arcun 2.0 – Technische Dokumentation (v5.9.2)

Inhalt

1 *arcun 2.0:* Grundlagen

Im Gegensatz zu ***arcun*** Version 1 wird in ***arcun 2.0*** nicht nur die jeweilige Datei auf den Repository Server kopiert, sondern auch die bei ***arcun 1.0*** nur lokal vorhandene oder in der Datenbank abgebildete ***arcun*** XMP Datei[1]. Die ***arcun*** XMP Datei enthält alle Angaben über die Datei, den lokalen Speicherort und Zeitpunkt des Speicherns im Repository und den Schlüssel der Datei im Repository.

Für die eigentliche Datei *(Payload)* kann weiterhin ein Container, im Augenblick nur ZIP, gewählt werden. Der Container erlaubt das Komprimieren und Verschlüsseln der Datei. Die Payload-Datei oder die Payload-Datei eingepackt in den Container erhält als Dateiname oder Dateischlüssel den SHA-2[2] *hash* über die Payload-Datei (Achtung, nicht über dem ZIP-Container).

Damit ist sichergestellt, dass die gleiche Datei auf dem WORM-Repository nur einmal gespeichert wird.

Lokal wird ebenfalls eine eine ***arcun*** XMP-Datei geschrieben. Alternativ oder parallel dazu kann diese Information auch in eine lokale ***arcun*** Datenbank geschrieben werden.

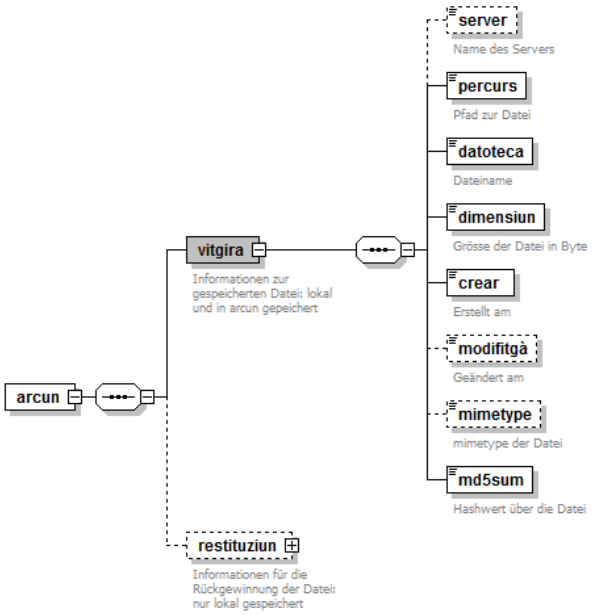
Wie oben erwähnt wird die XMP-Datei in ***arcun 2.0*** neu auch auf dem Repository abgelegt. Damit beim Umbenennen oder Verschieben einer Datei eine neue XMP-Datei mit den neuen Dateiinformationen erzeugt und auf dem Repository abgelegt werden kann, erhält die jeweilige XMP-Datei einen Dateinamen / Schlüssel im

[1] Das Format der XMP-Datei ist weiter unten beschrieben.

[2] SHA-2 in der Variante SHA-256, siehe dazu: https://de.wikipedia.org/wiki/SHA-2

Repository, der aus der Payload-Information (Frachtgut = *vitgira*) der XMP-Datei gebildet wird.

Der XMP-Dateischlüssel besteht aus einem SHA-2 *hash* über dem folgendem String: *percurs;datoteca;dimensiun;crear;md5sum*[3] getrennt durch Strichpunkt, ohne *white space* und ohne *Newline*-Zeichen am Zeilenende. *percus* (Pfad) und *datoteca* (Dateiname) sind UTF-8 kodiert, Leerschläge in Dateiname und Pfad müssen bewahrt werden. *crear* (Creation Time) ist in XML-Notation.



Hier ein Beispiel:

```
<arcun:vitgira>
        <arcun:server/>
        <arcun:percurs>Demo/ELAR/fünf</arcun:percurs>
        <arcun:datoteca>5.1.09.tiff</arcun:datoteca>
        <arcun:dimensiun>65670</arcun:dimensiun>
        <arcun:crear>2016-10-14T14:32:28</arcun:crear>
        <arcun:modifitgà/>
        <arcun:mimetype>image/tiff</arcun:mimetype>
        <arcun:md5sum>6f55c0426145e99ac3bc8fd72b21a883</arcun:md5sum>
</arcun:vitgira>
```

als String:

`Demo/ELAR/fünf;5.1.09.tiff;65670;2016-10-14T14:32:28;6f55c0426145e99ac3bc8fd72b21a883`

ergibt folgenden SHA-2-Schlüssel bzw. Dateinamen:

`360335de8c4e00bf0d0f586d1b72d9da4ed07b89d4d3cb64291df9d8ae652342.xmp`

Erstens ist so gesichert, dass eine identische Datei nur einmal unter ihrem SHA-2-Schlüssel auf dem Repository abgelegt wird. Jedes Kopieren, Verschieben und/oder Umbenennen erzeugt nur zusätzliche XMP-Dateien.

