Binary Markup Toolkit Quick Start Guide

Release v1.0.0.1 November 2016

Overview

Binary Markup Toolkit (BMTK) is a suite of software tools for working with Binary Markup Language (BML). BMTK includes tools to:

- Generate BML from “raw” image files, storage devices or file systems
- Convert BML to other convenient forms such as CSV and SQLite3 databases
- Display BML as an annotated hexadecimal dump

This **Quick Start Guide** is intended to give forensic practitioners the information necessary to install BMTK and perform basic BML operations. It includes a walkthrough describing how BMTK can be used to process an example “raw” FAT image file into a searchable SQLite3 database and then generate an annotated hexadecimal dump. Further examples are provided for a multi-partition image, NTFS file system and NTFS Master File Table ($MFT) file. This document assumes the reader is familiar with technology such as CSV and SQLite3.

What is Binary Markup Language?

BML is an XML-based language for the description of arbitrary binary data. It is designed for forensic computing, protocol debugging, reverse engineering and similar scenarios. Typically, a single BML file is associated with a single binary file, forensic “image” or protocol “dump”.

BML is human readable and can be authored by hand or generated automatically by software. It describes the location and size of fields within the underlying data. Optionally, it can also describe hierarchical data relationships, field names, interpreted data values/types and descriptions. BML provides an unambiguous *provenance*, or location and meaning, for fields within the data. A separate document provides a detailed description of BML syntax.

Is Binary Markup Language the same as *DFXML*?

DFXML is another XML based language for use in digital forensics. It was created by Simson Garfinkel and uses specific XML elements to describe supported file system metadata, file locations and Windows Registry values. With some exceptions DFXML does not describe the actual location of information such as metadata.

BML takes a different, lower-level, approach and is designed to record the detailed content of arbitrary binary data. BML is therefore typically more verbose than DFXML. In some cases BML and DFXML may be complementary.

Binary Markup Toolkit Technical Requirements

BMTK has the following technical requirements:

- Microsoft Windows XP/2003 or later (32-bit or 64-bit)
- 256MB of RAM (4GB or more recommended)
- Approximately 10MB of free storage for BMTK software and documentation
- Sufficient free storage for BML processing (application dependent) *

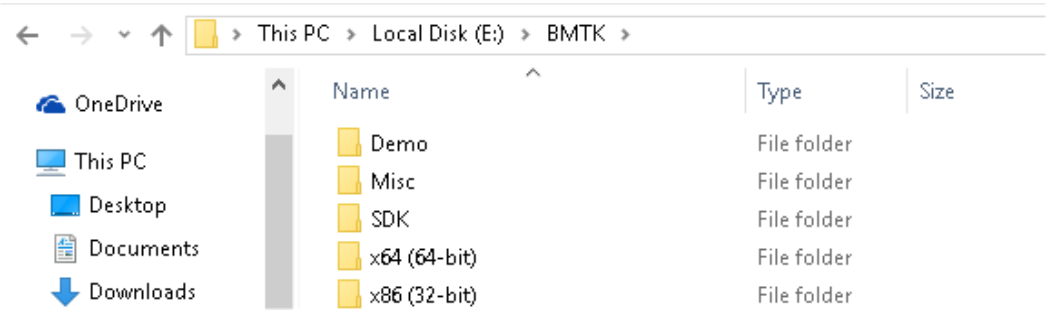
There is no functional difference between the 32-bit and 64-bit software and the 32-bit build may be used on most 64-bit systems. The 64-bit build may provide enhanced performance on 64-bit systems and is recommended where available. Please contact the author if you need to use BMTK on an operating system prior to Windows XP.

* BML can be verbose and a BML description of binary data can be several times the size of the original data. BML files >10GB are quite common and we suggest a typical system should have at least 100GB free storage for practical use with BMTK.

Binary Markup Toolkit Installation

BMTK requires a valid license file. This is called `"BMLicense.txt"` and is available on request from the author (support@datasysnergy.co.uk). To install the software proceed as follows:

1. Create an installation folder. For instance: `E:\BMTK`
2. Extract the BMTK distribution to this folder. The result should be similar to:



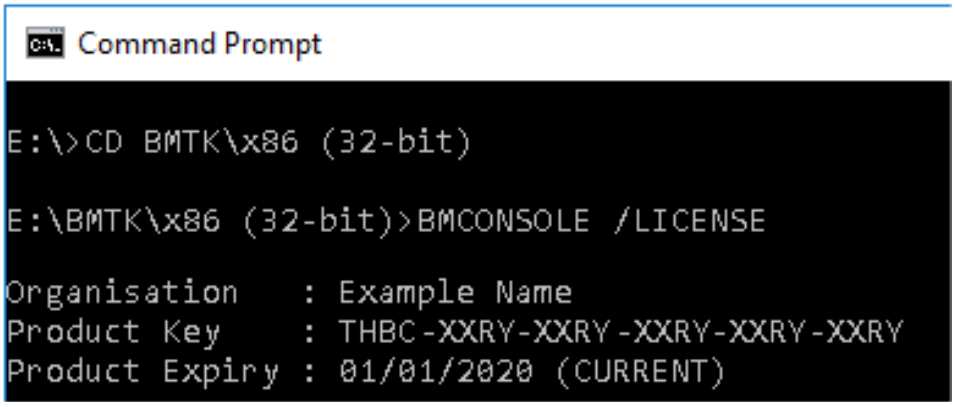
3. Copy the supplied `"BMLicense.txt"` file to the following locations:

```
E:\BMTK\x86 (32-bit)
E:\BMTK\x64 (64-bit)
```

   NB: This allows both the 32-bit and 64-bit software to be used. If you do not need one of these platforms you can ignore the associated folder in the following instructions. We recommend using the 64-bit version of BMTK where available.

4. Open the command prompt `(CMD.EXE)` and navigate to the installation folder

5. Use the `BMCONSOLE /LICENSE` command to confirm the license status:



Please remember that you are responsible for the safe keeping of your BMTK license file. The file is personal to you and should not be shared with anyone else.

Binary Markup Toolkit Components

BMTK currently provides the following tools:

Tool	Purpose
BMCONSOLE	Process a “raw” image file, storage device or file system into a BML document
BML2CSV	Convert a BML document to industry standard CSV format suitable for further processing using other tools
BML2DB	Convert a BML document to a SQLite3 database suitable for direct querying or further processing using other tools
BML2DUMP	Convert a BML database file (produced by BML2DB) to an annotated hexadecimal dump. This may be used to visualise the data descriptions provided by BML.

