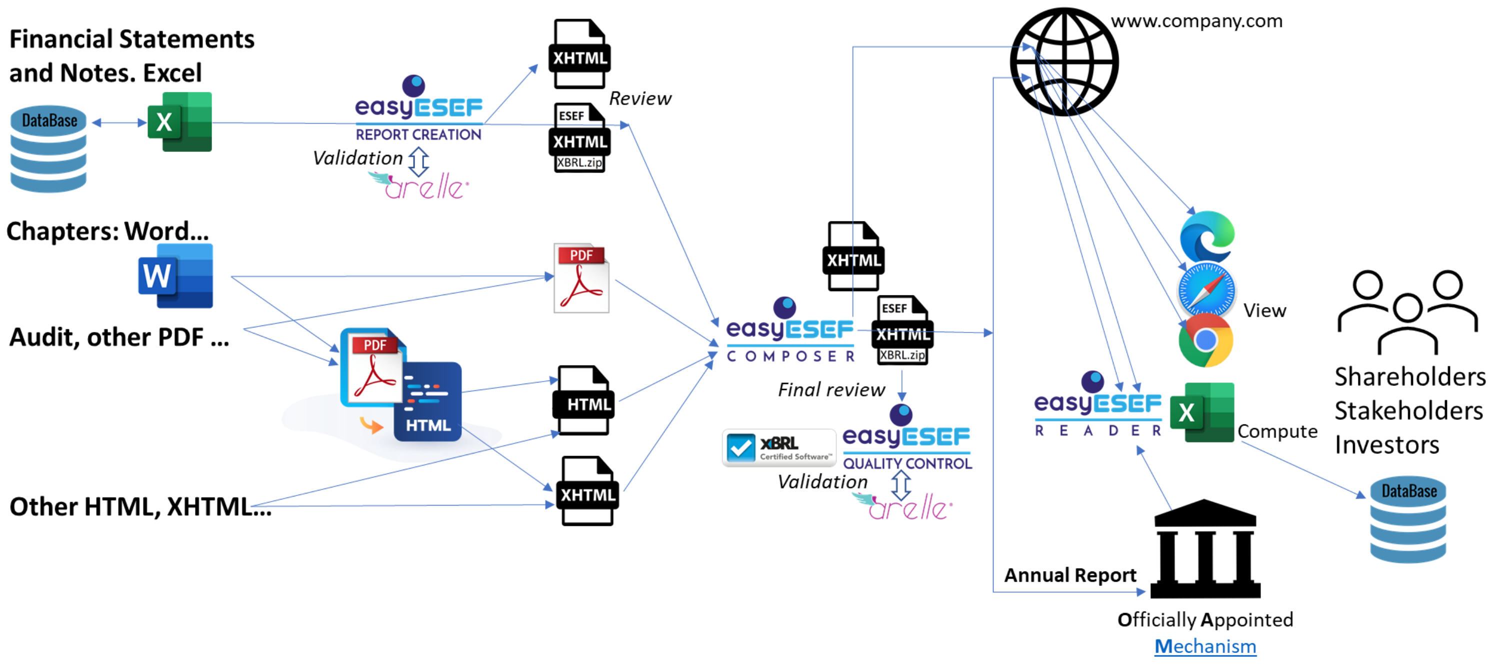
Technical reference

easyCOMPOSER: Compose the ESEF Annual Report

easyCOMPOSER is a tool designed to export a file with the Annual Report in ESEF format. This file is to be sent to the *Officially Appointed Mechanism* and that to be published on the company's website. Imports the Financial Statements and Notes already tagged in ESEF format as a .zip file, and the Chapters and other components in formats: **.pdf .html .xhtml**



easyCOMPOSER composes the Annual Report in ESEF format as a file with the Annual Report viewable in .xhtml and files with mandatory meta-information (*XBRL linkbases*), all compressed according to the standard *taxonomypackage.zip*

easyCOMPOSER exports Annual Report .zip in the directory indicated. It also exports a copy of the Annual Report in .xhtml for convenience of use.

easyCOMPOSER assumes that the imported files are correct. It is recommended to run a final review with the ***easyQC*** Quality Control tool, certified by XBRL International. This tool uses the Arelle validator internally, also certified by XBRL International.

easyCOMPOSER combines the Financial Statements and Notes .xhtml file (which comes inside the .zip file) with the other .pdf or .html or .xhtml files in the order indicated. You can only put a file .zip at most. It also works if you only put .pdf or .html or .xhtml files and omit the .zip file.

