditionSpread, ePubVersion, isScripted, hasFixedFormat, isBackwardCompatible

- **items**

The items section of the .JSON file contains a list of every file included in the ePUB container, and includes the following metadata about each item:

id, fileName, media_type, compressedSize, uncompressedSize, compressionMethod, checksum, isSpineItem, isLinear, navigationOrder, isHTML5, isFixedFormat, isScripted, scriptSrc, scriptTag, scriptInline, renditionLayout, renditionOrientation, renditionSpread, referencedItems []

The **referencedItems []** item is an array of files in the ePUB (or external links) used by the item.

In the case of ePUB files that do not include an id value in the ePUB manifest, ePUBCheck fabricates a name based on the path to the file in the ePUB.

- **messages**

The messages section of the .JSON file contains a list of the messages generated by ePUBCheck and a list of up to 25 "locations" in the ePUB file (file name, row and column number) that caused the message to be generated. When there are more than 25 locations that triggered the message, only the first 25 are included in the .JSON and a 26th record is included that tells the number of additional locations that triggered the error. Each message listed in the .JSON includes:

id, severity, message, suggestion text (if any), additionalLocations, and the array of locations

5. See Also “BookReporter.py.docx”

As part of the ePubCheck enhancement project, a prototype python script was created to validate the .JSON file structure and to ensure that the scenarios we wanted to enable through the .JSON file could be successfully implemented. Both the script and the [BookReporter.py document](#) are packaged with the enhanced ePubCheck java application.