

TZWorks® Chromium Cache Parser (*ccp*) Users Guide



Abstract

ccp is a standalone, command-line tool that parses cache files associated with the Chromium-based browsers. The tool can target various cache types and either report the results in a CSV type format or archive the results in a SQLite database. This tool has working versions for Windows, Linux and macOS.

Copyright © TZWorks LLC

www.tzworks.com

Contact Info: info@tzworks.com

Document applies to v0.14 of ***ccp***

Updated: Apr 15, 2024

Table of Contents

1	Introduction	2
1.1	Chromium Cache structure	3
1.1.1	Disk Cache	3
1.1.2	Simple Cache	4
1.2	Chromium-based Browsers.....	5
1.3	Location of the Cache data	5
2	How to Use <i>ccp</i>	8
2.1	Targeting Specific Cache files	9
2.2	Processing Cache Files in one or more Subdirectories	10
2.2.1	Archiving the Content Data.....	10
3	CSV Field Names / Meaning	11
4	Limitations.....	12
5	Available Options	13
6	Authentication and the License File.....	14
7	References	14

TZWorks® Chromium Cache Parser (*ccp*)

Users Guide

Copyright © TZWorks LLC

Webpage: http://www.tzworks.com/prototype_page.php?proto_id=54

Contact Information: info@tzworks.com

1 Introduction

The Chromium Cache Parser (*ccp*) targets various caches associated with Chromium-based browsers, or browsers that use the cache component of Chromium. This tool addresses parsing caches in these browsers (at least the later versions of these browsers): *Google Chrome*, *Microsoft Edge*, *Opera*, *Brave*, *Vivaldi*, and others.

Browser cache files contain useful information for the examiner. The *ccp* tool is not unique, in that there are other Chromium-based cache parsers available; a few are even free. This tool was primarily created based on a need to provide more insight into the association of the cache metadata (eg. timestamps, URL, http request/response, etc) and cache content data (eg. data for the webpage that is displayed), especially when applied to the various cache types that Chromium can use. In addition, and from a tool developer's standpoint, the *ccp* codebase can be used as framework for future prototyping work to evaluate Chromium cache artifact data that may be corrupted or fragmented.

As background, the Chromium cache, is a repository for web data a user has viewed or downloaded. In general, the purpose of the cache is to store data locally, and thus allow the browser quick access for later requests to a previously viewed website. The cache includes: website pages, files, scripts, images and other items that were viewed by a user or data that the browser needed to use. In addition to the raw data that was received from a web server, the *cache* also contains useful metadata associated with each item. From the point of view of the forensic examiner the cache provides insights to the user's Internet usage, since it contains items such as: the URL of the webpage, number of times the page was fetched from the *cache*, filename/type/size, last modified time, last fetched time, server time, etc. Having a tool available that can take advantage of this artifact data is necessary to have insights into the user's activity.

The Chromium cache can consist of various types of cache; each type can be determined by where it is located in the Chromium directory structure. These various types of cache, include: *normal Cache* data, *CacheStorage* type data, *Code Cache*, and *ScriptCache*, to name a few. A listing of all the cache types is shown below and is taken from the Chromium project's documentation. The **ccp** tool has only been tested on a few of these types, primarily due to the limited test samples available.

Cache Type	Meaning
DISK_CACHE	Disk is used as the backing storage
MEMORY_CACHE	Data is stored only in memory
REMOVED_MEDIA_CACHE	No longer in use
APP_CACHE	Special case of DISK_CACHE. cache storage, service worker script cache
SHADER_CACHE	Backing store for the GL shader cache
PNACL_CACHE	Backing store for the Portable native client translation cache
GENERATED_BYTE_CODE_CACHE	Backing store for renderer generated data like bytecode for JavaScript
GENERATED_NATIVE_CODE_CACHE	Backing store for renderer generated data native code for WebAssembly
GENERATED_WEBUI_BYTE_CODE_CACHE	Backing store for renderer generated data bytecode for JavaScript from WebUI pages

As a side note, not-withstanding that the above are the cache types, when the cache location is different than the traditional cache location (*Cache_Data*), it will be named using the location of where the cache was found. For example, if the cache was in the *scriptcache* section, it will be labelled as '*scriptcache*', or *GPUCache* if in the *GPUCache* section, etc. Below is the how the **ccp** tool maps how it labels it cache type to where the cache is located.

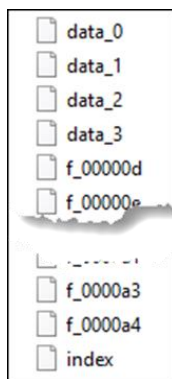
Cache Type labeled in <i>ccp</i> tool	location
disk cache	../Cache/Cache_Data
storage cache	../Service Worker/CacheStorage/..
code cache	../Code Cache/..
script cache	../Service Worker/ScriptCache/..
GPU cache	../GPUCache/..

1.1 Chromium Cache structure

The Chromium cache documentation discusses 2 main flavors of cache structure: (a) Disk Cache [2] and (b) Simple Cache [3].