Binary Markup Toolkit Syntax

The syntax of each BMTK tool can be found by using the `/?` or `/HELP` commands. For example, to find the supported syntax and usage for the for `BMCONSOLE` program use the command: `BMCONSOLE /?`

```
E:\BMTK\x86 (32-bit)>BMCONSOLE /?
Syntax:

BMCONSOLE [options] [/flags:abc] sourcefile|sourcedevice

Where options may be:

  /START:off       - Specifies start offset in source file in bytes (default 0)
  /LENGTH:len      - Specifies data length to process (default is all)
  /OUTPUT:file.xml - Specifies output filename (default is 'BMOUT.XML')
  /LOG:logfile.txt - Specifies optional log filename
  /AGENT:name      - Specify agent name(s). May include wildcards * and ? (default is *)
  /LISTAGENTS      - List available agents. May be used with /AGENT filter
  /LICENSE         - Displays current license information
  /HELP or /?      - Displays syntax usage

Examples:

  BMCONSOLE /listagents
  BMCONSOLE /flags:fax /output:myfile.xml sourcefile.bin
  BMCONSOLE /output:file.xml \\.\PHYSICALDRIVE2
  BMCONSOLE /agent:fatfsagent /agent:ntfsagent \\.\Z:
```

The specific syntax of each program is different but there are some common optional arguments supported by several programs. Examples include:

Argument	Meaning
`/START`	Specifies the byte offset to start processing. This may a decimal number or a hexadecimal number starting with the `0x` prefix. For example: `/START:1024` or `/START:0x7E00`
`/LENGTH`	Specifies the number of bytes to process. This may be a decimal number or a hexadecimal number starting with the `0x` prefix. For example: `/LENGTH:4096`
`/OUTPUT`	Each BMTK tool provides a default output filename. To override this use the `/OUTPUT` argument. For example: `/OUTPUT:myfile.xml`
`/DB`	Specifies the name of a SQLite3 database file. Several BMTK tools support this format for faster processing of larger BML descriptions.
`/FLAGS`	Override default processing options to enable/disable specific features.
`/? /HELP`	Display the supported command line syntax.

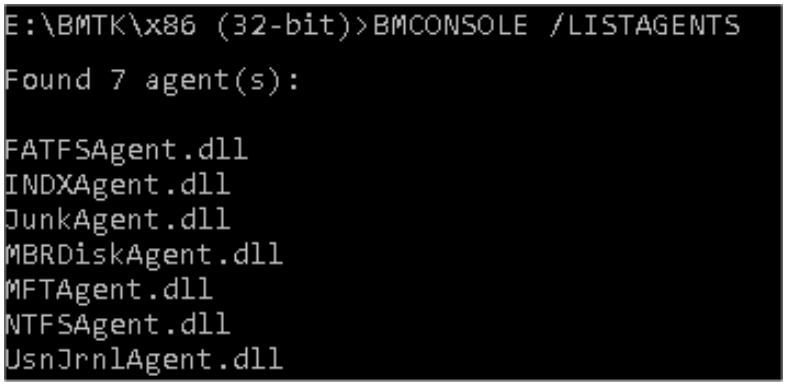
Binary Markup Toolkit Agents (plug-ins)

BMTK is extensible and uses agents (or “plug-ins”) to provide format specific processing. For example, an agent may target the JEPG file format or a complete file system such as FAT. BMTK agents are recursive and data identified by one agent (for instance in a file system) can be automatically routed to the most appropriate format specific agent. The following agents are currently available:

Agent	Purpose
JunkAgent	Demonstration agent for processing example JunkFS data. Used for testing and illustrating BMTK.
MBRDiskAgent	Partition table agent supporting MBR-style partitions (e.g. not GPT) and extended partitions. This agent allows complete disk images to be conveniently processed in a single session.
FATFSAgent	FAT file system agent supporting Microsoft implementation of FAT12/16/32 and long file name extensions.
NTFSAgent	NTFS file system agent supported Microsoft NTFS v1.2 (Windows NT v3.51) and later
MFTAgent	NTFS Master File Table agent for detailed processing of NTFS MFT records.
INDXAgent	NTFS Indx stream agent for detailed processing of non-resident NTFS indexes (e.g. typically directories)
UsnJrnlAgent	NTFS agent for detailed processing of filesystem USN Change Journal ($UsnJrnl:$J). The complete source code for this agent is in the SDK.
WinShellLinkAgent	Windows Shell Link (shortcut) agent

BMTK agents are implemented using standard Windows DLLs. The interface was primarily designed for the C/C++ languages but is compatible with any language than can produce native DLLs. Please contact the author (support@datasysnergy.co.uk) if you would like to write a BMTK agent.

You can use the `BMCONSOLE /LISTAGENTS` command to see the currently available agents:



All agents are enabled by default. To use only specific agent(s), use one or more `/AGENT` arguments. For example:

```
BMCONSOLE /AGENT:FATFSAgent /AGENT:OtherAgent myimage.img
```

Walkthrough: Using BMTK to process a “raw” FAT image file to BML

The BMTK is supplied with several “raw” format image files for illustration purposes. The examples are located in the `Demo` folder. The following sections explain how a small FAT formatted image file can be processed into a BML document and then converted to a variety of formats including an annotated hexadecimal dump for visualisation.

The supplied example file `freedosboot.img` is a FAT format boot disk image from the FreeDOS project. To process this image, and generate an equivalent BML description, proceed as follows:

1. Create a folder for BML output. For instance: `E:\BML`
2. Enter the following command:

```
BMCONSOLE /OUTPUT:e:\bml\freedosboot.xml
  e:\bmtk\demo\fat\freedosboot.img
```

3. BMTK will process the file. Depending on the input size this may take some time:

```
E:\BMTK\x86 (32-bit)>BMCONSOLE /OUTPUT:e:\bml\freedosboot.xml e:\bmtk\demo\fat\freedosboot.img

Input file:  e:\bmtk\demo\fat\freedosboot.img
Output file: e:\bml\freedosboot.xml
Processing started
Found 1 agent(s)
Loading: FATFSAgent.dll
FATFSAgent (c) Data Synergy UK Ltd 2016.
FAT filesystem  0%
Processing FAT table  100%
Processing directory  Root Directory 51%
Processing directory  DRIVER 100%
Processing directory  Root Directory 52%
Processing directory  FREEDOS 100%
Processing directory  Root Directory 100%
FAT filesystem  100%
Processing completed in 0 seconds
Unloaded 7 agent(s)
Processing finished. No errors reported.
```

4. Open the `e:\bml\freedosboot.xml` file using a text or XML editor (such as Notepad, Notepad++, Microsoft Visual Studio or XML Explorer). This contains the BML description of the data identified in the image file.

Walkthrough: Using BMTK to convert BML to CSV format (BML2CSV)

The BML document created in the previous section contains a description of the data identified in the original image file. This can be viewed directly with appropriate software (see above) or queried using XML capable tools. In some cases it may be more convenient to convert the BML document to another format.