Zweitens kann das gesamte lokale Repository auch ohne ***arcun***-Datenbank durch Lesen aller XMP-Dateien wieder aufgebaut werden.

Drittens ist es auch ohne lokale Datenbank oder lokale XMP-Datei möglich, die entsprechende XMP-Datei im Repository zu finden, d.h. den Schlüssel für die XMP-

[3] Für die Übersetzung der Tag-Namen: http://www.pledarigrond.ch/rumantschgrischun/

Datei zu generieren und entweder die lokale Datei zu validieren oder von dort via *accessID* auf die im Repository abgelegte Datei zuzugreifen.

2 Erweiterungen und Änderungen in arcunTAR 2.0

Neu steht die Transfer Option /AWS mit [AccessKeyId] und [SecretKey] als Ersatz der Option /LTA zur Verfügung:

```
:: Transfer Options :
/LTA:2342119993456330881 2:k145ST1@my.fotobank.de
/AWS:account:id:key@server:port@region@bucket
             :: account = Account Name (e.g. kosttest)
             :: id = Access Key ID, key = Secret Access Key,
             :: server:port = REST Endpoint (e.g. s3.amazonaws.com:443)
             :: region = eu-central-1 or defaultAWS,
             :: bucket[/folder...]
```

Ein Beispiel:

```
/AWS:kosttest:4079ce29b2bbe7d729649cc2a1e4bda0/KOSTTEST:DC41AFE26B4D45D59F7DE4010D3A@
     ds21S3.swisscom.com:443@defaultAWS@kosttest
```

Wird die Transfer Option /AWS gewählt, ist der Dateiname der *Upload*-Datei der SHA-2-Schlüssel der ursprünglichen Datei.

Weil der Swisscom AWS Server im *one-to-one-Mode* und nicht im *flat-Mode* betrieben wird (*one-to-one* heisst eigentlich *file-Mode*), muss künstlich eine Ordnerstruktur eingefügt werden. Durch Abspalten von je zwei Zeichen des SHA-2 Schlüssel erhalten wird zwei Ordnerebenen mit max. 65‘536 Ordnern, bei maximal 32‘000 Dateien pro Ordner können zwei Milliarden Dateien abgelegt werden.

Die *accessID* sieht also, ergänzt mit dem Bucketnamen, wie folgt aus:

```
bucket/[0-9a-f][0-9a-f]/[0-9a-f][0-9a-f]/SHA-2.zip
```

im Beispiel:

```
kosttest/98/e5/98e54547eebe0bb307272f0b1d40eb79160e70584290566454b3386577b7ec0e.zip
```

Des Weiteren wird die XMP-Datei ebenfalls auf den Repository Server geladen.

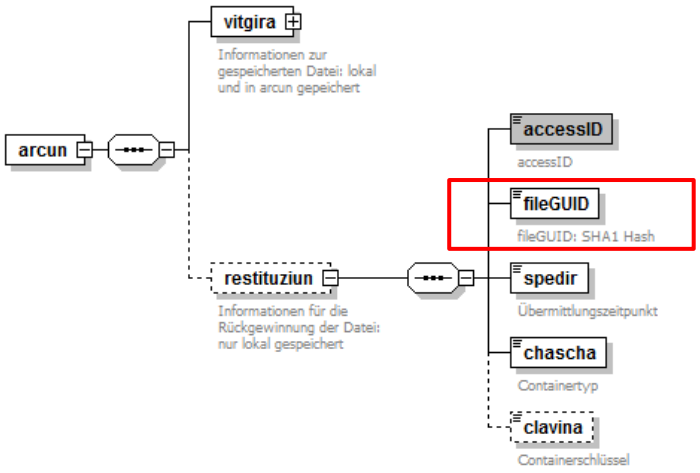
Zur Speicherung auf dem AWS-Repository wird für die XMP-Datei ein Dateiname bzw. eine URL nach den Regeln der *Upload*-Datei erzeugt:

```
kosttest/55/2a/552a4f2b4a53a4d9cabd90edd471ffe5d4001cc0126cf2521281dd09d1535f36.xmp
```

fileGUID in der XMP-Datei enthält neu den Dateinamen bzw. die URL der XMP-Datei, der sich wie oben beschrieben aus dem SHA-2 *hash* über dem String

percurs;datoteca;dimensiun;crear;md5sum

berechnet.