1.1.1 Disk Cache

The *disk cache* [2] consists of at least five files: an *index* file and four data files known as block files. The *index* file keeps track of the block files by managing the start of few linked lists that point to a *ranking* node, which in turn points to a previous or next *ranking* node. Block files contain records of fixed sized data, where *data_0* has the smallest chunk, and incrementally increases to the largest chunk in *data_3*. Cache entries that are larger than a multiple of 4 of the largest block size gets relegated to its own separate file. These separate files are prefixed with an *f_* followed by 6 characters that represent 3 hexadecimal bytes (eg. *f_00036d*); this name is then referenced back in one or more of the block files to maintain linkage. Below is a screenshot of the files created with a *disk cache*.



The block sizes and the maximum cache size for the respective *data_x* files are shown below.

File	Block size (bytes)	Max Data size (bytes)
data_0	36	ranking nodes
data_1	256	1K
data_2	1K	4K
data_3	4K	16K
f_XXXXXX	> 16K	separate files

From the table, the *data_0* file is usually dedicated to the *rankings* node data; the rankings node data is used internally for identifying the least recently used (LRU) cache item for the eviction algorithm. This algorithm determines which space can be reused by overwriting an old cache entry. From a forensics standpoint, this metadata in the ranking node is quite useful, since it contains *last access time* and *last modified time* of the cache entry in question.

1.1.2 Simple Cache

The *simple cache* [3] process creates a separate file for each cache entry. The internal structure of the *simple cache* is organized by using predefined key patterns to allow the metadata data and content data to be sandwiched, such as the browser request, the server's response, as well as, transaction times, IP address, etc. More information on this cache structure can be found by reviewing Chromium's documentation on their website.

1.2 Chromium-based Browsers

While the table below is not a complete list, the intent here was to identify some of the more popular browsers that use some form of the Chromium-based cache architecture. These are the ones that were used when testing out the **ccp** tool. Of those listed, all have browsers operate on at least the following desktop operating systems: Windows, Linux and macOS. Many of these same browsers work on iOS or Android as well.

Browser	Website	Popular Desktop OS's used on
Google Chrome	https://www.google.com/chrome	Windows, Linux and macOS
Brave	https://brave.com/download	Windows, Linux and macOS
Microsoft Edge (newer version)	https://www.microsoft.com/en-us/edge	Windows, Linux and macOS
Opera (newer version)	https://www.opera.com/download	Windows, Linux and macOS
Vivaldi	https://vivaldi.com/download	Windows, Linux and macOS

1.3 Location of the Cache data

Chromium-based cache artifacts are located in the user's directory. It was not obvious when developing the **ccp** tool just how many places Chromium-based cache files can be found. If one runs the **ccp** tool while enumerating all the files in the main browser's directly, the tool does a good job at identifying those locations where it finds a cache file, which in turn is reflected in the output reported to the user.

The actual location in this directory varies depending on the operating system used. Below is a table that breaks out the location by operating system. As a disclaimer, the list of cache locations shown below is not complete. What is shown are only those locations when installing a browser and looking at where the cache artifacts were placed. As the browser is used more extensively, other locations are populated and these may or may not be shown below.

OS	Cache location
Win XP	%userprofile%\Local Settings\Application Data\Google\Chrome\User Data\Default
Post XP Win	%userprofile%\AppData\Local\Google\Chrome\User Data\Default\Cache %userprofile%\AppData\Local\Google\Chrome\User Data\Default\GPUCache %userprofile%\AppData\Local\Google\Chrome\User Data\Default\Code Cache %userprofile%\AppData\Local\Google\Chrome\User Data\Default\Service Worker\ScriptCache %userprofile%\AppData\Local\Google\Chrome\User Data\Default\Service Worker\CacheStorage %userprofile%\AppData\Local\Google\Chrome\User Data\Default\Storage\ext\<random>\def\Code Cache %userprofile%\AppData\Local\Google\Chrome\User Data\Default\Storage\ext\<random>\def\GPUCache %userprofile%\AppData\Local\Google\Chrome\User Data\ShaderCache\GPUCache %userprofile%\AppData\Local\Google\Chrome\User Data\GrShaderCache\GPUCache %userprofile%\AppData\Local\Microsoft\Edge\User Data\Default\Cache %userprofile%\AppData\Local\Microsoft\Edge\User Data\Default\GPUCache %userprofile%\AppData\Local\Microsoft\Edge\User Data\Default\Code Cache %userprofile%\AppData\Local\Microsoft\Edge\User Data\Default\Service Worker\ScriptCache %userprofile%\AppData\Local\Microsoft\Edge\User Data\Default\Service Worker\CacheStorage %userprofile%\AppData\Local\Microsoft\Edge\User Data\Default\Storage\ext\<random>\def\Code Cache %userprofile%\AppData\Local\Microsoft\Edge\User Data\Default\Storage\ext\<random>\def\GPUCache %userprofile%\AppData\Local\Microsoft\Edge\User Data\ShaderCache\GPUCache %userprofile%\AppData\Local\Microsoft\Edge\User Data\GrShaderCache\GPUCache %userprofile%\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\Cache %userprofile%\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\GPUCache