easyCOMPOSER internally converts .pdf and .html format files to .xhtml format as required by ESEF format. The presentation does not vary. See details in annex *Conversion .html to .xhtml*

easyCOMPOSER detects internally if a .xhtml file may affect the presentation of another file. For example, because it have defined a style (<*style*>) that applies to all structures .xhtml of a particular type (for example <*table*> or <*td*>). Or because several files .xhtml use styles (<*style*>) with the same name and different contents. See details in annex *Disambiguation of .xhtml styles*

easyCOMPOSER exports the adapted .pdf and .html and .xhtml files to the selected directory, for further control.

easyCOMPOSER combines the .xhtml source files distinguishing three parts: <style> <body> and "*other.xhtml*". The combined <style> and <body> is the unchanged concatenation of each <style> and each <body> of each file .xhtml source to combine. It can be checked with any text editor that the <style> and <body> of the combined target file has not changed, which is exactly the sequence of the <style> and the <body> of each combined source file. On the other hand, the combination of "*other.xhtml*" follows their specific rules. See details in annex *Combine .xhtml files*

Installation requirements: Microsoft Excel, on Windows or Mac versions.

Composer Sheet

Folder with files: The cell below specifies the *path* of a default directory where:

- There are the files to be imported that do not have their *path* specified
- Exported files are placed
- The necessary work files will be created and deleted.

Ordered import files (.zip .xhtml .html) to compose: In the cells below, the files are put in order, .zip .html and .xhtml to be combined. If a file does not have its path specified, the default directory is used.

Click Run • to run

Composed file exported: The file .xhtml and (if applicable) the Annual Report .zip composed are show. Both files will be in the default directory.

Files with .xhtml adjusts exported: For each file .xhtml that has needed some kind of adjustment, that file is show, which will be in the default directory.

Converter from .pdf to .html: Absolute path, or relative from the default directory, to the *converter.exe* .pdf to .html adding the appropriate parameters (zoom 1.1 for example).

pdf2htmlEX-win32-0.14.6-upx-with-poppler-data-pdf2htmlEX.exe --zoom 1.1

Converter (free download) used: soft.rubypdf.com/software/pdf2htmlex-windows-version

It is open source, and the most widely used in ESEF, in its different versions.

As a commercial converter, PDF2HTML Reflow Paragraph Module has been tested with good results. It does not have a zoom option. The demo version can be downloaded and works perfectly, but only converts odd pages.

ListOfFiles Sheet

Folder with files: The cell below specifies the *path* of a directory from which all the files and subdirectories it contains will be displayed. It is a useful aid.

Click Run ● to run

List of files and folders: Displays the list of files and subdirectories contained in the specified directory.

To know more

For more detailed information and questions on ***easyCOMPOSER***, please contact info@easyESEF.eu

See also our other tools for ESEF at easyesef.eu/resources. In addition to tools, highly specialized support for error identification and correction is available. This service is independent of any provider and is applicable to any ESEF generation software.

FAQ:

Copyright © by easyESEF Ltd. Ireland. All rights reserved. Deposited at WIPO. Reverse reengineering is prohibited. Code may be protected by obfuscat.org solution in some versions.

Running Excel and/or downloading Arelle may require security authorizations in your Entity.

END OF THE MAIN TEXT OF THE TECHNICAL REFERENCE

Annex: Conversion .html to .xhtml

The ESEF format requires the file to be in .xhtml However, there are times when the file is in .html and a conversion is required. The conversion can be checked with https://validator.w3.org/

easyCOMPOSER converts a .html file in a .xhtml file in the following steps:

a) Transfer certain tags[1] and special characters[2] from .html to .xhtml
b) Enclose into a comment the instructions of <![if ! IE]> type as to <!--<![if ! EI]>-->
c) Remove CR LF TAB, double spaces on labels, according to https://www.w3.org/TR/xhtml1/guidelines.html C5
d) Tag names and attributes in tags that are fully uppercase letters are changed to lowercase, such as COLSPAN=3 to colspan=3
e) Attribute values in tags are always enclosed in quotation marks, such as colspan=3 to colspan="3"
f) The tags <img... > and <meta... > that are not closed with </img> or </ meta > are always closed as <img... /> and <meta... />.
g) The <img > tag must have the attribute alt="...." defined. Include alt="" otherwise.
h) Auto-closing tags should leave a space just before the final />, such as <br />
i) On the label <html.... > if the namespace xmlns="http://www.w3.org/1999/.xhtml" is missing, include it, as <html.... xmlns="http://www.w3.org/1999/.xhtml" ... >

Annex: Disambiguation of .xhtml styles

easyCOMPOSER reviews each .xhtml file to combine, in the order in which they were declared in the *Composer* sheet, to determine whether it need disambiguation adjustments. If there is a file .zip, this file is reviewed first.