CSV is an industry standard format for information exchange. A BML document that is converted to CSV format may be viewed as a spreadsheet (such as Microsoft Excel) or imported into a RDBMS system (such as SQLite, MySQL or MS SQL Server).

To convert the previous BML document to CSV format proceed as follows:

1. Enter the following command:

```
BML2CSV /OUTPUT:e:\bml\freedosboot.csv e:\bml\freedosboot.xml
```

2. BMTK will process the file. Depending on the input size this may take some time.

3. Open the resulting `freedosboot.csv` file using compatible software. For instance, Microsoft Excel:

	A	B	C	D	E	F	G	H	I
1	StartAbsO	EndAbsOf	Length	ParentOff	TypeID	Name	Text	Value	ValueType
2	0	1474559	1474560	0	1	freedosboot.img			
3	0	1474559	1474560	0	1	FATVolume			
4	0	511	512	0	2	FATBootRecord			
5	0	0	0	0	0		FAT12 Boot Record		
6	0	2	3	0	3	cJump	x86 Jump	eb3c90	Bin
7	3	10	8	3	3	cOEMIdentifier	OEM ID	4c494e55	Bin
8	11	12	2	11	3	wBytesPerSector	Bytes/Sector=0x200	0x200	UInt16
9	13	13	1	13	3	cSectorsPerCluster	Sectors/Cluster=0x1	0x1	Byte
10	14	15	2	14	3	wReservedSectors	Reserved Sectors=0x	0x1	UInt16
11	16	16	1	16	3	cNumberOfFats	Num FAT=2	0x2	Byte
12	17	18	2	17	3	wRootEntries	Root Entries=0xe0 (2	0xe0	UInt16
13	19	20	2	19	3	wNumberOfSectors	Sectors=0xb40 (288(	0xb40	UInt16

Walkthrough: Using BMTK to convert BML to SQLite3 format (BML2DB)

The previous section explained how a BML document could be converted to CSV format. BMTK includes a tool called `BML2DB` to directly convert BML to a SQLite3 database. This can be conveniently queried using SQL commands to identify and filter fields in the underlying data.

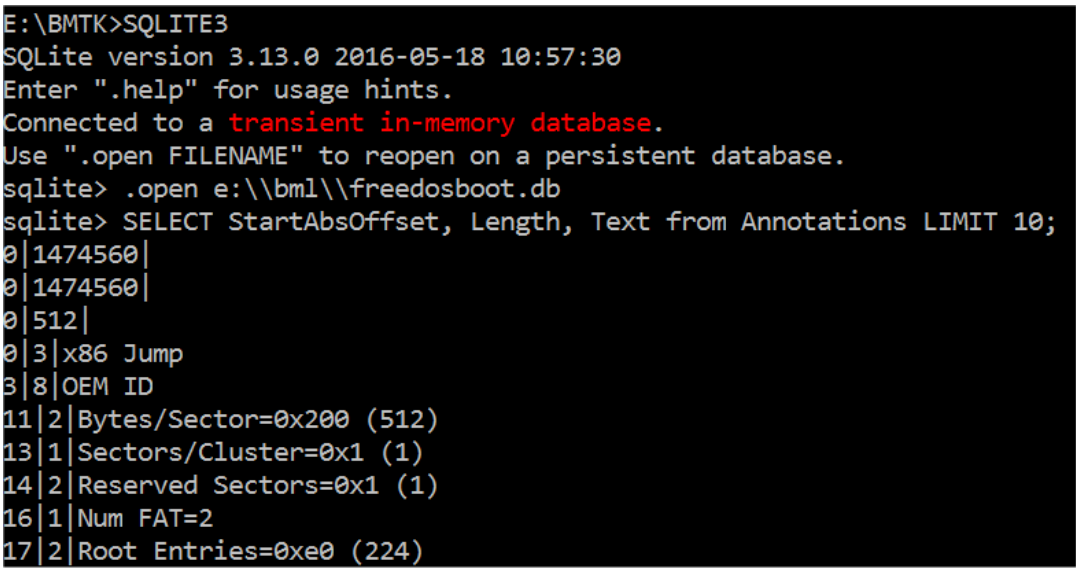
To convert the previous BML document to a SQLIte3 database proceed as follows:

1. Enter the following command:

   ```
   BML2DB /DB:e:\bml\freedosboot.db e:\bml\freedosboot.xml
   ```

2. BMTK will process the file. Depending on the data size this may take some time.

3. Open the resulting `freedosboot.db` file using `Sqlite3.exe` and query the desired data range. For example to find the first 10 records:



Tip: The SQLite tools can be downloaded from: https://www.sqlite.org/download.html

Walkthrough: Using BMTK to convert BML to an annotated hex dump

It may be useful to visualise the underlying data directly as an annotated hexadecimal dump. BMTK provides the `BML2DUMP` tool for this purpose.

To generate an annotated hexadecimal dump proceed as follows:

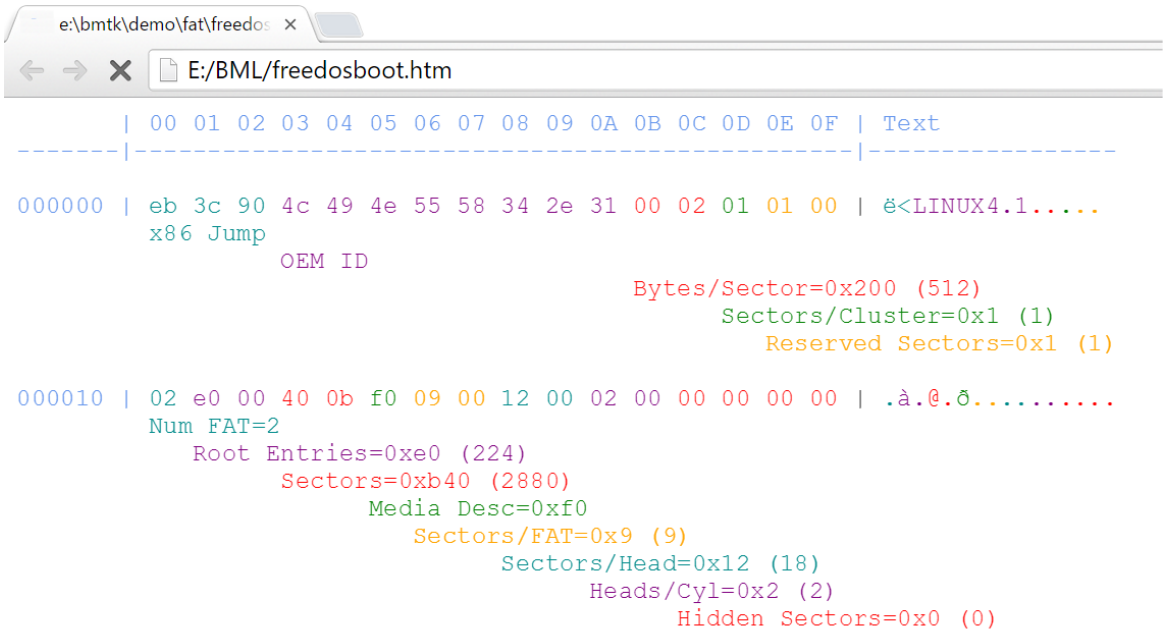
1. Convert the BML document to a SQLite3 database (see previous section)