	%userprofile%\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\Code Cache %userprofile%\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\Service Worker\ScriptCache %userprofile%\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\Service Worker\CacheStorage %userprofile%\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\Storage\ext\<random>\def\Code Cache %userprofile%\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\Storage\ext\<random>\def\GPUCache %userprofile%\AppData\Local\BraveSoftware\Brave-Browser\User Data\ShaderCache\GPUCache %userprofile%\AppData\Local\BraveSoftware\Brave-Browser\User Data\GrShaderCache\GPUCache %userprofile%\AppData\Local\Vivaldi\User Data\Default\Cache %userprofile%\AppData\Local\Vivaldi\User Data\Default\GPUCache %userprofile%\AppData\Local\Vivaldi\User Data\Default\Code Cache %userprofile%\AppData\Local\Vivaldi\User Data\Default\Service Worker\ScriptCache %userprofile%\AppData\Local\Vivaldi\User Data\Default\Service Worker\CacheStorage %userprofile%\AppData\Local\Vivaldi\User Data\Default\Storage\ext\<random>\def\Code Cache %userprofile%\AppData\Local\Vivaldi\User Data\Default\Storage\ext\<random>\def\GPUCache %userprofile%\AppData\Local\Vivaldi\User Data\ShaderCache\GPUCache %userprofile%\AppData\Local\Vivaldi\User Data\GrShaderCache\GPUCache %userprofile%\AppData\Local\Opera Software\Opera Stable\Cache\Cache_Data %userprofile%\AppData\Local\Opera Software\Opera Stable\System Cache\Cache_Data %userprofile%\AppData\Roaming\Opera Software\Opera Stable\Code Cache %userprofile%\AppData\Roaming\Opera Software\Opera Stable\GPUCache %userprofile%\AppData\Roaming\Opera Software\Opera Stable\GrShaderCache %userprofile%\AppData\Roaming\Opera Software\Opera Stable\ShaderCache
OSX	/Users/[user acct]/Library/Caches/Google/Chrome/Default/Cache /Users/[user acct]/Library/Caches/Google/Chrome/Default/Code Cache /Users/[user acct]/Library/Caches/Google/Chrome/Default/Storage/ext/<random>/def/Cache /Users/[user acct]/Library/Caches/Google/Chrome/Default/Storage/ext/<random>/def/Code Cache /Users/[user acct]/Library/Caches/Google/Chrome/Default/Storage/ext/<random>/def/GPUCache /Users/[user acct]/Library/Application Support/Google/Chrome/Default/GPUCache /Users/[user acct]/Library/Application Support/Google/Chrome/Default/Storage/ext/<random>/def/Code Cache /Users/[user acct]/Library/Caches/Microsoft Edge/Default/Default/Cache /Users/[user acct]/Library/Caches/Microsoft Edge/Default/Default/Code Cache /Users/[user acct]/Library/Caches/com.microsoft.edgemac/Cache.db /Users/[user acct]/Library/Caches/Microsoft Edge/Default/Storage/ext/<random>/def/Code Cache /Users/[user acct]/Library/Caches/Microsoft Edge/Default/Storage/ext/<random>/def/GPUCache /Users/[user acct]/Library/Caches/BraveSoftware/Brave-Browser/Default/Cache /Users/[user acct]/Library/Caches/BraveSoftware/Brave-Browser/Default/Code Cache /Users/[user acct]/Library/Caches/com.brave.browser/Cache.db /Users/[user acct]/Library/Application Support/BraveSoftware/Default/GPUCache /Users/[user acct]/Library/Application Support/BraveSoftware/Default/Service Worker/CacheStorage /Users/[user acct]/Library/Application Support/BraveSoftware/Default/Service Worker/ScriptCache /Users/[user acct]/Library/Caches/Vivaldi/Default/Cache/Cache_Data /Users/[user acct]/Library/Caches/Vivaldi/Default/Code Cache /Users/[user acct]/Library/Caches/Vivaldi/Default/Storage/ext/<random>/def/Code Cache /Users/[user acct]/Library/Application Support/Vivaldi/Default/GPUCache /Users/[user acct]/Library/Application Support/Vivaldi/GrShaderCache/GPUCache /Users/[user acct]/Library/Application Support/Vivaldi/ShaderCache/GPUCache /Users/[user acct]/Library/Application Support/Vivaldi/Default/Storage/ext/<random>/def/GPUCache /Users/[user acct]/Library/Caches/com.operasoftware.Opera/Cache /Users/[user acct]/Library/Caches/com.operasoftware.Opera/Code Cache /Users/[user acct]/Library/Caches/com.operasoftware.Opera/System Cache /Users/[user acct]/Library/Application Support/com.operasoftware.Opera/GPUCache /Users/[user acct]/Library/Application Support/com.operasoftware.Opera/ShaderCache /Users/[user acct]/Library/Application Support/com.operasoftware.Opera/GrShaderCache
