User documentation

"Data structures of OCFA"

The Open Computer Forensics Architecture
Data structures explained

March 2011

KLPD, Driebergen

Author: J. van der Wal

User documentation

“Data structures of OCFA”

User documentation

“Data structures of OCFA”

Table of contents

1 Introduction

This document should help developers to develop interfaces on the internal data structures of the Open Computer Forensics Architecture (OCFA).

1.1 Remarks

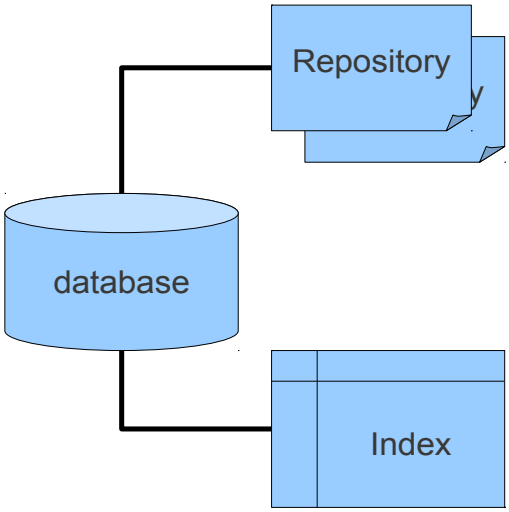
Not all aspects of ocfa are described yet. Please feel free to contact the author for suggestions.

This document assumes the usage of ***ocfa version 2.2.x*** or higher.

2 Results of a wash process

OCFA was build to recursively extract content and meta information from a seized computer image. The extracted content is stored in three places:

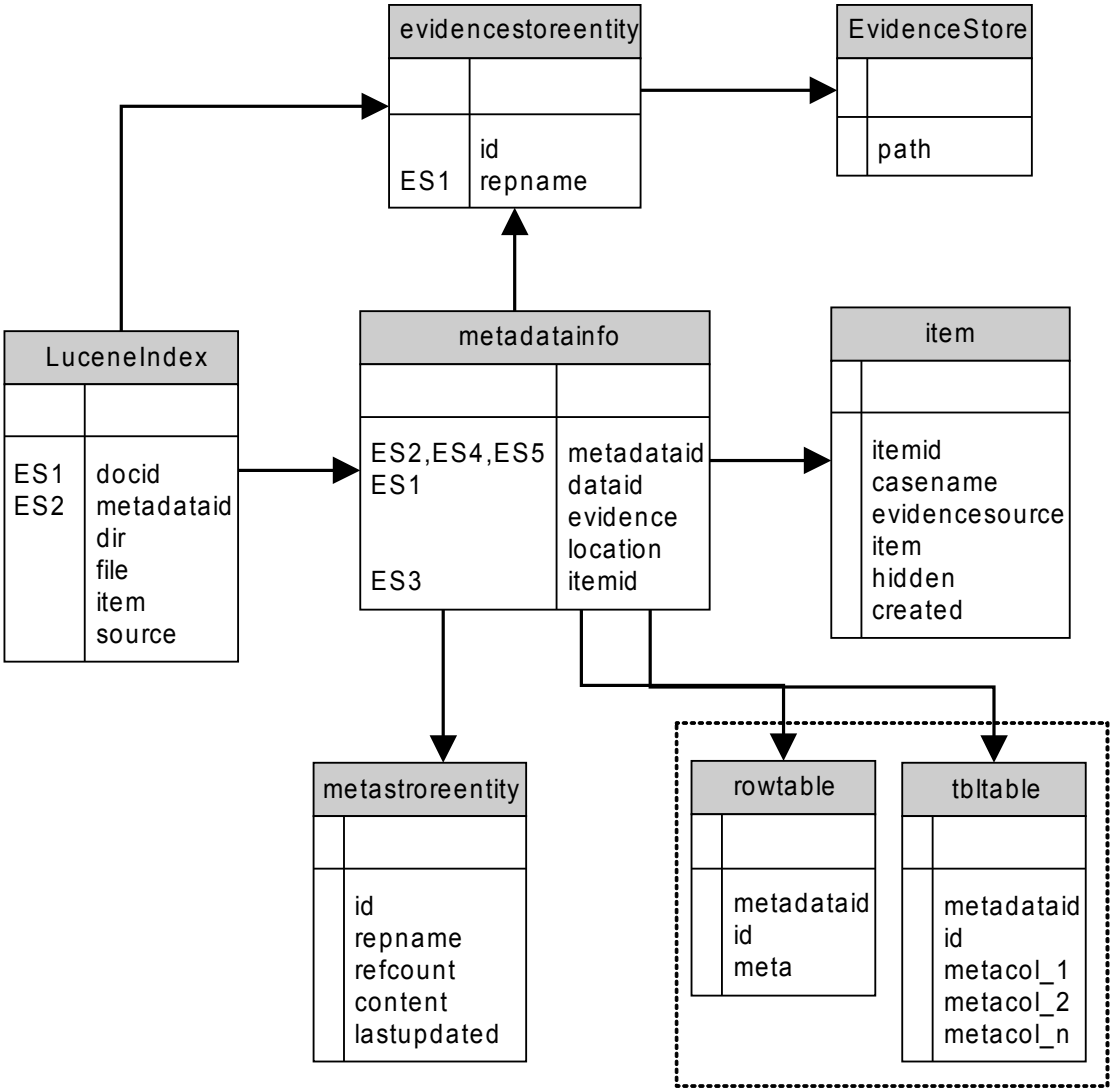
1. database (postgresql)
2. repository (file-tree)
3. full text index (Apache-Lucene)



