Descriptive Information Converter 2020

# Kurzbeschreibung

Die "Descriptive Information Referenzimplementierung", neu als *"Descriptive Information Converter"* bezeichnet ist eine Webapplikation, die aber auch lokal ausgeführt werden kann. Die App konvertiert eCH-0160 Metadaten in xIsadg, EAD oder RIC <https://kost-ceco.ch/isadg/app/>

Die zur App hochgeladene Metadaten Datei wird mit einer Schema Validierung überprüft, verschiedene Metadatenschema werden erkannt; konvertiert werden können aber nur eCH-0160 Metadaten. Ziel ist das Extrahieren der ISAD(G) relevanten Metadaten, die Descriptive Information nach OAIS für den Import in ein Datamanagement oder ein Findmittel, nicht aber die vollständige Abbildung aller eCH-0160 Metadaten in einem anderen Schema. Grundlage der Konvertierung ist das Papier [Extraktion deskriptiver Metadaten aus der archivischen Ablieferungsschnittstelle eCH-0160](https://test.ech.ch/alfresco/s/ech/download?nodeid=e05d969b-1bff-4593-ad57-d751cc5473aa) und der Crosswalk [eCH-0160\_xIsadg&EAD\_v2.1.xlsx](https://kost-ceco.ch/cms/dl/91141f46022f5f2dd1f8fa9409d2dcab/eCH-0160_xIsadg%26EAD_v2.1.xlsx)

Unterstützt werden die Ausgangsformate eCH-0160 v1.0 und eCH-0160 v1.1; Zielformat ist xIsadg v1.6.1 und xIsadg v2.1, EAD in der Ausprägung METS Matterhorn Profil und RIC-O Version 0.1.

Technisch basiert die Konvertierung auf einem Set von XSLT Transformationen die von der WebApp ausgeführt werden. Die Transformationen können auch in einer Windows Batch Datei mit [Saxon9](http://saxon.sourceforge.net/) und [xmllint](http://xmlsoft.org/xmllint.html) ausgeführt werden.

Die Ausgabe (im Browser) ist jeweils eine XML bzw. eine RDF/XML Datei. Nach einer Konvertierung in RIC Format steht der RDF Content auch unter der angegebenen URI für eine SPARQL Auswertung zur Verfügung.

Die Basisdokumente für die Metadatenschema sind die folgenden *Data Dictionaries* :

[eCH-0160\_V1.1\_ArchivischeAblieferungsschnittstelle\_DataDictionary.pdf](http://ech.ch/de/dokument/4bf53ff1-0e4f-4fb9-9ccc-75a4219bb20c) und [xIsadg\_DataDictionary\_v2.1.pdf](https://kost-ceco.ch/cms/dl/b890ad3a2222023b4247d17779062619/xIsadg_DataDictionary_v2.1.pdf).

# Dokumentation

Es gibt neben dieser Dokumentation eine Ältere aus den Anfängen der App 2012/2013   
*DI-Programmbeschreibung\_2013.docx* die in weiten Teilen durchaus noch gültig ist. Die darein beschriebene prozedurale Prüfung (Aggregation & Vererbung) bei der Validierung von xIsadg-Metadaten wurden jedoch nie umgesetzt.

# Ablagestruktur

Die Ablage auf GIT Hub <https://github.com/KOST-CECO/DI-Extractor> ist wie folgt aufgebaut:

*/app* Enthält die vollständige Applikation, er muss integral auf den Webserver kopiert werden (im Augenblick <https://kost-ceco.ch/isadg/app/ingest.php> ). Im Ordner /wdir werden die hochgeladenen Metadaten-Files abgelegt, der Inhalt kann ab und zu gelöscht werden.