Linux (Ubuntu)	/home/[user acct]/.cache/google-chrome/Default/Cache /home/[user acct]/.cache/google-chrome/Default/Code Cache /home/[user acct]/.cache/google-chrome/Default/Storage/ext/<random>/def/Cache /home/[user acct]/.cache/google-chrome/Default/Storage/ext/<random>/def/Code Cache /home/[user acct]/.config/google-chrome/Default/GPUCache /home/[user acct]/.config/google-chrome/Default/Storage/ext/<random>/def/GPUCache /home/[user acct]/.config/google-chrome/GrShaderCache/GPUCache /home/[user acct]/.config/google-chrome/ShaderCache/GPUCache /home/[user acct]/.cache/microsoft edge/default/Default/Cache /home/[user acct]/.cache/microsoft edge/default/Default/Code Cache /home/[user acct]/.cache/microsoft edge/default/Default/Storage/ext/<random>/def/Cache /home/[user acct]/.cache/microsoft edge/default/Default/Storage/ext/<random>/def/Code Cache /home/[user acct]/.config/ microsoft edge/Default/GPUCache /home/[user acct]/.config/ microsoft edge/Default/Service Worker/CacheStorage /home/[user acct]/.config/ microsoft edge/Default/Service Worker/ScriptCache

```

/home/[user acct]/.config/microsoft edge/Default/Storage/ext/<random>/def/GPUCache
/home/[user acct]/.config/microsoft edge/GrShaderCache/GPUCache
/home/[user acct]/.config/microsoft edge/ShaderCache/GPUCache

/home/[user acct]/.config/BraveSoftware/Brave-Browser/GrShaderCache/GPUCache
/home/[user acct]/.config/BraveSoftware/Brave-Browser/ShaderCache/GPUCache
/home/[user acct]/.config/BraveSoftware/Brave-Browser/Default/GPUCache
/home/[user acct]/.config/BraveSoftware/Brave-Browser/Default/Service Worker/ScriptCache
/home/[user acct]/.config/BraveSoftware/Brave-Browser/Default/Service Worker/CacheStorage
/home/[user acct]/.cache/BraveSoftware/Brave-Browser/Default/Cache
/home/[user acct]/.cache/BraveSoftware/Brave-Browser/Default/Code Cache
/home/[user acct]/snap/brave/[#]/.config/BraveSoftware/Brave-Browser/Default/GPUCache
/home/[user acct]/snap/brave/[#]/.config/BraveSoftware/Brave-Browser/GrShaderCache/GPUCache
/home/[user acct]/snap/brave/[#]/.config/BraveSoftware/Brave-Browser/ShaderCache/GPUCache
/home/[user acct]/snap/brave/common/BraveSoftware/Brave-Browser/Default/Cache
/home/[user acct]/snap/brave/common/BraveSoftware/Brave-Browser/Default/Code Cache

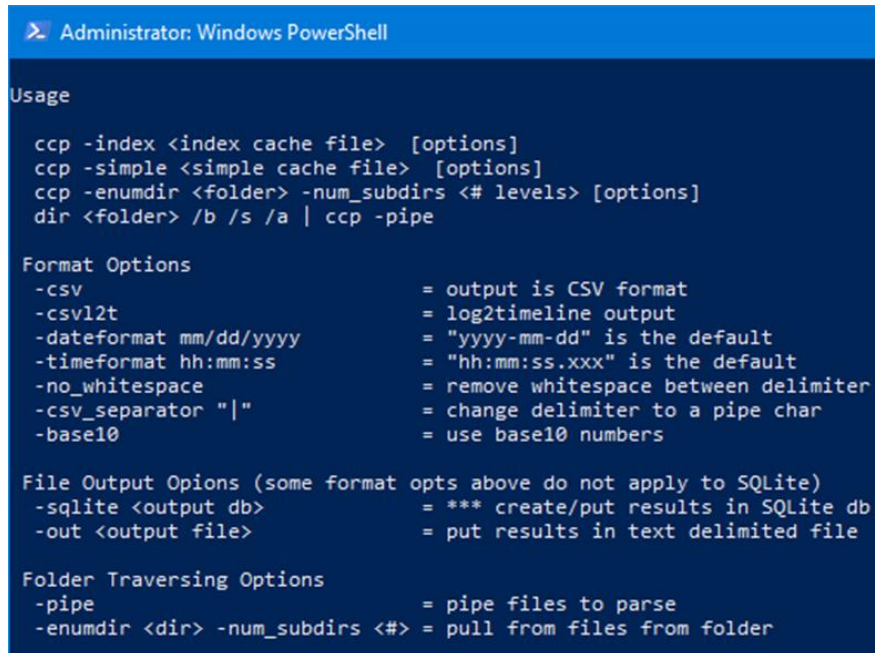
/home/[user acct]/.config/Opera/GPUCache
/home/[user acct]/.config/Opera/GrShaderCache/GPUCache
/home/[user acct]/.config/Opera/ShaderCache/GPUCache
/home/[user acct]/.config/Opera/Service Worker/ScriptCache
/home/[user acct]/.cache/opera/Cache
/home/[user acct]/.cache/opera/Code Cache
/home/[user acct]/.cache/opera/System Cache
/home/[user acct]/snap/opera/[#]/.config/Opera/GPUCache
/home/[user acct]/snap/opera/[#]/.config/Opera/GrShaderCache/GPUCache
/home/[user acct]/snap/opera/[#]/.config/Opera/ShaderCache/GPUCache
/home/[user acct]/snap/opera/common/opera/Cache
/home/[user acct]/snap/opera/common/opera/Code Cache
/home/[user acct]/snap/opera/common/opera/System Cache

/home/[user acct]/.cache/vivaldi/Default/Cache
/home/[user acct]/.cache/vivaldi/Default/Code Cache
/home/[user acct]/.cache/vivaldi/Default/Storage/ext/<random>/def/Cache
/home/[user acct]/.cache/vivaldi/Default/Storage/ext/<random>/def/Code Cache
/home/[user acct]/.config/vivaldi/GrShaderCache/GPUCache
/home/[user acct]/.config/vivaldi/ShaderCache/GPUCache
/home/[user acct]/.config/vivaldi/Default/GPUCache
/home/[user acct]/.config/vivaldi/Default/Service Worker/CacheStorage
/home/[user acct]/.config/vivaldi/Default/Service Worker/ScriptCache
/home/[user acct]/.config/vivaldi/Default/Storage/ext/<random>/GPUCache