2. Enter the following command:

```
BML2DUMP /OUTPUT:e:\bml\freedosboot.htm
  /DB:e:\bml\freedosboot.db e:\bmtk\demo\fat\freedosboot.img
```

3. BMTK will process the file. Depending on the data size this may take some time.

4. Open the `e:\bml\freedosboot.htm` using a web browser such as Google Chrome* to view the annotated hexadecimal dump:



*** Note:** Some browsers may have problems displaying very large HTML documents. See Tips section below for information about specific web browsers.

Walkthrough: Using BMTK to process a “raw” $MFT image file

The previous sections described the procedure to process a small FAT formatted image file to BML and subsequent convert it to CSV, SQLite3 and finally an annotated hexadecimal dump. The following section briefly describes how the same process can be applied to a standalone NTFS Master File Table ($MFT) file.

To process a $MFT file to an annotated hexadecimal dump proceed as follows:

1. Generate a BML description file from the $MFT file dump using the command:

```
BMCONSOLE /OUTPUT:e:\bml\mft.xml
  e:\bmtk\demo\ntfs\mft.img
```

2. Convert the BML file to a SQLite3 database with the command:

```
BML2DB /DB:e:\bml\mft.db e:\bml\mft.xml
```

3. Generate the annotated dump with the command:

```
BML2DUMP /OUTPUT:e:\bml\mft.htm /DB:e:\bml\mft.db
  e:\bmtk\demo\ntfs\mft.img
```

4. Finally, open the `e:\bml\mft.htm` using a web browser such as Google Chrome.

5. The example contains a resident file at offset 0x8400. The resident data itself can be found at offset 0x8510:

```
008510 | 80 00 00 00 48 00 00 00 00 00 18 00 00 00 01 00 | ...H...........
         Attribute Type# 0x80 $DATA Resident Unnamed
         TypeCode=0x80
                     RecordLength=0x48
                                 FormCode=0x00 (Resident)
                                    NameLength=0x0
                                       NameOffset=0x18
                                             Flags=0x0 (NONE)
                                                   Instance=0x1

008520 | 2c 00 00 00 18 00 00 00 54 68 69 73 20 69 73 20 | ,.......This is
         Resident ValueLength=0x2c
                     Resident ValueOffset=0x18
                              Reserved

008530 | 61 6e 20 65 78 61 6d 70 6c 65 20 6f 66 20 61 20 | an example of a
008540 | 73 6d 61 6c 6c 20 72 65 73 69 64 65 6e 74 20 66 | small resident f

008550 | 69 6c 65 2e 00 00 00 00 ff ff ff ff 82 79 47 11 | ile.....ÿÿÿÿyG.
                                 $END
```

Walkthrough: Using BMTK to process a complete NTFS file system

The previous section described the procedure to process a standalone NTFS $MFT file. The following section briefly describes how the same procedure can be used to process a NTFS file system image.

To process a complete NTFS file system to an annotated hexadecimal dump proceed as follows:

1. Generate a BML description file of the NTFS file system using the following command:

   ```
   BMCONSOLE /OUTPUT:e:\bml\ntfs.xml e:\bmtk\demo\ntfs\ntfs.img
   ```

2. Convert the BML file to a SQLite3 database with the command:

   ```
   BML2DB /DB:e:\bml\ntfs.db e:\bml\ntfs.xml
   ```

3. Generate the annotated dump with the command:

   ```
   BML2DUMP /OUTPUT:e:\bml\ntfs.htm /DB:e:\bml\ntfs.db
     e:\bmtk\demo\ntfs\ntfs.img
   ```

4. Finally, open the `e:\bml\ntfs.htm` using a web browser such as Google Chrome:

Walkthrough: Using BMTK to process a disk image (multiple partitions)

The previous section described the procedure to process a NTFS file system image. The following section briefly describes how the same procedure can be used to process a complete “raw” disk image containing multiple (MBR style) partitions.

To process a disk image to an annotated hexadecimal dump proceed as follows:

1. Generate a BML description file of the NTFS file system using the following command:

   ```
   BMCONSOLE /OUTPUT:e:\bml\3partitions.xml
     e:\bmtk\demo\mbr\3partitions.img
   ```

2. Convert the BML file to a SQLite3 database with the command:

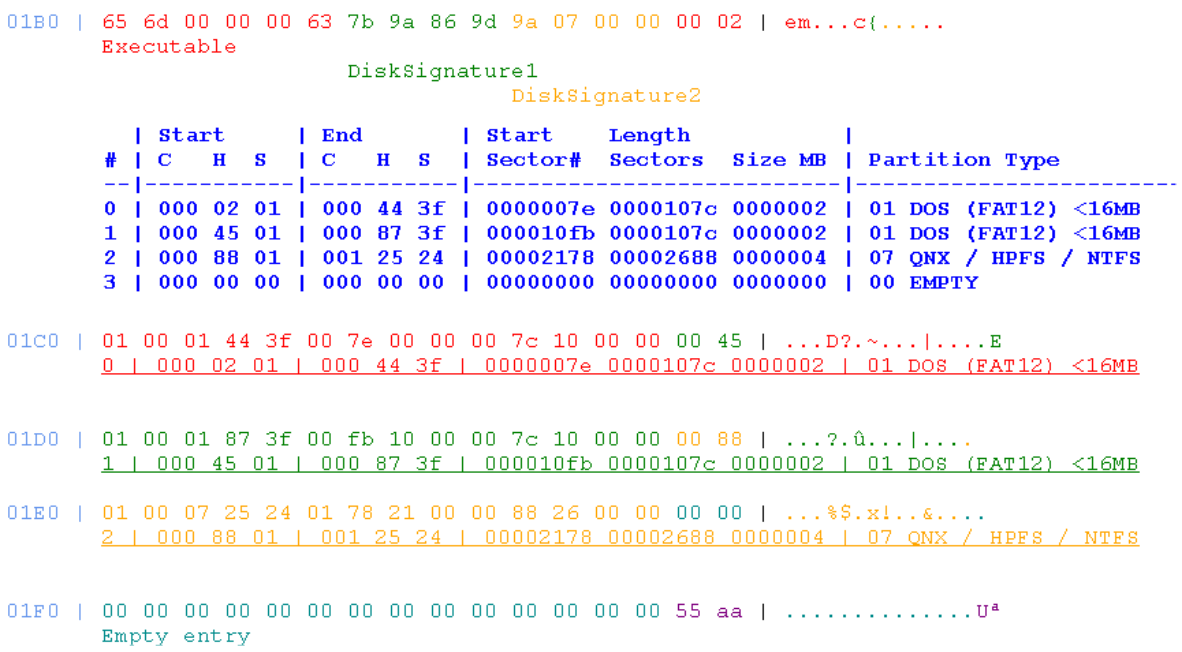
   ```
   BML2DB /DB:e:\bml\3partitions.db e:\bml\3partitions.xml
   ```