*/docu* Enthält die gesamte Dokumentation

*/eCH-0160-sample* Beispiel Metadaten in diversen alrelda Versionen

*/rico-sample* Beispiel RIC-O v0.1 Metadaten, mehrheitlich von den *Archives nationales de France*

*/xIsadg-sample* Beispiel Metadaten in xIsadg v1.6, v2.0 und v2.1 aus mehreren Quellen

*/technotes* Links zu weiteren Informationen

*/xslt\_ead-rico* EAD -> RIC-O v0.1 Transformation der *Archives nationales de France*

*/xslt\_sample* Beispiele für XSL Transformationen

├───app

│ ├───fonts

│ ├───ric

│ ├───wdir

│ ├───xsd\_v3.13.2

│ ├───xsd\_v4

│ └───xsd\_v4.1

├───docu

├───eCH-0160-sample

├───rico-sample

├───xIsadg-sample

├───technotes

├───xslt\_ead-rico

└───xslt\_sample

# Programmaufbau

Die Metadaten Transformationen werden jeweils durch ein Set von XSLT Transformationsdateien ausgeführt. Ein Set ist jeweils durch Kürzel am Dateianfang gekennzeichnet, dass das Zielformat bezeichnen soll. Gestartet wird das Set jeweils durch eine sprechende Transformationsdatei, welcher eine Anzahl von Parametern übergeben werden kann:

eCH2xIsadg.xsl und xI…xsl

eCH2xI2sadg.xsl und xI2…xsl

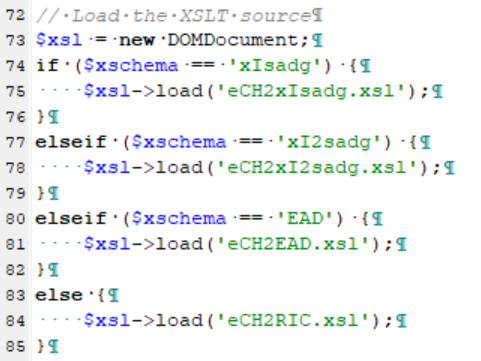
eCH2EAD.xsl und EAD…xsl

eCH2RIC.xsl und RIC….xsl

Der Aufbau der Transaktionen ist immer ähnlich und richtet sich nach der Struktur von eCH-0160. Detailliert ist das in *DI-Programmbeschreibung\_2013.docx* Kapitel 2.2 *Aufbau der XSL-Transaktion*, Seite 7, beschrieben.

Die Web App ist ein PHP Applikation, die i.A. unter PHP 7.2 läuft. Der Aufbau der PHP Applikation ist ebenfalls in *DI-Programmbeschreibung\_2013.docx* Kapitel 2.1 *Aufbau der WebAPP,* Seite 5, beschrieben.

Eine Weiter in XSLT formulierte Transformation kann einfach in *processingest.php* Zeile 72 ff. eingefügt werden:



# Entwicklungsumgebung und Testen

Für PSPad steht eine PSPad Projektdatei *\_DI\_app.ppr* zur Verfügung, die einen guten Überblick über die Programmdateien vermittelt (<http://www.pspad.com/> )

Es empfiehlt sich die PHP und XSL Dateien schreibgeschützt zu halten, das wird leider von GIT nicht unterstützt.

Mit Saxon9 können die Transformationen lokal auf dem Windows Rechner getestet werden. Dazu gibt es eine Reihe von Batch Dateien, die aber auf den BIT Rechner nicht ausgeführt werden können. Dort muss im CMD Fenster getestet werden. Eine Reihe von Beispielen sind in der Datei *run.txt*

Dieses Beispiel konvertiert die Datei *RIS\_St.Gallen\_3Dossier\_arelda.xml* in RIC/RDF mit der Konvertierung *eCH2RIC.xsl* in die Ausgabedatei *scratch.rdf* mit einer Reihe von Parametern: *fondtitle, archsig* und *baseuri* mit dezimaler Signatur:

REM Dezimale Signatur 1, 1.1, 1.2, ----------------------------------------

java -jar C:\JTools\saxon9\saxon9.jar -versionmsg:off -s:"..\rico-sample\RIS\_St.Gallen\_3Dossier\_arelda.xml" -xsl:eCH2RIC.xsl -o:"scratch.rdf" fondtitle="RIS ein Dossier" archsig="sig" baseuri="http://arelda\_v4.0.ch/" reffilename="null.xml"

Eine Konvertierung mit fortlaufender Signaturnummerierung ist ausserhalb der Webapp etwas aufwändiger und muss in drei Schritten erfolgen:

REM Fortlaufende Signatur 1, 2, 3, ----------------------------------------

REM create unique reference for each archival object

java -jar C:\JTools\saxon9\saxon9.jar -versionmsg:off -s:"..\rico-sample\RIS\_St.Gallen\_3Dossier\_arelda.xml" -xsl:createRef.xsl -o:"createRef.xml" fondtitle="RIS ein Dossier" archsig="sig"

REM create running number for each archival object

java -jar C:\JTools\saxon9\saxon9.jar -versionmsg:off -s:createRef.xml -xsl:elementRef.xsl -o:numberRef.xml fondtitle="RIS ein Dossier" archsig="sig"

java -jar C:\JTools\saxon9\saxon9.jar -versionmsg:off -s:"..\rico-sample\RIS\_St.Gallen\_3Dossier\_arelda.xml" -xsl:eCH2RIC.xsl -o:"scratch.rdf" fondtitle="RIS ein Dossier" archsig="sig" baseuri="http://arelda\_v4.0.ch/" reffilename="numberRef.xml"

# App installieren

Das Installations-Script *install\_app.cmd* installiert alle notwendigen Dateien in den gewählten Ordner *htdocs* zusammen mit einer ZIP Datei, welche von der Webapp aus referenziert wird <https://kost-ceco.ch/isadg/app/eCH-0160_xIsadg&EAD&RIC.xsl.zip>

Diese ZIP Datei enthält alle für die Transformationen notwendigen Dateien.

# SPARQL Abfragen

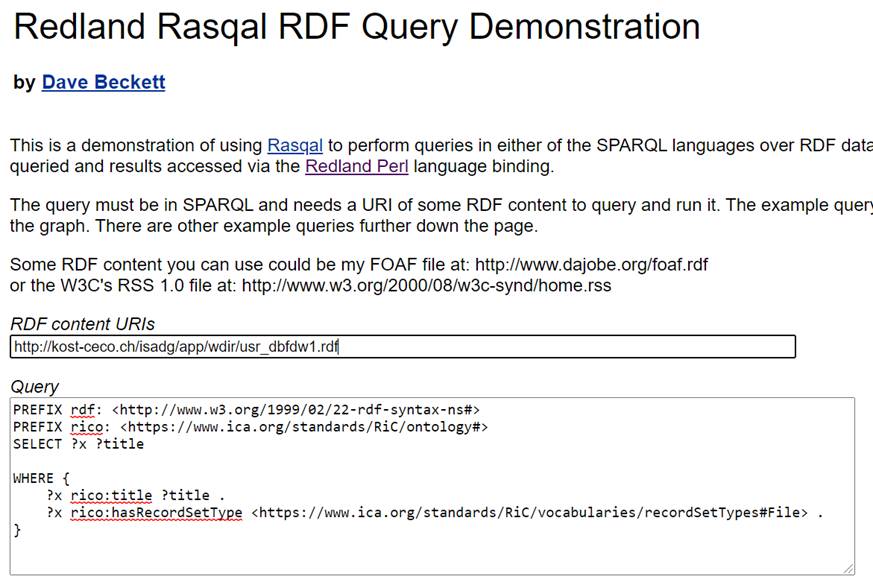
Nach einer Konvertierung in RIC Format steht der RDF Content auch unter der angegebenen URI für eine SPARQL Auswertung zur Verfügung.

Diese Abfrage liefert die Dossiertitel in einem SIP

PREFIX rdf: <[http://www.w3.org/1999/02/22-rdf-syntax-ns#](http://www.w3.org/1999/02/22-rdf-syntax-ns)>  
PREFIX rico: <[https://www.ica.org/standards/RiC/ontology#](https://www.ica.org/standards/RiC/ontology)>  
SELECT ?x ?title  
FROM <<http://kost-ceco.ch/isadg/app/wdir/usr_dbfdw1.rdf>>  
WHERE {  
 ?x rico:title ?title .  
 ?x rico:hasRecordSetType <<https://www.ica.org/standards/RiC/vocabularies/recordSetTypes#File>> .  
}

Achtung, bei der Abfrage mit Redland Rasqal RDF Query <http://librdf.org/query/>

funktioniert das FROM Statement nicht, stattdessen *RDF content URI* verwenden



Alternativ kann die RDF Datei auch heruntergeladen und lokal mit twinkle <http://ldodds.com/projects/twinkle/> befragt werden.