```


2 How to Use *ccp*

The screenshot below shows the options available. The formatting options are similar to the rest of the *TZWorks* tools. The output can be rendered in either: delimited text (CSV and Log2Timeline) or SQLite. The SQLite option was added primarily to allow one to parse the cache records into their metadata components while archiving the companion cache content data with the metadata results.



```
Administrator: Windows PowerShell

Usage

ccp -index <index cache file> [options]
ccp -simple <simple cache file> [options]
ccp -enumdir <folder> -num_subdirs <# levels> [options]
dir <folder> /b /s /a | ccp -pipe

Format Options
-csv                      = output is CSV format
-csvl2t                  = log2timeline output
-dateformat mm/dd/yyyy  = "yyyy-mm-dd" is the default
-timeformat hh:mm:ss    = "hh:mm:ss.xxx" is the default
-no_whitespace           = remove whitespace between delimiter
-csv_separator "|"      = change delimiter to a pipe char
-base10                  = use base10 numbers

File Output Options (some format opts above do not apply to SQLite)
-sqlite <output db>      = *** create/put results in SQLite db
-out <output file>       = put results in text delimited file

Folder Traversing Options
-pipe                    = pipe files to parse
-enumdir <dir> -num_subdirs <#> = pull from files from folder
```

To process cache files, *ccp* can either target a folder (which is the preferred method) or an individual cache file. The tool will automatically determine which version of the format the cache files are in and adjust the parsing engine accordingly. In fact, when parsing many subdirectories of Chromium-based cache artifacts in one session, the tool will dynamically address each cache type it knows about, resulting in a cohesive mapping of artifact metadata to the *cache content* data sorted. The retention of the raw *cache content* is a benefit of using the SQLite option, since a separate table will be generated, one for the *cache metadata* and another for the *cache content* data. The mapping between the tables is implemented via a unique key value that correlates the records of the two tables.

If processing a directory of cache files (either by using the *-pipe* command or the *-enumdir* command), the user should point to the browser subdirectory that is desired to be parsed.

2.1 Targeting Specific Cache files

While not recommended for processing a collection of artifacts, if one only wants to target a specific cache file or a block cache, one can use the **-simple <simple cache file>** or the **-index <block cache index file>** option, respectively. These were left in as options primarily for debugging of the **ccp** tool. In the example below, we are targeting a disk cache folder by passing in the 'index' file. The results would be rendered in the **test.csv** file.

```
>ccp64 -index D:\test\Cache\Cache_Data\index -out test.csv
```

Pipe delimited text is the default output that is rendered by the tool. This can be adjusted to either a comma or tab character by using the **-csv_separator <delimiter>** option.

When looking at the output in a spreadsheet type app, on the left is the Chromium type and version of the cache format (major and minor version separated by a dot). Many of the other fields are the metadata associated with the Chromium cache internals, requesting a page/data and the server serving up the webpage/data.