3. Generate the annotated dump for the first 512 bytes* with the command:

   ```
   BML2DUMP /START:0 /LENGTH:512 /OUTPUT:e:\bml\3partitions.htm
      /DB:e:\bml\3partitions.db e:\bmtk\demo\mbr\3partitions.img
   ```

   *The demo "`3partitions.img`" file is only 9MB in size and can be displayed in Google Chrome. However, a practical disk image will typically be much larger and not suitable for viewing as a single file. See the Tips section below for further information.

4. Open the `e:\bml\3partitions.htm` using a web browser such as Google Chrome and view the Master Boot Record (MBR):



Walkthrough: Processing $UsnJrnl to BML using USN Change Journal agent

This walkthrough describes how to use the UsnJrnl agent to generate a BML description of a NTFS USN Change Journal and optionally visualise it as a coloured, annotated, hexadecimal dump. The next section describes how to parse the BML using C#/XPath and build a basic forensic timeline. An example $UsnJrnl:$J file is provided in the `Demo\USNJrnl` folder of the BMTK distribution.

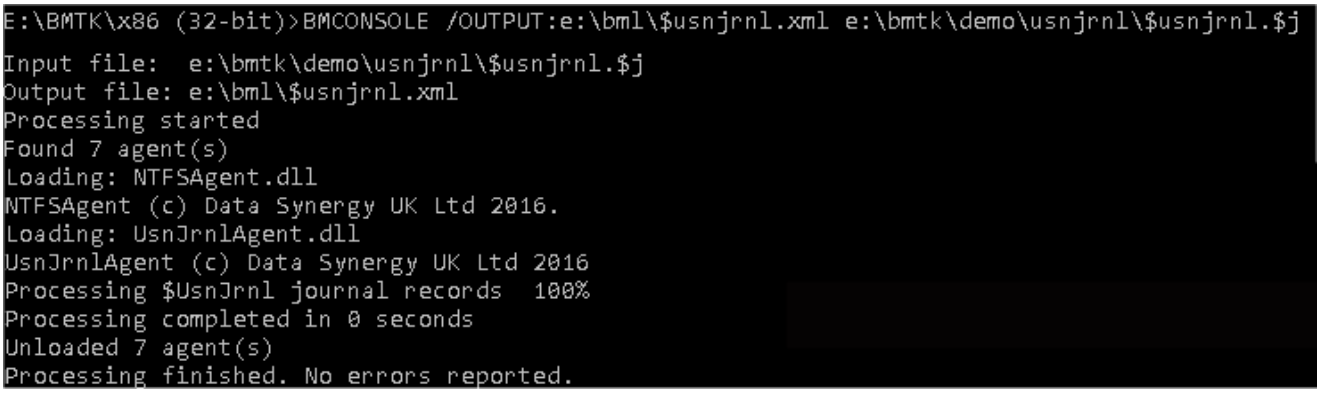
To process a USN Change Journal into BML proceed as follows:

1. Obtain a USN Change Journal ($UsnJrnl:$J) file. For instance, this can be extracted using the X-Ways WinHex tool using this procedure:

   a. Select **Tools/Open Disk**
   b. Select the desired NTFS volume and click **OK**
   c. Navigate to the `$Extend\$UsnJrnl` file
   d. Select the `$J` stream
   e. Right-click and select **Recovery/Copy**
   f. Select a target folder and click **OK**
   g. Tip: The extract file may initially be hidden
   h. For convenience, rename the extracted file to: `$UsnJrnl.$J`

2. Generate a BML description of the $UsnJrnl.$J file using the command:

```
BMCONSOLE /OUTPUT:e:\bml\$usnjrnl.xml
  e:\bmtk\demo\usnjrnl\$usnjrnl.$j
```



3. View the BML description using appropriate XML software.

> **Tip:** "XML Explorer" is an excellent free XML viewer that supports large document sizes and optional XML syntax highlighting. It can be downloaded from: https://xmlexplorer.codeplex.com

4. To convert the BML file to an annotated hexadecimal dump use the following two commands:

```
BML2DB /DB:e:\bml\$usnjrnl.db e:\bml\$usnjrnl.xml

BML2DUMP /OUTPUT:e:\bml\$usnjrnl.htm /DB:e:\bml\$usnjrnl.db
     e:\bmtk\demo\usnjrnl\$usnjrnl.$j
```

5. Finally, open the annotated hexadecimal dump using a web browser such as Google Chrome. The screenshot on the following page shows the annotated dump for the previous USN Change Journal record.

Note: This journal record is located at offset 0x3668 in the change journal. The Usn field also reports 0x3668. This is expected because the Usn field is simply an offset into the journal file. The record is for a file called "`MrStrong.jpg`" that was located in $MFT reference 0x32 and within the parent directory with $MFT reference 0x27. The `Demo\USNJrnl` folder contains a copy of the associated $MFT file.


```
3650 | 53 00 74 00 72 00 6f 00 6e 00 67 00 2e 00 6a 00 | S.t.r.o.n.g...j.

3660 | 70 00 67 00 00 00 00 00 58 00 00 00 02 00 00 00 | p.g.....X.......
       Filename='MrStrong.jpg'
                                 USN_RECORD_V2 File: 'MrStrong.jpg'
                                 RecordLength=0x58
                                              MajorVersion=0x2
                                                    MinorVersion=0x0

3670 | 32 00 00 00 00 00 02 00 27 00 00 00 00 00 01 00 | 2.......'.......
       FileReferenceNumber=0x2000000000032
                                 ParentFileReferenceNumber=0x1000000000027

3680 | 68 36 00 00 00 00 00 00 12 fd c5 d4 83 41 d2 01 | h6.......ýÅÔÃAÒ.
       USN=0x3668
                                 TimeStamp=20161118 100921343

3690 | 20 00 20 00 00 00 00 00 00 00 00 00 20 00 00 00 | . ......... ...
       Reason=0x200020
       Named $data attribute extended,
       Named $data added, removed or renamed
                    SourceInfo=0x0
                                 SecurityId=0x0
                                              FileAttributes=0x20
                                              ARCHIVE

36A0 | 18 00 3c 00 4d 00 72 00 53 00 74 00 72 00 6f 00 | ..<.M.r.S.t.r.o.
       FileNameLength=0x18
             FileNameOffset=0x3c

36B0 | 6e 00 67 00 2e 00 6a 00 70 00 67 00 00 00 00 00 | n.g...j.p.g.....
       Filename='MrStrong.jpg'
```