The styles of the <style> area of the . xhtml file will be adjusted if detected:

- A style directly applicable to tags .xhtml as table{} td{} which would be directly applicable to all tags <table> <td> in the . combined xhtml.
- A style name already used in a previous .xhtml file but with other content (for example, with a different letter font).

In each file to be adjusted:

- A unique suffix (such as "_____N") is prepared for each file to be adjusted.

---

[1] <Br></Br>| <Br>| <hr></hr>| <hr>|<embed|</embed>|<style type=""text/css"">| <del>| </del>| <u>| </u>| <nobr>| </nobr>
<Br />| <Br />|<hr />|<hr />| <img |</img>| <style>| <span style=""text-decoration:line-through;"" >| </span>| <span style=""text-decoration:underline;"" >| </span>| <span style=""white-space:nowrap;"" >| </span>

[2] &nbsp;|&quot;|&amp;|&apos;|&lt;|&gt;
&#160;|&#34;|&#38;|&#39;|&#60;|&#62;

- It is disambiguated by concatenating each original style name with the unique suffix to create a unique style name. This eliminates the possibility of duplicate style names. Example: .ft0 #page_1 changed to .ft0____N #page_1____N
- For styles directly applicable to tags .xhtml such as table{} or td{}, a period "." is added to the beginning of the style name. Examples: table{} td{} changed to .table____N {} .td____N {}
- In the <body> zone, the original style names change to the new unique style names Example: <p class="ft0 "> <div id="page_1"> changed to <p class="ft0___N"> <div id="page_1___N">
- In the <body> zone, in tags with styles directly applicable to .xhtml tags, as table{} or td{}, the new unique style name is added as a class="" attribute. Examples: <table > <td > changed to <table class=". table___N" > <td class=". td___N" >

<u>Annex: Combine .xhtml files</u>

easyCOMPOSER takes each .xhtml source file to combine, in the order in which they were declared in the *Composer* sheet. From each file, it extracts the contents of <style>PAYLOADSTYLE</style> and <body>PAYLOADBODY</body> as well as certain information from *"other .xhtml"* tags such as <title>... </title>.

The target file .xhtml is the result of concatenating these contents into a single <style>... </style> and in a single <body>....</body>, as well as composing *"other .xhtml"* tags.

From each source file "n" to be combined, three types of information are extracted:

1. Content of (there may be none, one or more) <style>PAYLOADSTYLEn</style>
2. content of <body> PAYLOADBODYn</body>
3. Content of "*other.xhtml*", for example:
   a) Starting line <?xml version="1.0" encoding="UTF-8"?>
   b) Tag <! DOCTYPE > (ignored)
   c) Additional namespaces in <html... >
   d) Tag <title>....</title>
   e) Tags not repeated different from the previous ones, taking note of their relative position in the .xhtml structure

easyCOMPOSER exports the .xhtml combined, with a structure .xhtml type:

```
<? xml... >
<html....>
        <head>
                <title>... </title>. The file .zip has priority here
                < /> and other non-duplicate tags, in the position they had
                <style>PAYLOADSTYLE1PAYLOADSTYLE2PAYLOADSTYLE3</style>
        </head>
<body>PAYLOADBODY1PAYLOADBODY2PAYLOADBODY3</body>
</html>
```

Annex: SHA256 hash codes in .xhtml

Practical problem.

A Company is preparing its audited Annual Report. Each Chapter is hypothetically prepared by a different Department. The Auditor receives the different Chapters as they are available, to advance in parallel, given the usual rush of deadlines. It must be verified that the Annual Report consists solely and exclusively of the integrity of the audited Chapters.

In the case of the single European Single Electronic Format ESEF, the Annual Report is a file that can be viewed in any Internet browser (format .xhtml), plus a series of non-viewable files complementary to the financial statements (iXBRL standard).

Due to homogeneity, the Chapters are sent individually to the Auditor as .xhtml viewable files. At the end, the Chapters are combined creating the .xhtml Annual Report.