Shown in the screenshot is only the server timestamp, but also in the data (but truncated in the screenshot), is up to 6 – 8 timestamps that range from those associated with the server, browser and file timestamps. There is a mix of JSON-like fields in the sense they contain key/value pairs. In this way all the parsed data could be rendered in a CSV type format where each record is delimited.

cache_version	http_response_status	serv_name	serv_timezone	serv_date	sc	fetch count	url	URL visited
chromium disk cache [2.0]	HTTP/1.1 200 OK	Microsoft-IIS/10.0	GMT	03/22/2022 02:28:31		17	https://api.edgeoffer.microsoft.com/edgeoffer/experiments?appid=ed	
chromium disk cache [2.0]	HTTP/1.1 200 OK	Windows-Azure-Blob/1.0	GMT	01/20/2022 14:49:16	01	1	https://assets.msn.com/bundles/v1/edgeChromium/latest/vendors.dff5b21	
chromium disk cache [2.0]	HTTP/1.1 200 OK	Windows-Azure-Blob/1.0	GMT	01/20/2022 14:49:16	01	1	https://assets.msn.com/bundles/v1/edgeChromium/latest/common.b23336	
chromium disk cache [2.0]	HTTP/1.1 200 OK	Windows-Azure-Blob/1.0	GMT	01/20/2022 14:49:16	01	1	https://assets.msn.com/bundles/v1/edgeChromium/latest/experience.97b3	
chromium disk cache [2.0]	HTTP/1.1 200 OK	Windows-Azure-Blob/1.0	GMT	01/20/2022 14:49:15	01	8	https://ntp.msn.com/bundles/v1/edgeChromium/latest/web-worker.912	
chromium disk cache [2.0]	HTTP/1.1 200 OK	Windows-Azure-Blob/1.0	GMT			1	https://r.bing.com/rp/Rls5OrCOL-c1q4RObEplaxSyyuKE.svg	
Cache type/version	TP/1.1 200 OK	Kestrel	GMT	Server timestamp		2	https://ntp.msn.com/resolver/api/resolve/v3/config?expType=AppCo	

content_type	content filename	content encoding	content size	extra fields	Field populated with a JSON-like key/value pairs
application/json; charset=	experiments	gzip	0	["server_url":"https://microsoft.com";server_response=["ip_addr":"40.71.99.188";"request-context":"ap	
application/javascript	vendors.dff5b21	gzip	27033	["server_url":"https://msn.com";server_response=["ip_addr":"23.50.113.135";"access-control-allow-cr	
application/javascript	common.b23336	gzip	227649	["server_url":"https://msn.com";server_response=["ip_addr":"23.50.113.135";"access-control-allow-cr	
application/javascript	experience.97b3	gzip	34003	["server_url":"https://msn.com";server_response=["ip_addr":"23.50.113.135";"access-control-allow-cr	
application/javascript	web-worker.912	gzip	38934	["server_url":"https://msn.com";server_response=["ip_addr":"204.79.197.203";"accept-ch":"Sec-CH-UA	
image/svg+xml	Rls5OrCOL-c1q4f		924	["server_url":"https://bing.com";server_response=["ip_addr":"13.107.21.200";"accept-ch":"Sec-CH-UA	
application/json; charset=		gzip	42848	["server_url":"https://msn.com";isolation_key":"_dk_";server_response=["ip_addr":"204.79.197.203";	
application/json; charset=	appanon		2	["server_url":"https://msn.com";server_response=["ip_addr":"204.79.197.203";"accept-ch":"Sec-CH-UA-A	

2.2 Processing Cache Files in one or more Subdirectories

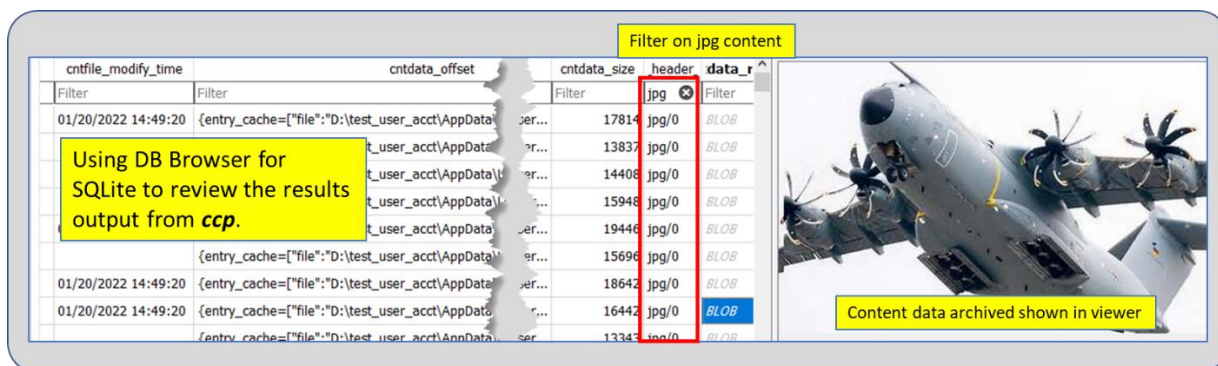
If desiring to process many Chromium cache files in one pass, one can make use of **ccp's** piping option (**-pipe**) or the folder enumeration option (**-enumdir**). This is the more typical use of the **ccp** tool. Either of these options allow one to target multiple subdirectories during the parsing operation. Below is a simple way to target the cache files in an extracted account. In this case the output is sent to a SQLite file named 'results1.sqlite'

```
>dir D:\test_user_acct /b /s /a | ccp64 -pipe -sqlite results1.sqlite
```

If desiring more control on the number of subdirectories to traverse, one can use the **-enumdir** option along with the **-num_subdirs** sub-option. It would look like this for the above example:

```
>ccp64 -enumdir D:\test_user_acct -num_subdirs 15 -sqlite result2.sqlite
```

Below is an example of the SQLite output when rendered with the DB Browser for SQLite. Notice the actual content of the webpage is archived in the *cache_ctxdata_entries* database table using this option.