Walkthrough: Using USN Change Journal BML to create a basic forensic timeline with C# and XPath

The previous walkthrough described how to use the UsnJrnl agent to process the NTFS USN Change Journal into an equivalent BML description. This section describes how to quickly convert the BML description to a basic forensic timeline using *XPath* and a small BML parser written in C#. The walkthrough is intend to illustrate the basic method and does not attempt to further analyse the timeline. The suggested procedure is:

1. Confirm the Microsoft .NET v4 framework is installed. This is present on most Windows systems. At the time of writing the current (v4.6.2) release can be downloaded here: https://www.microsoft.com/en-us/download/details.aspx?id=53344

   Note: The installation procedure varies in each Windows release. Please consult the Microsoft documentation for detailed instructions on how to install the.NET framework.

2. Copy the C# source code shown at the end of this walkthrough into a text file (a copy of the code is available in the Examples\UsnJrnlBML2Time folder of the BMTK SDK) and save the file as UsnJrnlBml2Time.cs

3. Open a command prompt and navigate to the folder containing the C# source file. For instance:

   ```
   E:

   CD e:\bmtk\sdk\examples\usnjrnlbml2time
   ```

4. Use the .NET C# compiler to build the UsnJrnlBml2Time.cs file into a standalone executable. For instance, to use the standard 32-bit compiler:

   ```
   c:\windows\microsoft.net\framework\v4.0.30319\CSC.EXE
      /out:UsnJrnlBml2Time.exe UsnJrnlBml2Time.cs
   ```

5. Use the resulting UsnJrnlBml2Time.exe program to convert the BML document to a CSV file. For instance:

   ```
   USNJRNLBML2TIME e:\bml\$usnjrnl.xml e:\bml\$usnjrnl.csv
   ```

6. Open the resulting CSV file using Microsoft Excel (or similar) spreadsheet software

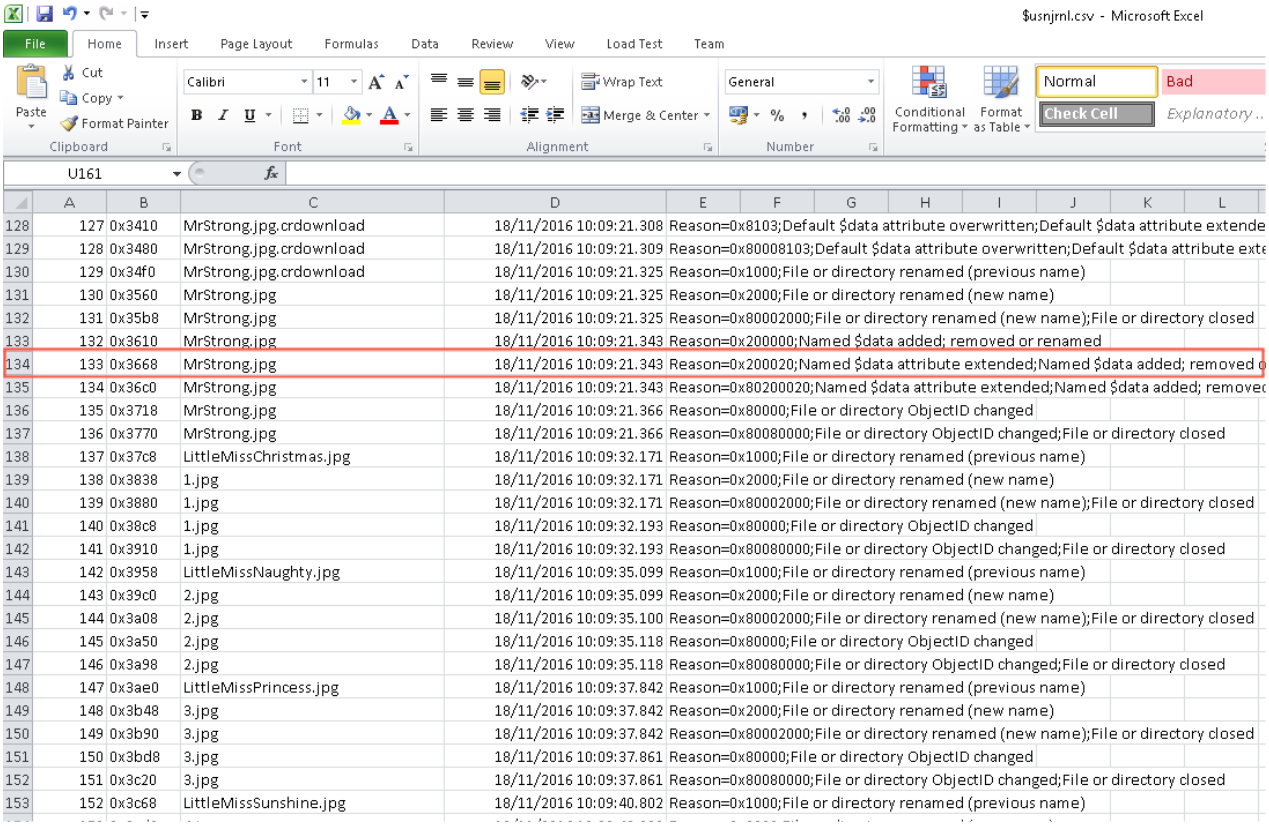
7. Select 4th column (labelled “Timestamp”), right-click, and select **Format Cells**.

8. Navigate to **Custom** and enter the custom date format specifier:

   ```
   dd/mm/yyyy hh:mm:ss.000
   ```

NB: This instructs Excel to display the timestamp including to millisecond precision.

9. Arrange the spreadsheet columns for convenient viewing and scroll down to view the USN Change Journal entries arranged in USN (chronological) order. This could be used as the basis for a forensic activity timeline. The screenshot below shows the USN journal entry from the previous walkthrough (located at USN = 0x3668):



$usnjrnl.csv - Microsoft Excel

File Home Insert Page Layout Formulas Data Review View Load Test Team

Cut Copy Format Painter Paste Clipboard | Calibri 11 Font | Wrap Text Merge & Center Alignment | General Number | Conditional Formatting Format as Table | Normal Bad Check Cell Explanatory ...