SHA256 hash code as digital fingerprint

A hash code is the digital fingerprint of the contents of a file. There are several algorithms to obtain the hash code of a file. The *hash* code is obtained using the standard reference algorithm in SHA256 cryptography that produces an unrepeatable 256-bit[3] digital fingerprint. Any alteration of the contents of the file will alter the SHA256 hash code. The SHA256 algorithm is of strong security, so is unknown the method for creating a file whose obtained SHA256 has precisely a defined value. In this way the integrity is guaranteed: if a known SHA256 is obtained from the file, it is that the file has not undergone any alteration.

The SHA256 hash code does not depend on the operating system or any particular software. The SHA256 hash code depends on the encoding of the characters, which will be UTF8 according to the ESEF format. Obtaining the SHA256 of a file or character string is easy with program libraries and online solutions.

Therefore, knowing the SHA256 hash codes of a series of files, you can always check that those files have not undergone alterations.

Concatenating multiple source files into a single target file does not alter the SHA256 hash code of concatenated files. If you know the size of each source file to be concatenated, and the concatenation sequence, you can extract one by one ("deconcatenate") each source file of the concatenation and recalculate its SHA256 hash code.

[3] These 256-bit Usually represent or Not bad how 64 characters Hexadecimal (0-9:A-F, base 16) or Not bad how 44 characters A-Z:a-z:0-9:+:/ in base 64 and with a = at the end.

Composition of files.xhtml

A .xhtml file can basically be described as three types of xml structures:

- <body>....</body> which contains the texts and images to be represented, and the structure of how it represents them (paragraphs, tables, lists...)
- <style>....</style> which contains the styles applicable to <body>....</body>, such as fonts and font sizes, certain repetitive images, colors, alignments ...
- *"another .xhtml"* as <title>... </title>, namespaces, and other meta-information.

When combining multiple .xhtml source files in a .xhtml target file, on the one hand the area <body>....</body> of each .xhtml source file is combined, to produce a single <body>....</body> in the .xhtml target file, since there can only be a single structure <body>....</body> in a .xhtml file

A similar combination is made for the <style> zone. And certain duplicate parts of each source file (such as the <body></body> tags themselves) simply disappear as redundant.

Each .xhtml has been deconstructed into several zones, which have been independently recombined in the .xhtml target. You can no longer recalculate the SA256 hash code of each .xhtml source file looking at the .xhtml target file.

PROBLEM: How to assure the integrity of each .xhtml or .html source file when combined into a single .xhtml target file?

Combination of source files.xhtml with <style> and <body> invariant

A simple procedure to obtain a .xhtml target file is to concatenate the contents of the structure <body>PAYLOADx</body> of each and every one of the .xhtml source file in a single structure <body>PAYLOAD1PAYLOAD2PAYLOADn</body> in the .xhtml target file. Each .xhtml file has one and only one <body>x</body> structure.

The size and SHA256 of the PAYLOADx of each .xhtml source file are saved.

Knowing the size of each PAYLOADx, and knowing the order that they are concatenated (one placed just after others), the SHA256 of each PAYLOADx is recalculated in the structure <body>PAYLOAD1PAYLOAD2PAYLOADn</body> of the .xhtml target file.

If the calculated SHA256 matches the saved SHA256, then it is demonstrated that the respective PAYLOADx have not changed.

The same mechanics can be followed for <style>PAYLOADx</style> structures. Each .xhtml source file has zero, one or more structures <style>PAYLOADxy</style>. First all the <style>PAYLOADxy</style> structures of each .xhtml source file are concatenated as if they were a single <style>PAYLOADx</style> structure (which may be empty). Its size and its SHA256 are saved (if the size is greater than zero). And it is concatenated in the structure <style>PAYLOAD1PAYLOAD2PAYLOADn</style> of the .xhtml target file. Knowing the sizes, you can recalculate the SHA256 and check that the contents of the .xhtml target file is invariant with respect to the concatenation of .xhtml source files.

As the .xhtml presentation is made from texts and images (<body> zone) and styles to be applied (<style> zone), with this SHA256 hash codes mechanism you can check that the .xhtml source files are combined in the .xhtml target file without any alteration.

Combination of .xhtml source files with <style> and <body> non-invariant: "distilled"

There may be cases where there are variations in the <style>PAYLOADx</style> and/or <body>PAYLOADx</body> zones of a .xhtml source.