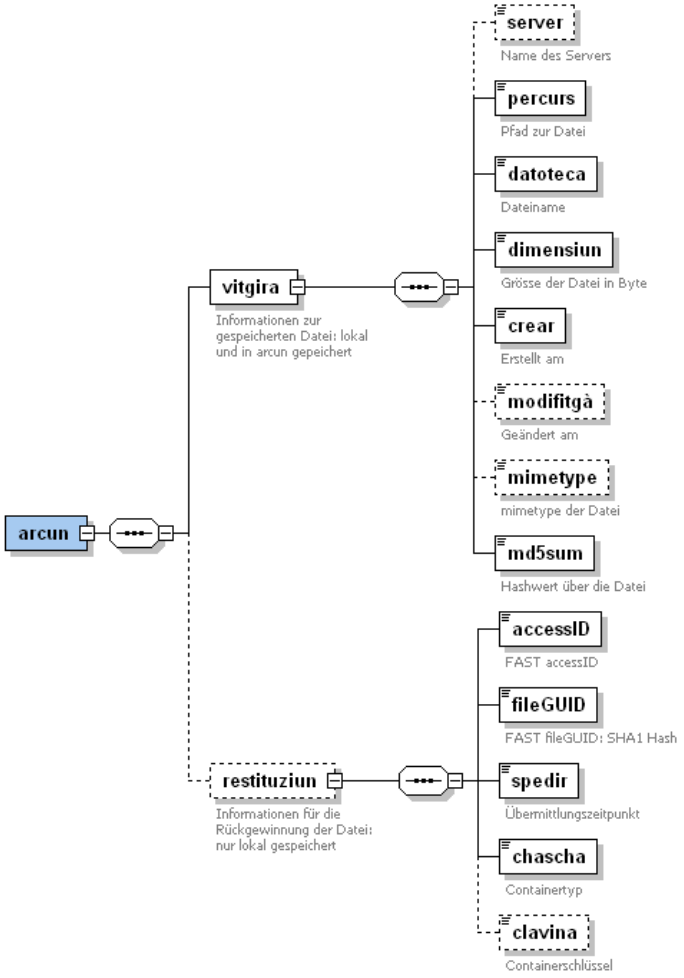
3 XMP-Datei

Hier wird im Detail die *arcun* XMP-Datei[4] beschrieben, ihr Aufbau ist gegenüber *arcun 1.0* weitgehend unverändert. XMP-Datei und Datenbankrecord sind weitgehend analog aufgebaut, deshalb werden sie hier auch zusammen behandelt. Die Elemente bzw. Feldnamen sind dem Rätoromanischen entnommen. Hier die Übersetzung:

vitgira:	Informationen zur gespeicherten Datei (lokal und in *arcun* gepeichert)
server:	Name des Servers
percurs:	Pfad zur Datei
datoteca:	Dateiname
dimensiun:	Grösse der Datei in Byte
crear:	erstellt am
modifitgà:	geändert am
mimetype:	mimetype der Datei
md5sum:	Hashwert über die Datei
restituziun:	Informationen für die Rückgewinnung der Datei (nur lokal gespeichert)
accessed:	FAST accessID
fileGUID:	FAST fileGUID (sha1 Hash)
spedir:	Übermittlungszeitpunkt
chascha:	Containertyp; zip, jar, tar
clavina:	Containerschlüssel

[4] Die Extensible Metadata Platform (XMP) ist ein Standard, um Metadaten in digitale Medien einzubetten. Der Standard wurde von Adobe im Jahr 2001 veröffentlicht. Der Standard steht mit Spezifikationen und einem SDK unter einer Open-Source-Lizenz zur Verfügung
http://de.wikipedia.org/wiki/Extensible_Metadata_Platform
http://www.adobe.com/products/xmp/indepth.html