U161

	A	B	C	D	E
128	127	0x3410	MrStrong.jpg.crdownload	18/11/2016 10:09:21.308	Reason=0x8103;Default $data attribute overwritten;Default $data attribute extende
129	128	0x3480	MrStrong.jpg.crdownload	18/11/2016 10:09:21.309	Reason=0x80008103;Default $data attribute overwritten;Default $data attribute exte
130	129	0x34f0	MrStrong.jpg.crdownload	18/11/2016 10:09:21.325	Reason=0x1000;File or directory renamed (previous name)
131	130	0x3560	MrStrong.jpg	18/11/2016 10:09:21.325	Reason=0x2000;File or directory renamed (new name)
132	131	0x35b8	MrStrong.jpg	18/11/2016 10:09:21.325	Reason=0x80002000;File or directory renamed (new name);File or directory closed
133	132	0x3610	MrStrong.jpg	18/11/2016 10:09:21.343	Reason=0x200000;Named $data added; removed or renamed
134	133	0x3668	MrStrong.jpg	18/11/2016 10:09:21.343	Reason=0x200020;Named $data attribute extended;Named $data added; removed
135	134	0x36c0	MrStrong.jpg	18/11/2016 10:09:21.343	Reason=0x80200020;Named $data attribute extended;Named $data added; removed
136	135	0x3718	MrStrong.jpg	18/11/2016 10:09:21.366	Reason=0x80000;File or directory ObjectID changed
137	136	0x3770	MrStrong.jpg	18/11/2016 10:09:21.366	Reason=0x80080000;File or directory ObjectID changed;File or directory closed
138	137	0x37c8	LittleMissChristmas.jpg	18/11/2016 10:09:32.171	Reason=0x1000;File or directory renamed (previous name)
139	138	0x3838	1.jpg	18/11/2016 10:09:32.171	Reason=0x2000;File or directory renamed (new name)
140	139	0x3880	1.jpg	18/11/2016 10:09:32.171	Reason=0x80002000;File or directory renamed (new name);File or directory closed
141	140	0x38c8	1.jpg	18/11/2016 10:09:32.193	Reason=0x80000;File or directory ObjectID changed
142	141	0x3910	1.jpg	18/11/2016 10:09:32.193	Reason=0x80080000;File or directory ObjectID changed;File or directory closed
143	142	0x3958	LittleMissNaughty.jpg	18/11/2016 10:09:35.099	Reason=0x1000;File or directory renamed (previous name)
144	143	0x39c0	2.jpg	18/11/2016 10:09:35.099	Reason=0x2000;File or directory renamed (new name)
145	144	0x3a08	2.jpg	18/11/2016 10:09:35.100	Reason=0x80002000;File or directory renamed (new name);File or directory closed
146	145	0x3a50	2.jpg	18/11/2016 10:09:35.118	Reason=0x80000;File or directory ObjectID changed
147	146	0x3a98	2.jpg	18/11/2016 10:09:35.118	Reason=0x80080000;File or directory ObjectID changed;File or directory closed
148	147	0x3ae0	LittleMissPrincess.jpg	18/11/2016 10:09:37.842	Reason=0x1000;File or directory renamed (previous name)
149	148	0x3b48	3.jpg	18/11/2016 10:09:37.842	Reason=0x2000;File or directory renamed (new name)
150	149	0x3b90	3.jpg	18/11/2016 10:09:37.842	Reason=0x80002000;File or directory renamed (new name);File or directory closed
151	150	0x3bd8	3.jpg	18/11/2016 10:09:37.861	Reason=0x80000;File or directory ObjectID changed
152	151	0x3c20	3.jpg	18/11/2016 10:09:37.861	Reason=0x80080000;File or directory ObjectID changed;File or directory closed
153	152	0x3c68	LittleMissSunshine.jpg	18/11/2016 10:09:40.802	Reason=0x1000;File or directory renamed (previous name)

```
//Example C# program to process UsnJrnl BML description to CSV file
//
//Author: James Clark
//
//Note: This simple demonstration program has no error checking

using System;
using System.Xml;
using System.IO;

namespace DataSynergy
{
    class UsnJrnlBML2Time
    {
        //Syntax:   UsnJrnlBml2Time inputfile.xml outputfile.csv
        //Example:  UsnJrnlBml2Time e:\bml\$usnjrnl.xml e:\bml\usnjrnltimes.csv"
        static void Main(string[] args)
        {
            //Open output CSV file
            using (var stream = File.CreateText(args[1]))
            {
                //Write CSV header row
                stream.WriteLine("Index,USN,Filename,Timestamp,ReasonText");

                //Load input XML document
                XmlDocument xmlDoc = new XmlDocument();
                xmlDoc.Load(args[0]);

                //Process all nodes with Name="USN_RECORD_V2"
                XmlNodeList xnList = xmlDoc.SelectNodes("/BMLRoot/Region/Structure[@Name='USN_RECORD_V2']");
                ulong lineCount = 0;
                foreach (XmlNode xn in xnList)
                {
                    //Find USN, FileName, TimeStamp, and Reason nodes
                    XmlNode usnNode = xn.SelectSingleNode("*[@Name='USN']");
                    XmlNode fileNameNode = xn.SelectSingleNode("*[@Name='FileName']");
                    XmlNode timeStampNode = xn.SelectSingleNode("*[@Name='TimeStamp']");
                    XmlNode reasonNode = xn.SelectSingleNode("*[@Name='Reason']");

                    //Increment line counter and write CSV line
                    stream.WriteLine("{0},{1},{2},{3},{4}", ++lineCount,
                                                            usnNode.Attributes["Value"].Value,
                                                            fileNameNode.Attributes["Value"].Value,
                                                            timeStampNode.Attributes["Value"].Value,
                                                            SanitizeReasonText(reasonNode.InnerText));
                }
            }
        }

        //Ensure reason text is valid CSV and remove <br/> HTML tag
        static string SanitizeReasonText(string reasonText)
        {
            reasonText = reasonText.Replace(",<br/>", ";");  //Convert ,<br/> to ;
            reasonText = reasonText.Replace("<br/>", ";");   //Convert <br/> to ;
            reasonText = reasonText.Replace(',', ';');       //Convert , to ;
            return reasonText;
        }
    }
}
```

Source code for "UsnJrnlBml2Time" program. Note: this code has no error checking.