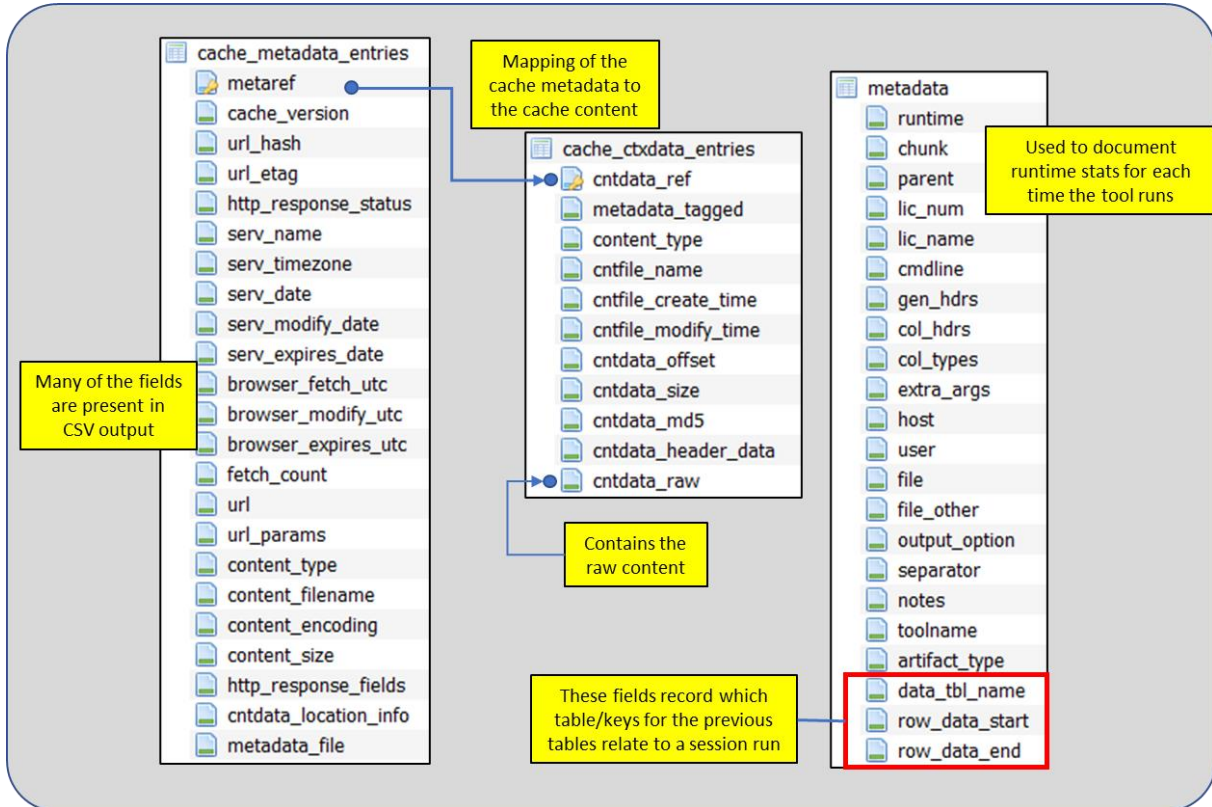
2.2.1 Archiving the Content Data

With the default option, the tool sends the parsed output to delimited text. This is fine when only wanting the results associated with the metadata such as URLs visited, timestamps of the visit, etc. If desiring to archive the content data as well, then one would run the tool with the ***-sqlite <db_name>*** option, which then tells ***ccp*** to create a database table for the metadata and a database table for the content data.

To view the results, one will need to be familiar with the SQL syntax to query the database, or alternatively, will need a separate SQLite viewer to look at the data. A good SQLite viewer is the “DB Browser for SQLite” and a reference is located at the end of this document.

The database schema created by **ccp** consists of 4 tables: (a) *cache_metadata_entries*, (b) *cache_ctxdata_entries*, (c) *metadata* and (d) *ref*. Only the first two tables have the records from the

parsed metadata and content data, respectively. The *metadata* table is used to record the session parameters used when running the parser. The last table (*ref*), is not shown in the diagram, and is used internally by *ccp* for bookkeeping only. The fields for the first three tables and their relationship are shown below.



The records in the *cache_metadata_entries* table are similar to the information rendered in the CSV output. The actual content data is stored in the *cache_ctxdata_entries* table under the field name "*ctxdata_raw*". This is a 'blob' type since the data can be either text or binary.

3 CSV Field Names / Meaning

Below is a reference of all the CSV fields used and their meanings.