Die Datenstruktur der XMP-Datei



Die Datenbank-Struktur

Hier das entsprechende *CREATE TABLE* Statement für die Datenbank:

```
CREATE TABLE xmp (
      percurs VARCHAR(258),
      datoteca VARCHAR(260),
      dimensiun LONG, crear DATE,
      modifitga DATE,
      mimetype VARCHAR(80),
      md5sum CHAR(32),
      accessID CHAR(80),
      fileGUID CHAR(40), spedir DATE,
      chascha VARCHAR(4),
      clavina VARCHAR(128),
      path VARCHAR(520),
      server VARCHAR(80)
);
CREATE INDEX idx_percurs ON xmp(percurs);
CREATE INDEX idx_datoteca ON xmp(datoteca);
CREATE INDEX idx_spedir ON xmp(spedir);
CREATE INDEX idx_path ON xmp(path);
```

Hier ein Beispiel für eine vollständige XMP-Datei (im Prinzip eine XML-Datei, die mit jedem gängigen XML-Editor gelesen werden kann:

```
<?xpacket begin="" encoding="UTF-8" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/">
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
        <rdf:Description rdf:about="UploadFile.jpg" xmlns:arcun="http://kost-ceco.ch/arcun">
          <arcun:arcun xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:arcun="http://kost-ceco.ch/arcun"
                       xsi:schemaLocation="http://kost-ceco.ch/arcun http://kost-ceco.ch/arcun/arcun.xsd">
            <arcun:vitgira>
              <arcun:percurs>Tools/workbench/1_workbench</arcun:percurs>
              <arcun:datoteca>UploadFile.jpg</arcun:datoteca>
              <arcun:dimensiun>184672</arcun:dimensiun>
              <arcun:crear>2017-01-10T10:24:11</arcun:crear>
              <arcun:modifitgà>2016-11-21T10:04:26</arcun:modifitgà>
              <arcun:mimetype>image/jpeg</arcun:mimetype>
              <arcun:md5sum>fbdc84c10f8c1cadc5556efebe2baad8</arcun:md5sum>
            </arcun:vitgira>
            <arcun:restituziun>
              <arcun:accessID>kosttest\98\e5\98e54547eebe0bb307272f0b1d40eb79160e70584290566454b3386577b7ec0e.zip</arcun:accessID>
              <arcun:fileGUID>552a4f2b4a53a4d9cabd90edd471ffe5d4001cc0126cf2521281dd09d1535f36.xmp</arcun:fileGUID>
              <arcun:spedir>2017-01-20T15:59:15</arcun:spedir>
              <arcun:chascha>zip</arcun:chascha>
              <arcun:clavina>false</arcun:clavina>
            </arcun:restituziun>
          </arcun:arcun>
        </rdf:Description>
    </rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>
```

4 S3 Browser Commandline

Auf Empfehlung unseres Dienstleisters Swisscom verwenden wir S3 Browser Commandline als direktes Interface zum Swisscom AWS Repository. Der Vorteil dabei ist, dass wir uns um die Problematik der Autorisierung und des Proxy-Handling nicht kümmern müssen.

S3 Browser kann von NetSDK Software, LLC als Freeware und Trialversion bezogen werden: http://s3browser.com/. Das Installieren ist denkbar einfach. Ein ZIP-File mit dem Programm wird am richtigen Ort entpackt und funktioniert ohne Administratorrechte. Eine Upgrade auf die Pro Version für 29$ ist aber dringend zu empfehlen, die Übertragung erfolgt in der Pro Version viel schneller und stabiler.

Zu beachten ist, dass S3 Browser sowohl Konfigurationen wie auch die Lizenz lokal, d.h. im *Roaming Profil des eingeloggten Users* gespeichert. Die S3 Browser Lizenz ist zudem noch an die Maschiene gebunden und kann nur auf drei unterschiedlichen Maschinen freigeschaltet werden.

Für Details betreffend Installation und Anlegen von Konten siehe die Installationsanleitung_arcun2.0.pdf.

5 Erweiterung arcunTAR (update/activate)

Mehrfach wurde der Wunsch nach Löschen, bzw. Bereinigen eines Datenbestandes geäussert. Gemeint ist der Fall, dass in einem Datenbestand einzelne Ordner oder Dateien umbenannt oder gelöscht und durch andere Dateien ersetzt werden. Das ist im neuen Datenmodell ***arcun 2.0*** problemlos und ohne übermässigen WORM Speicher möglich.