Available in: SDK\Examples\UsnJrnlBML2Time folder of the BMTK SDK

Binary Markup Toolkit Tips

The following section provides some tips for working effectively with BMTK:

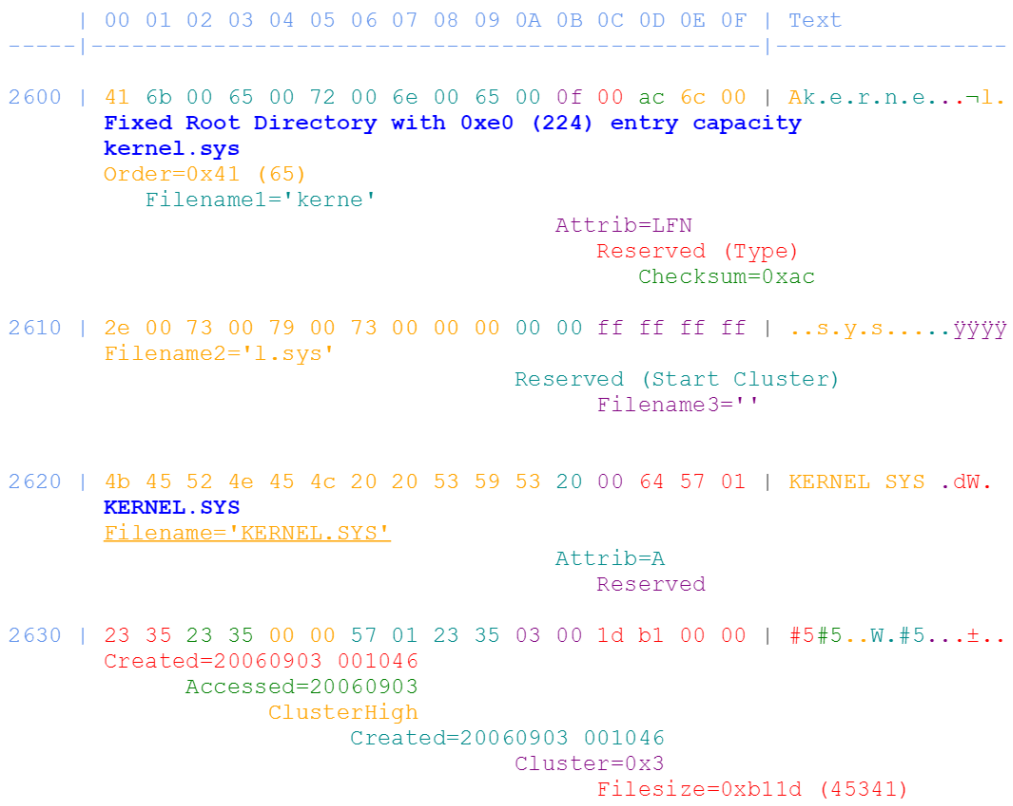
1. BML documents can be very large. Some editors (for example Windows Notepad) load the entire document into memory. This is impractical for very large documents. “XML Explorer” is an excellent free XML viewer that supports large document sizes and optional XML syntax highlighting. It can be downloaded from: https://xmlexplorer.codeplex.com/

2. Annotated hexadecimal dumps can be very large. Practical experience has found that the Microsoft Internet Explorer and Edge browsers do not perform well with large HTML files. Google Chrome does not have this limitation and will work reliably with much larger documents.

3. The `/START` and `/LENGTH` arguments can be used to produce an annotated dump of a selected data region. This can be used to overcome the problems with very large HTML files. For instance, to view the root directory in the previous example use the command:

   ```
   BML2DUMP /START:0x2600 /LENGTH:0x1C00
    /OUTPUT:e:\bml\freedosboot_rootdir.htm
    /DB:e:\bml\freedosboot.db e:\bmtk\demo\fat\freedosboot.img
   ```

   For example:

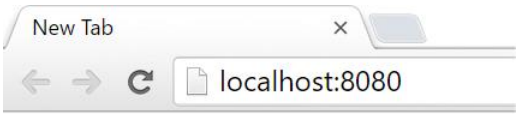


4. Very large annotated hexadecimal dumps (typically > 50MB) may be impractical to view as a single document (even in Google Chrome). The `BML2DUMP` tool includes an experimental web server feature that can serve requested portions from a much larger document. This can be an effective way to navigate very large annotated dumps. To use the web server feature on port 8080 proceed as follows:

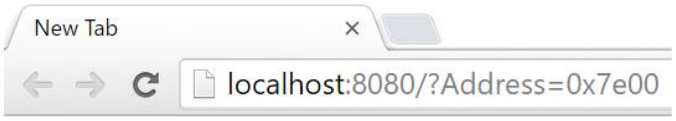
   a. Start the web server with the command:

   ```
   BML2DUMP /SERVER:8080
     /DB:e:\bml\freedosboot.db e:\bmtk\demo\fat\freedosboot.img
   ```

   b. Open the web browser (for instance Google Chrome) and enter the server URL `http://localhost:8080`. For example:

   c. Browse the annotated dump as desired. You can amend the request to specify a particular data location using the `Address` parameter. For example:

   d. Similarly, you can request a particular range of data using the `Length` parameter:

   Note: The `Address` and `Length` parameters support both decimal and hexadecimal notation. To specify a hexadecimal value use the `0x` prefix.

5. `BML2DB` permits multiple BML documents to be imported in to the same SQLite3 database. The first session is index 1, the second index 2 and so on. `BML2DUMP` provides the `/SESSION` argument to access a specific session. For instance:

   ```
   BML2DUMP /OUTPUT:myoutput.htm /SESSION:3
     /DB:mydatabase.db myimage.img
   ```

6. Both `BMCONSOLE` and `BML2DUMP` support direct operations with *live* data sources such as storage devices and file systems. To work with these sources proceed as follows:

   a. Open an administrator (e.g. elevated on Windows Vista and later) command prompt

b. Execute the desired BMTK command

c. For example, to process the Z: file system use the command:

```
BMCONSOLE \\.\Z:
```

d. Similarly, to process the hard disk #2 (e.g. the third) use the command:

```
BMCONSOLE \\.\PhysicalDrive2
```

e. Finally, `BML2DUMP` can be used to directly generate a hexadecimal dump from a live file system. For instance:

```
BML2DUMP /FORMAT:text /START:0 /LENGTH:512 \\.\e:
```

7. Direct operation with live data sources (see previous tip) may be useful for training or demonstration purposes but is not recommend for practical operation because the data source may be volatile and not provide a forensically sound data source. A basic tool to generate “raw” image files is provided in the `Misc` folder of the BMTK distribution.

   To generate an image of the Z: file system, open an administrator command prompt and use the following command:

```
DISKDUMP /num:0 /export:raw.img \\.\z:
```

Feedback

This is the first public release of BMTK. The software and the BML language are still under development and therefore subject to change. The author would welcome questions, feedback and suggestions about both BMTK and BML.

BMTK and BML were created by James Clark. The project is sponsored by Data Synergy UK Ltd. To contact the author please email support@datasysnergy.co.uk