CSV Field	Definition
cache_version	Cache type and version
url_hash	SHA1 hash of the key contained in the metadata. This is a computed value by <i>ccp</i> .
url_etag	The HTTP etag that was present in the HTTP response
request_type_reply_status	HTTP request type (eg. GET, POST), and reply status (eg. HTTP/1.1 200 OK)

serv_name	Server name recorded in the HTTP Response
serv_timezone	Server time zone
serv_date	Server timestamp included in the HTTP Response
serv_modify_date	Server modify timestamp included in the HTTP Response
serv_expires_date	Server expire timestamp included in the HTTP Response
browser_fetch_utc	Browser - last time the cache was fetched
browser_modify_utc	Browser modify timestamp associated with the cache
browser_expires_utc	Browser expire timestamp associated with the cache
content_create_utc	Actual content data file create timestamp. This is only present if the content file is a separate file. For Linux and OSX, this is the status change timestamp
content_modify_utc	Actual content data file modify timestamp. This is only present if the content file is a separate file.
fetch_count	Number of times the cache was fetched
url	URL of the webpage visited
url_params	Any URL parameters used. This is formatted as JSON.
content_type	The content data type (eg. GIF, JPEG, text, etc) extracted from the HTTP response
content_filename	Last part of the URL prior to the URL parameters extracted from the HTTP response
content_encoding	The encoding used on the content data (eg. gzip, br, etc) extracted from the HTTP response
content_size	Size of the content data extracted from the HTTP response
content_location_info	The file and offset (if not zero) within the file where the content data is located. This is formatted as JSON.
extra_fields	The key/value pairs extracted from the HTTP response. This is formatted as JSON.
file	The original path/file containing the metadata

4 Limitations

This version of the tool has a number of limitations. They are listed below.

- The tool is still prototype in nature being that this is the first version released. It still needs to be tested against various types of files, corrupted files, etc. to ensure the tool can perform consistently.

5 Available Options

Option	Description
-csv	Outputs the data fields delimited by commas. Since filenames can have commas, to ensure the fields are uniquely separated, any commas in the filenames get converted to spaces.
-csvl2t	Outputs the data fields in accordance with the log2timeline format.
-sqlite	Outputs the data into a SQLite database. The syntax is: -sqlite <db name to create or use> .
-pipe	Used to pipe files into the tool via STDIN (standard input). Each file passed in is parsed in sequence.
-enumdir	Experimental. Used to process files within a folder and/or subfolders. Each file is parsed in sequence. The syntax is -enumdir <folder> -num_subdirs <#> .
-filter	Filters data passed in via STDIN via the -pipe option. The syntax is -filter <"*.ext *partialname* ..."> . The wildcard character '*' is restricted to either before the name or after the name.
-no_whitespace	Only applies to -csv and -csvl2t options. Used in conjunction with -csv option to remove any whitespace between the field value and the CSV separator.
-csv_separator	Only applies to -csv and -csvl2t options. Used in conjunction with the -csv option to change the CSV separator from the default comma to something else. Syntax is -csv_separator "<char>" to change the CSV separator to the pipe character. To use the tab as a separator, one can use the -csv_separator "tab" OR -csv_separator "\t" options.
-dateformat	Output the date using the specified format. Default behavior is -dateformat "yyyy-mm-dd" . Using this option allows one to adjust the format to mm/dd/yy, dd/mm/yy, etc. The restriction with this option is the forward slash (/) or dash (-) symbol needs to separate month, day and year and the month is in digit (1-12) form versus abbreviated name form.
-timeformat	Output the time using the specified format. Default behavior is -timeformat "hh:mm:ss.xxx" . One can adjust the format to microseconds, via "hh:mm:ss.xxxxxx" or nanoseconds, via "hh:mm:ss.xxxxxxxxxx" , or no fractional seconds, via "hh:mm:ss" . The restriction with this option is a colon (:) symbol needs to separate hours, minutes and seconds, a period (.) symbol needs to separate the seconds and fractional seconds, and the repeating symbol 'x' is used to represent number of fractional seconds.
-quiet	Show no progress during the parsing operation.
-utf8_bom	All output is in Unicode UTF-8 format. If desired, one can prefix an UTF-8 <i>byte order mark</i> to the CSV output using this option.

6 Authentication and the License File

This tool has authentication built into the binary. The primary authentication mechanism is the digital X509 code signing certificate embedded into the binary (Windows and macOS).

The other mechanism is the runtime authentication, which applies to all the versions of the tools (Windows, Linux and macOS). The runtime authentication ensures that the tool has a valid license. The license needs to be in the same directory of the tool for it to authenticate. Furthermore, any modification to the license, either to its name or contents, will invalidate the license.

7 References

1. Chromium Design documents [<https://www.chromium.org/developers/design-documents>]
2. Chromium disk cache [<https://www.chromium.org/developers/design-documents/network-stack/disk-cache>]
3. Chromium simple cache [<https://www.chromium.org/developers/design-documents/network-stack/disk-cache/very-simple-backend>]
4. SQLite library statically linked into tool [Amalgamation of many separate C source files from SQLite version 3.32.3].
5. SQLite documentation [<http://www.sqlite.org>].
6. DB Browser for SQLite [<http://sqlitebrowser.org/>]