The database is a linking pin between the full text index and the repository.

3 The OCFA database

3.1 Tables



The table diagram above is a simplified view of the ocfa postgres database. The numerous different row-tables containing all different types of metadata are shown as a single rowtable table.

Furthermore, the database model is extended with the Lucene index and the file-tree base repository as if they were tables in the database, because they can be linked to columns in the database. These links are also shown. The “ES” notation in the database diagram suggest the presence of foreign keys. Nevertheless, in this diagram, they have no meaning.

3.2 Table Content Explained

3.2.1 Metadatainfo table

The metadatainfo table basically is a linking pin between most of the tables in the ocfa database. Two columns need extra explanation:

evidence character varying(1024)
0
0
0.j0e0
0.j0e0
0.j0e0.j0e0
0.j0e0.j0e0
0.j0e0.j0e0.j0e0
0.j0e0.j0e0.j0e0
0.j0e0.j0e0.j0e0.j0e0
0.j0e0.j0e0.j0e0.j0e0
0.j0e0.j0e0.j0e0.j0e1
0.j0e0.j0e0.j0e0.j0e2
0.j0e0.j0e0.j0e1

- evidence
  Holds information about evidence parent-child relations. The dot in the evidence string separates the parent-child relation. The 'j' stands for job and the 'e' for evidence. This column can be used to find all children of an evidence, or to find the parent of an evidence. See the examples alongside.
- location
  The location column contains a verbose description of the actual path to the evidence. For example:

```
"/testrun1/0_NTFS/<root>/Documents and Settings/All Users/Application
Data/Microsoft/User Account Pictures/Default Pictures/beach.bmp/slack"
```

3.2.2 Item table

The item table is filled during the kickstart process, and contains columns describing the seized evidence. This links the database to the actual human labeled evidence. The identifying columns are:

- casename. The name of the case seized.
- evidencesource. The source of the seized evidence. Sometimes this field is used to annotate the premises location name or it can refere to the label on the seized evidence.
- item. The item within an evidence source or the label on the seized evidence.

3.2.3 Evidencestoreentity table

The “repname” column in the evidencestoreentity table contains the path name of the evidence stored in the file-tree based evidence store. The path is normally build from the sha1 hash value of the file content. The root directory of the repository is defined in the case configuration file.

repname character varying(255)
/b4/9d/7f48300701235231f6b6fc3d92a5630f9e70
/53/ea/2cb716f312714685c92b6be27e419f8c746c
/da/39/a3ee5e6b4b0d3255bfef95601890afd80709
/da/39/a3ee5e6b4b0d3255bfef95601890afd80709
/da/39/a3ee5e6b4b0d3255bfef95601890afd80709
/27/b1/e1d91dd07e22d3ef9de74c7ed0d98e9fead7

3.2.4 Metastoreentity table

The most important column of the metastoreentity table is the “content”. This column contains the XML description of the washing process of this particular evidence. This XML contains all extracted meta information of the evidence and all jobs involved in processing the evidence. The XML will be explained later on in the document.

3.2.5 rowtable and tbltable

The rowtable and tbltable are represented as a single abstract representation of the different meta tables found in the database. The rowtables represent the single value based meta and the tbltable represent the array based meta. This means that the number of columns in the tbltable can differ depending on the meta array dimension. "metacol_n" is an abstract notation of n numbers of meta columns.

The type of the meta columns is derived "*by example*" from the XML by the datastore module (DSM). Most of the tables in the database model are linked to the OCFA architecture. The rowtable and tbltable are different. These tables could be re-factored easily by a redesign of the DSM if they don't fit the needs of the interface developer.

