Representing File and File System information using CASE/UCO

This document describes the design decisions for representing information that is contained within a file or file system, and associated metadata.

Decision 1: Designate the Trace object with a general File property bundle to specify details of a specific file itself. Enrich the Trace with additional proper bundles that are specific to the file system (e.g., ExtInode, MftRecord), or that are specific to the class of file (e.g., Image, RasterPicture, SQLiteBlob,WindowsPEBinaryFile).

Decision 2: Use a Relationship object to represent where a File is located within another trace such as a Volume, Partition or another File. Within this Relationship object, to specify where the File is located within another object such as a Volume, Partition or another File, either use the PathRelation property bundle (logical location) or the DataRange property bundle (physical, bitwise location).

Decision 3: By default, use a ContentData property bundle on the "File" trace to specify details (e.g., hashes, size, magicNumber) of the actual content (the bits) of the file. Optionally, if the actual content of the file is relevant beyond its single instantiation in the File (e.g., content as it exists in memory) then the content may be captured using a ContentData property bundle on a separate Trace object and related to the "File" trace with a "has-content" Relationship object.

Overview Diagram

The following diagram provides a general overview of how a File is represented using CASE, and how it can be linked to an associated File System.

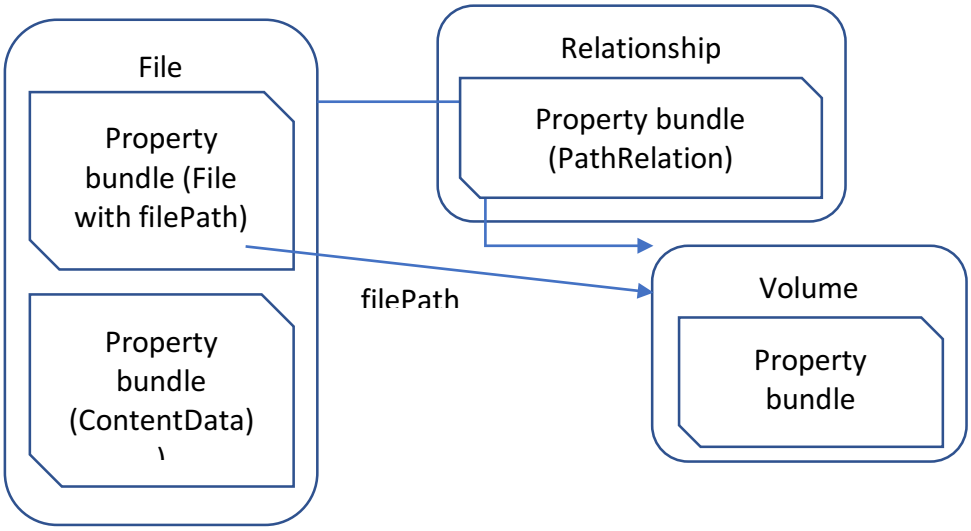

Figure: Depiction of a file trace (e.g., in an NTFS file system) represented using the File property bundle and a file system specific property bundles with a Relationship mapping the location of the file within a file system.

Descriptive JSON-LD Examples

The simplest case of representing a single file and its content.


```
{
   "@id": "file-38e5cd74-19b2-3f0c-b324-1c4b25a34f12",
   "@type": "Trace",
   "propertyBundle": [
    {
     "@type": "File",
     "createdTime": "2017-09-15T10:12:19.32Z",
     "extension": "dd",
     "fileName": "IMG-425634.JPG",
     "fileSystemType": "NTFS",
     "filePath": "C:/SecretStash/IMG-425634.JPG",
     "isDirectory": false,
     "allocationStatus": "unallocated",
     "sizeInBytes": 4138616
    },
    {
     "@type": "ContentData",
     "hash": [
      {
       "@type": "Hash",
       "hashMethod": "SHA256",
       "hashValue":
"ed1b9496953a9e9d2e797fb68fee7150cfb9e6d3ff97c0f64a35068264672918"
      }
     ],
     "sizeInBytes": 4138616
    }
   ]
 },
```


In addition to representing files with associated file system properties such as filePath, it is possible to represent properties of file system itself such as a Partition with a Volume formatted as NTFS, using a FileSystem property bundle (the only two properties currently used for this are fileSystemType and clusterSize).


```
{
 "@id": "partition-27d6ac58-24c1-2a0b-c314-4b4b35a35f25",
 "@type": "Trace",
 "propertyBundle": [
  {
     "@type": "DiskPartition",
     "diskPartitionType": "MSDOS"
     "partitionID": "06"
     "partitionOffset": "63"
```

```
      "partitionLength": "245235063"
    },
        {
      "@type": "FileSystem",
      "diskPartitionType": "NTFS"
    },
    {
      "@type": "ContentData",
      "sizeInBytes": 245235000,
      "hash": [
       {
        "@type": "Hash",
        "hashMethod": "SHA256",
        "hashValue":
"0611ea093d19b1c73a5285ff43741dd77f2a8d983c1c71044eb072e44f5dcb0a"
       }
      ]
     }
   ]
 }
```

A Relationship object is used to represent where a file is located within another object.

```
{
  "@id": "trace-relationship-uuid",
  "@type": "Relationship",
  "source": "file-38e5cd74-19b2-3f0c-b324-1c4b25a34f12",
  "target": "partition-27d6ac58-24c1-2a0b-c314-4b4b35a35f25",
  "kindOfRelationship": "contained-within",
  "isDirectional": true,
  "propertyBundle": [
    {
     "@type": "PathRelation",
     "path": " C:/SecretStash/IMG-425634.JPG"
    }
  ]
},
```

The typical use case for digital forensic purposes is to use a single trace to represent both the file properties and file system properties. Alternatively, if you want to convey a single file existing in multiple filesystems without duplicating intrinsic file information across multiple traces you can do so by specifying a trace for the intrinsic properties (ContentData), specifying a trace for each filesystem instantiation of that file, and specifying “has-content” Relationships between each filesystem instantiation and the intrinsic file properties trace.