For example, it is a .html to be transformed to .xhtml (see Chapter *Conversion .html to .xhtml*). Or that certain definitions of styles in the file affect other .xhtml source files (see Chapter *Disambiguation of styles .xhtml*).

In these cases, there can be a multitude of changes within xml structures, for example if a style is renamed to disambiguate it, or if the values of the attributes are put in quotation marks according to the .xhtml standard format.

If character sequences in xml structures vary, sizes vary, and SHA256 hash codes vary.

What does not vary are the texts or the images. Transformations in .xhtml meta-information (how texts and images are to be represented) do not alter what is the itself information of texts and images (their values). Where there is a 3, a 4 is not going to appear. And where there is an image of a car, the image of an airplane is not going to appear.

However, checking that there have been no changes in texts and images with a classic character-to-character comparison (such as comparing documents in Word) detects so many differences that it is not useful.

The proposed solution is to ignore the .xhtml structures and take as invariants only text and images.

Even if the display styles or parts of the tagging varies in the .xhtml or .html file, it must not vary texts or images. And if text and images do not vary, text and images can be "distilled" (extracted in order) from the <body>PAYLOAD</body> structure of a .xhtml (or .html) file to a working "distilled" file.

The sizes and the SHA256 hash codes of the working "distilled" file corresponding to each .xhtml or .html source file are saved.

The .xhtml target file is created by concatenation of the <body></body> structures of the source files. The resulting "distilled" target file will in turn be the concatenation of the "distilled" source files, since neither text nor images vary in the combination process.

Knowing the size and SHA256 of each "distilled" source file, it can be verified that they are invariant in the resulting "distilled" target file.

Once the Annual Report is prepared, its "distilled" content is split according to the "distilled" size of each Chapter that composes it. The SHA256 of each piece has to match the SHA256 of the "distillate" of the respective Chapter. In this way it is assured that the .xhtml Annual Report is precisely the concatenation of the.xhtml or .html Chapters that compose it, regardless of any changes in xml structures that may have occurred. Any alteration will cause the SHA256 to no longer match.

In theory, there are possibilities to alter the visualization in such a way that it seems that the content is different. It may be an interesting subject of academic research, but it is impractical, since the manipulation would be of limited effects and too notorious.

A PRACTICAL SOLUTION:

easyCOMPOSER calculates the SHA256 sizes and hash codes of the <style>, <body> and *"distilled"* (only text and images) of each source file to be combined, and also of the .xhtml target file combined. This information is placed as a non-visible comment at the end of the target file combined. The SHA256 hash code of the <style> and <body> of each .xhtml source file to be combined is identical to the SHA256 hash code of the corresponding zone in the <style> and <body> of the target file combined. The SHA256 hash code of the *"distillate"* of each source file .xhtml or .html to be combined is identical to the SHA256 hash code of the *"distillate"* of the target file combined, although when combining certain tags or attributes may have been modified in <style> and/or <body>. See details in chapter *SHA256 hash* codes *in .xhtml files*

easyCOMPOSER calculates and shows the sizes and the SHA256 hash codes of the <style>, <body> and *"distilled"* of each file indicated in the *HashCodes* sheet, and checks that they match if it is a combined file. The utility is that it can be audited in parallel Chapter by Chapter in .xhtml .html with integrity assurance. The SHA256 hash codes of each Chapter are saved and it can be verified that their SHA256 hash codes have not changed in the Annual Report. In fact, the SHA256 of the *"distillate"* (text and images only) does not vary even if ***easyCOMPOSER*** has converted the .html file to .xhtml or had to adjust styles.

easyCOMPOSER calculates each time each SHA256 hash code and checks (if there is enough information) that matches those declared in the meta information of the composed file .xhtml by putting an ***OK*** to the right of the SHA256. If it does not match, it displays the discrepant SHA256 or the detected error.

HashCodes Sheet

Folder with files: The cell below specifies the *path* of a default directory where:

- There are the files and subdirectories that do not have their *path* specified
- The necessary work files will be created and deleted.

List of files (and folders) for hash codes: In the cells below are placed the files of which the SHA256 hash codes will be show. If a subdirectory is put in, all the files it contains will be processed.

Click Run • to run

File name: Name of the file or subdirectory processed.

- If it is a .zip file or a directory, it expands and processes all the files and directories it contains.
- Indenting (putting the file name to the right) is used to display the hierarchical order within the file or directory.
- If it is a composed .xhtml file, the following lines have the name in parentheses of each file that is part of the composition.
    - The first file is the composite file itself, but without the composition meta-information (about 500 bytes).
    - The following are each source file that make up the composite file.