3.3 Links

Nevertheless the absence of foreign keys in the database model, there are import links between the tables. These links are needed for variety of different query's. The following links exist:

Table	metadatainfo	evidencestoreentity
Linking column	dataid	id

Table	metadatainfo	metastoreentity
Linking column	metadataid	id

Table	metadatainfo	item
Linking column	itemid	itemid

Table	metadatainfo	rowtable
Linking column	metadataid	metadataid

Table	metadatainfo	LuceneIndex
Linking column	metadataid	metadataid

Table	evidencestoreentity	LuceneIndex
Linking column	id	docid

3.4 XML

The XML of an evidence is stored in the content column of the metastoreentity table. The jobs concerning the routing and some trivial jobs are removed from the example xml below. The example is about a jpeg image taken with a Nikon e950 camera.

```
<evidence case="badguy" id="0.j0e5.j0e1.j0e1.j0e12" item="run1"
md5="b4204dd79d4b5e0c130e4c98e9dbbeaf" sha="3730cac5130d4a560e25d2f1e67adabce2812930"
src="testset" status="NEW" storeref="264">
  <location name="nikon-e950.jpg">/0/extractor/exif/data</location>
  <job etime="2010-10-27T01:47:49" status="DONE" stime="2010-10-27T01:47:49">
    <moduleinstance host="10.31.79.158" instance="java1288180032805" module="snorkelkickstart"
namespace="default"/>
    <meta name="inodetype" type="scalar">
      <scalar type="string">file</scalar>
</meta>
    <meta name="nodetype" type="scalar">
      <scalar type="string">file</scalar>
</meta>
    <meta name="size" type="scalar">
      <scalar type="string">164151</scalar>
</meta>
</job>

  <job etime="2010-10-27T13:52:27" status="DONE" stime="2010-10-27T13:52:27">
    <moduleinstance host="127.0.0.1" instance="Inst52364" module="file" namespace="default"/>
    <meta name="mimetop" type="scalar">
```

```
      <scalar type="string">image</scalar>
    </meta>
    <meta name="mimetype" type="scalar">
      <scalar type="string">image/jpeg</scalar>
    </meta>
    <meta name="mimeinfo" type="scalar">
      <scalar type="string">ocfa_undef</scalar>
    </meta>
    <meta name="fileinfo" type="scalar">
      <scalar type="string">JPEG image data, JFIF standard 1.02</scalar>
    </meta>
    <meta name="fileextension" type="scalar">
      <scalar type="string">jpg</scalar>
    </meta>
  </job>

  <job etime="2010-10-27T13:52:35" status="DONE" stime="2010-10-27T13:52:35">
    <moduleinstance host="127.0.0.1" instance="Inst52367" module="exif" namespace="default"/>
    <meta name="CameraBrand" type="scalar">
      <scalar type="string">NIKON</scalar>
    </meta>
    <meta name="CameraModel" type="scalar">
      <scalar type="string">E950</scalar>
    </meta>
    <meta name="CameraSoftware" type="scalar">
      <scalar type="string">v981-79</scalar>
    </meta>
    <meta name="ImageOrientation" type="scalar">
      <scalar type="string">Top, Left-Hand</scalar>
    </meta>
    <meta name="HorizontalResolution" type="scalar">
      <scalar type="string">300 dpi</scalar>
    </meta>
    <meta name="VerticalResolution" type="scalar">
      <scalar type="string">300 dpi</scalar>
    </meta>
    <meta name="dtPictureTaken" type="scalar">
      <scalar type="datetime">2001-04-06T11:51:40:E950</scalar>
    </meta>
    <meta name="ExposureTime" type="scalar">
      <scalar type="string">1/77 sec</scalar>
    </meta>
    <meta name="FNummer" type="scalar">
      <scalar type="string">f/5.5</scalar>
    </meta>
    <meta name="ExposureProgram" type="scalar">
      <scalar type="string">Normal Program</scalar>
    </meta>
    <meta name="ISORating" type="scalar">
      <scalar type="int">80</scalar>
    </meta>
    <meta name="ExposureBias" type="scalar">
      <scalar type="string">0 EV</scalar>
    </meta>
    <meta name="MeteringMode" type="scalar">
      <scalar type="string">Pattern</scalar>
    </meta>
    <meta name="LightSource" type="scalar">
```

```
      <scalar type="string">Unknown</scalar>
    </meta>
    <meta name="Flash" type="scalar">
      <scalar type="string">No Flash</scalar>
    </meta>
    <meta name="FocalLength" type="scalar">
      <scalar type="string">12.80 mm</scalar>
    </meta>
    <meta name="ColorSpaceInformation" type="scalar">
      <scalar type="string">sRGB</scalar>
    </meta>
    <meta name="Width" type="scalar">
      <scalar type="int">1600</scalar>
    </meta>
    <meta name="Height" type="scalar">
      <scalar type="int">1200</scalar>
    </meta>
  </job>
</evidence>
```

3.5 Configuration

In some circumstances you have to know where the "ocfaroot" resides.