Es bleibt aber das Problem, dass bei einer Analyse mit *arcunTAR* Option *–l* oder *–d*[5] die Umbenannten oder fehlenden Ordner oder Dateien in der lokalen Dateiablage als fehlend reklamiert werden. Mit der Option –d werden sie in der lokalen Dateiablage automatisch wieder hergestellt.[6]

Jetzt kann es aber sein, dass umbenennen oder löschen sehr wohl beabsichtigt ist. Vom WORM Repository können sie nicht mehr entfernt werden, in der lokalen Datenbank, die ja eine Spiegelung des Repository darstellt sollen sie aber als gelöscht markiert werden.

Das geschieht folgendermassen.

1. Mit der Option *–u* oder – *–update* von *arcunTAR* wird den zwei Tabellen XMP und DIR in der ***arcun*** Datenbank eine dritte Tabelle HIST angelegt und die *accessID* aller Dateien eingetragen, die lokal nicht mehr zu finden sind, aber bereits im Repository gespeichert sind. Nach einem – *–update* Lauf werden alle Dateien die sich in der Tabelle HIST befinden nicht mehr beachtet.
2. Trifft *arcunTAR* mit der Option –c bzw. *–create* erneut auf eine Datei auf dem lokalen Fileserver die sich in HIST befindet, wird der Eintrag dort gelöscht, die Datei ist damit wieder im normalen Gebrauch.
3. [in arcunTAR_v.5.8.zip noch nicht umgesetzt, weitgehend durch Punkt 2 abgebildet] Mit der Option *–a* für – *–activate* könnte im betrachteten Dateibaum die inaktivierten/ bzw. als gelöscht markierte Datei wieder zugänglich gemacht werden und mit *–x* oder – *–extract* werden sie wieder aus dem Repository hervorgeholt.

Mit dem Script **`tweak_arcun_db.bat`** können alle mit der Option – –update als gelöscht markierte Dateien wieder aktiviert werden. Damit wird ab Version 5.9.2 ein Ersatz für die fehlende Option– *–activate* zur Verfügung gestellt.
Das Script löscht den Inhalt der Tabelle HIST.

6 Wiederherstellung bei Crash der lokalen Infrastruktur

Ein *arcun 2.0* Repository kann jederzeit auch ohne die Software *arcunTAR* und ohne die lokale Datenbank *arcunDB* aus der Kopie auf dem Cloud Server wiederhergestellt werden. Dazu sind genaugenommen zwei Codezeilen notwendig.

Als erstes wird das gesamte S3 Bucket auf einen lokalen Server kopiert, am einfachsten mit S3 Browser. Anschliessend werde alle ZIP Dateien in diesem Bucket expandiert. Weil in jeder ZIP Datei nicht nur die Primärdatein sondern auch Pfad, Dateinamen und *Creation Date* gespeichert sind, wird beim Auspacken die ganze ursprüngliche Ablage im *Working Directory* der Commandozeile wieder hergestellt.

Im Prinzip ist es auch möglich die lokale Datenbank *arcunDB* aus den heruntergeladenen XMP Dateien wiederherzustellen, dass soll hier aber nicht beschrieben werden.

Grau markiert die notwendigen PowerShell Befehle:

---

[5] `-l, --compare    compare entries in DB or XMP with local content`
`-d, --diff       compare local with remote content`

[6] Grundlegende WORM Funktion von *arcunTAR*.

```
REM Prerequisites
mkdir c:\workbench
cd c:\workbench

REM Using S3 Browser in CMD:  account “arcun2” has to be specified in
    the S3 Browser gui wizard
"c:\Program Files (x86)\S3_Browser\s3browser-con.exe" download arcun2
    arcun2 c:\workbench\arcun2

REM using S3 Browser in PowerShell:
& 'c:\Program Files (x86)\S3 Browser\s3browser-con.exe' download
    arcun2 arcun2 c:\workbench\arcun2

REM Using MinGW32 (Minimalist GNU for Windows) tools
find c:\workbench\arcun2 -name "*.zip" -exec unzip {} ;

REM Using PowerShell and UnZip 6.00 http://www.info-zip.org/
Get-ChildItem "C:\workbench\arcun2" -Recurse -Filter *.zip | ForEach-
    Object {  & unzip  $_.FullName }

REM PowerShell native “Expand-Archive” doesn't work, it creates a
    folder that has the same name as the ZIP file
REM Get-ChildItem "C:\workbench\arcun2" -Recurse -Filter *.zip |
    ForEach-Object { Expand-Archive $_.FullName “c:\tmp\$($_.Basename)"
    -Force }
```