File size: File size in bytes.

File SHA256: SHA256 hash code of the set of all bytes of the file.

Body size: Size of the <body> zone in bytes.

Body SHA256: SHA256 hash code of the set of all bytes in the <body> zone.

Style size: Size of the <style> zone in bytes.

Style SHA256: SHA256 hash code of the set of all bytes in the <style> zone.

Distilled size: Size of the <body> zone "*distilled"* (text and images only) in bytes.

Distilled SHA256: SHA256 hash code of all bytes in the <body> zone *"distilled"* (text and images only).

☑***Check for export distilled as file .distilled.txt*** If actived, a file is exported containing the <body> zone "*distilled"* (text and images only) of the file. The export file is created in the default directory, and will keep the same file name by adding at the end "*.distilled.txt*"

Distillation algorithm (texts and images only) of an .xhtml or .html file:

1. Delete from the beginning of the file to the ***<body....>*** tag
2. Delete from the ***</body>*** tag to the end of the file
3. Delete all comments ***<!-- ... -->***
4. Replace each image ***<img.... src="data:image/..........." />*** by only ***data:image/...........***
5. Clear all the ***<tag....>*** and ***</tag>*** XML tags
6. Clear all character escape sequences ***&xxx;*** such as &nbsp; or &#160;
7. Clear carriage returns, line breaks, tabs, and spaces.

In the end there is an invariant *"distilled"* content, something like:

Memoryofthe exercise2020data:image/OrK....5CYII=AnnualAccountsoftheexercise.....

The *"distillation"* algorithm can be programmed in any language or with regular expressions. The character encoding is UTF8 according to the ESEF format.

Save SHA256 sizes and hash codes

It is recommend to use the following .xhtml comment structure to save size and SHA256 hash code information. This structure is repeated for each source file and for the target file itself. These structures are placed after </body> in the target file.

```
<!--hashfile order="" file=""
filesize="" filehash=""
bodysize="" bodyhash=""
stylesize="" stylehash=""
distilledsize="" distilledhash=""
-->
```

order="" Required. Sequence of digits 0.. 9 not repeated. Indicates the order of this source file within the target file. order="0" indicates target file.

file="" Required. File name, with no path

filesize="" Optional. File size. In the case of order="0", the file size is up to the tag </body> inclusive.

filehash="" Optional. Only if filesize > 0. SHA256 hash code of that file. In the case of order="0", the file size is up to the tag </body> inclusive.

bodysize="" Optional. Size of the PAYLOAD in the file structure <body... >PAYLOAD</body>

bodyhash="" Optional. Only if bodysize > 0. SHA256 hash code of PAYLOAD in the file structure <body...>PAYLOAD</body>

stylesize="" Optional. Size of the PAYLOAD in the structure <style... >PAYLOAD</style> of the file, concatenated if there are several.

stylehash="" Optional. Only if stylesize > 0. SHA256 hash code of the PAYLOAD in the file structure <style... >PAYLOAD</style>, concatenated if there are several.

distilledsize="" Optional. Size of the "distillate" of the PAYLOAD in the file structure <body...>PAYLOAD</body>.

distilledhash="" Optional. Only if distilledsize > 0. SHA256 hash code of the PAYLOAD "distillate" in the file structure <body...>PAYLOAD</body>

Example:

```
</body><!--hashfile order="0" file="company-2021-12-31.xhtml"
filesize="2051206" filehash="TockHgZ0BgsQCgLEx13ZsWdNLCqlDmFKC5rwR6SBVyQ="
bodysize="2042432" bodyhash="zxm9BacGlski1dbp+DfQ/zVuj2agm9rmaVIXu9dI36I="
stylesize="5580" stylehash="Bw+W0qjof9UuTfJasOUOpNV32d7ZBXRbotth+uqtTSA="
distilledsize="133796"
distilledhash="jmc62732j6gzjXcA8GvgEn7mTRghJmLMQYby2ZW8Q2U="   -->
```

Pre-constituted evidence:

A simple way to indelibly save SHA256 is to send them as a message via Whatsapp, Instagram, Twitter or another secure messaging service. Both sender and recipient (or anyone else who had access) could use that message as pre-constituted judicial evidence. [4]

[4] Can be used also digital signature, or advanced solutions as sigstore.dev (software transparent signature) or openfiling.info/blockchain (signature blockchain, of the same author of this document)