1. The ocfaroot is stored as a parameter in the environment
   env(OCFAROOT)
2. The ocfaroot is the home directory of the user "ocfa".

3.5.1 Datastore configuration (dsm.conf)

Where to find the dsm.conf:

- $OCFAROOT/etc/dsm.conf

The configuration file "dsm.conf" describes which of the meta information stored in the XML file will also have a table representation in the database. All these meta tables will start with the prefix "row" for single value mate and with "tbl" for array meta values. The dsm.conf file is a text file in which each row represents a meta value as named in the XML file.

3.5.2 Case configuration file

Where to find the case.conf file:

- $(OCFAROOT)/etc/casename.conf
- $(OCFAROOT)/etc/casename**parked**.conf

The casename referes to the casename used to process the seized evidence and is also stored in the item table. There are two configuration files, one for the processing of the case (casename.conf) and one for the parked querying optimized database (casenameparked.conf)

The case configuration file is a text based file with key value pairs. The for querying important lines are:

```
#XML schema
schemadir=/usr/local/digiwash/schema/

# Postgres database parameters
storedbhost=12.34.56.78
storedbuser=ocfa
storedbpasswd=ocfa
storedbname=testgeheim
storeimpl=pgblob

#dsm - data store module
metatabledescription=/usr/local/digiwash/etc/dsm.conf

#Router rulelist
rulelist=/usr/local/digiwash/etc/rulelist.csv

digestdbdir=/usr/local/digiwash/static/hashsets
regexfile=/usr/local/digiwash/etc/regexes.conf
categoryfile=/usr/local/digiwash/etc/categories.conf

#Lucene indexer parameters
indexdir=/var/ocfa/testgeheim/index
extraIndexCount=4
indexdir0=/var/ocfa/testgeheim/index2
indexdir1=/var/ocfa/testgeheim/index3
indexdir2=/var/ocfa/testgeheim/index4
indexdir3=/var/ocfa/testgeheim/index5
```

```
varroot=/var/ocfa/testgeheim

repository=/var/ocfa/testgeheim/repository
thumbnails=/var/ocfa/testgeheim/thumbnails
```

4 File-tree base repository (EvidenceStore)

EvidenceStore	
	path

The file-tree based repository contains all (extracted) content of the processed case. The files are stored as named with the sha1 hash value of the file content. For example, if the sha1 hash of a file is *"9cfc258cf8e1248f732bdba9b57c14b465a531e3"* then the file is stored in:

```
$(REPOSITORY_PATH)/9c/fc/258cf8e1248f732bdba9b57c14b465a531e3
```

A reference to the repository path is stored in the evidencestoreentity table in the field repname.

The nodes in the file-tree based repository could be either a real file or a link to a file or a link to a carv-path.

4.1 CarvFS and carv-path notations

A node in the file-tree based repository could be a link to a carv-path. Explanation of carv-path structures go beyond the scope of this document. You can use this links as if they were files to resolve the content of the evidence.

5 Full text index

LuceneIndex	
ES1 ES2	docid metadataid dir file item source

OCFA uses the java Apache Lucene indexer to index full text content. The index contains the following fields:

- content
  The textual content of the indexed file
- dir
  The complete path to the indexed file. Can also be resolved from the metadatainfo table. Has only lexical meaning. The real content is located in the repository.
- docid. Refers to the id in the evidencestoreentity table and can be used to retrieve the actual content. Refers also to the docid in the metadatainfo table.
- file
  The file name of the indexed file. Has only lexical meaning. The real content is located in the repository.
- item
  Refers to the item name of the evidence
- metadataid
  Refers to the metadataid in the database (table metadatainfo)
- source
  Refers to the source of the evidence

#	Sc	D	content	dir	docid	file	item	metadataid	source
)	L	;		/Asus4G/0_NTFS/<root>/Program Files/VideoLAN/VLC	6516	AUTHORS.txt	Asus4G	7238	3964652
.	)	?		/Asus4G/0_NTFS/<root>/Program Files/VideoLAN/VLC	7689	THANKS.txt	Asus4G	8570	3964652
!	)	)		/Asus4G/0_NTFS/<root>/Documents and Settings/Eigenaar/L	1997	f_00017c	Asus4G	2241	3964652
:	)	3		/Asus4G/0_NTFS/<root>/Documents and Settings/Eigenaar/L	1571	f_0001a0	Asus4G	1815	3964652
ł	)	;		/Asus4G/0_NTFS/<root>/Documents and Settings/Eigenaar/L	1882	f_000136	Asus4G	2126	3964652

6 Abbreviation

dsm	data store module.
duif	digital user interface for forensics
KLPD	Korps Landelijke Politiediensten (Netherlands Police Agency)
OCFA	Open Computer Forensics Architecture
ppq	persistent priority queue. Part of the anycast relay.