**NEW SAMPLE ENTITY CREATION - STEP BY STEP REFERENCE**

**ENTITY CREATION**

Entities are the high-level business objects in an enterprise. MarkLogic Entity Services is a set of tools and interfaces that make it easier to create applications that manipulate these business entities, even when your raw data has a different structure.

Here is the sample JSON model descriptor that defines a TestEntityNew entity type with the following properties.:

" TestEntityNew ": {

"properties": {

"EmployeeID": {"datatype": "string"},

"employeeName": {"datatype": "string"},

"DOB": {"datatype": "string"},

"Account": {"datatype": "string"},

"ManagerName": {"datatype": "string"}

},

...

}

**1.CREATE THE NEW ENTITY FOLDER UNDER**

Project\plugins\entities

Eg : \credistore-dh3\plugins\entities\TestEntityNew

**2.SAMPE ENTITY.JSON** - An entity instance is a concrete instantiation of an entity type defined in a model.

Now for the above model, create the entity.json file under the entity directory.(eg: \credistore-dh3\plugins\entities\TestEntityNew)

TestEntityNew.entity.json

{  
 "info": {  
 "title": "TestEntityNew",  
 "version": "1.0.0",  
 "baseUri": "http://nl.abnamro.com/cre/cdm/",  
 "description": "New Test Entity"  
 },  
 "definitions": {  
 "TestEntityNew": {  
 "required": [],  
 "namespace": "http://nl.abnamro.com/cre",  
 "namespacePrefix": "cre",  
 "properties": {  
 "EmployeeID": {  
 "datatype": "string",  
 "collation": "http://marklogic.com/collation/codepoint"  
 },  
 "employeeName": {  
 "datatype": "string",  
 "collation": "http://marklogic.com/collation/codepoint"  
 },  
 "DOB": {  
 "datatype": "string",  
 "collation": "http://marklogic.com/collation/codepoint"  
 },  
 "Account": {  
 "datatype": "string",  
 "collation": "http://marklogic.com/collation/codepoint"  
 },  
 "ManagerName": {  
 "datatype": "string",  
 "collation": "http://marklogic.com/collation/codepoint"  
 }  
 }  
 }  
 }  
}

**3.SAMPLE INPUT DATA** – Here are some sample input data,

(Data saved in filesystem in delimited text format with delimiter as ‘~’

- C:\Users\C58851\testentitytestData\TestData )

Filename **: TestData.csv**

EmployeeID~employeeName~DOB~Account~ManagerName~ACTION~URI

A56824~John~28/12/1991~ABN~Thomas~N~xyz/testdata-38f9aca147c14cb5a84c

B785663~Chris~11/02/1990~ABN~Thomas~N~xyz/testdata-38f82061c5c89b5a84

C142541~Joseph~03/03/1996~Citi~Ram~N~xyz/testdata-38f82061c5c89b69aca

D12457~Mary~07/05/1995~Citi~Ram~N~xyz/testdata-69aca147c14cb5a84c

**INPUT FLOW AND RAW DATA INGESTION**

**4.INPUT FLOW** – Create the directory for the input flow inside \plugins\entities\TestEntityNew\input\

(Eg: credistore-dh3\plugins\entities\TestEntityNew\input\IngestTestEntityNew)

Files to be added in input flow:

Properties file (eg : IngestTestEntityNew.properties)

Mapping .json (eg : IngestTestEntityNew.mapping.json)

IngestTestEntityNew.properties -

#  
#Tue Mar 27 15:39:56 CEST 2018  
mainModule=../../../../com.marklogic.dmf/builtins/steps/input/main.xqy  
mainCodeFormat=xqy  
codeFormat=xqy  
dataFormat=xml

IngestTestEntityNew.mapping.json – which is having mapping settings

Eg:

{  
 "info" : {  
 "title" : "IngestTestEntityNew",  
 "version" : "0.0.1",  
 "baseUri" : "http://nl.abnamro.com/cre/cdm",  
 "description" : "Reference entity model"  
 },  
 "pathNamespaces": [  
 {  
 "prefix": "es",  
 "namespaceUri": "http://marklogic.com/entity-services"  
 },  
 {  
 "prefix": "cdm",  
 "namespaceUri": "http://nl.abnamro.com/cre/cdm",  
 "default": true  
 },  
 {  
 "prefix": "fn",  
 "namespaceUri": "http://www.w3.org/2005/xpath-functions"  
 },  
 {  
 "prefix": "fcm",  
 "namespaceUri": "http://abnamro.nl/data-hub/common"  
 }  
 ],  
 "definitions": {  
 "TestEntityNew": {  
 "headers": {  
 "hid": {  
 "expression": "fcm:md5(fn:string-join((EmployeeID,employeeName), '-'))"  
 }  
 }  
 }  
 }  
}

The expression will create the hid for each raw document.md5 function of particular string join will provide the unique value for hid for ach document.

**5.CREATE GRADLE TASK TO TRIGGER INGESTION :**

Go to build.gradle file in the project

Create an mlcp ingestion gradle task in buil.gradle file. The gradle task is named as loadTestEntityNew.

task loadTestEntityNew(type: com.marklogic.gradle.task.MlcpTask) **{** classpath = configurations.mlcp  
 port = mlStagingPort as Integer  
 command = "IMPORT"  
 username = mlAppName + "-ingester"  
 password = dmfIngesterPassword  
 database = mlFinalDbName  
 input\_file\_path =testEntityDataFolder  
 input\_file\_pattern = "TestData"  
 output\_collections = 'http://nl.abnamro.com/cre/raw,latest,/job-id/' + project.getProperty("jobid")  
 transform\_param = 'entity-name=TestEntityNew,flow-name=IngestTestEntityNew,options={\\"source\\":\\"xyz\\", \\"jobid\\":\\"'+ project.getProperty("jobid")+'\\"}'  
 options\_file = "ingestion-options/IngestTestEntityDataSSL.txt"  
**}**loadTestEntityNew.group = INGEST\_GROUP

The parameters passing in this task are specified in gradle-local.properties file. :

Eg.:

* port = mlStagingPort as Integer

The parameter mlStagingPort in gradle-local.properties file

mlStagingAppserverName=credits-dh-STAGING  
mlStagingPort=8010  
mlStagingDbName=credits-dh-STAGING  
mlStagingForestsPerHost=4  
mlStagingAuth=digest

* database = mlFinalDbName

The parameter mlFinalDbName in gradle-local.properties file

mlFinalAppserverName=credits-dh-FINAL  
mlFinalPort=8011  
mlFinalDbName=credits-dh-FINAL  
mlFinalForestsPerHost=4  
mlFinalAuth=digest

* input\_file\_path =testEntityDataFolder

The parameter testEntityDataFolder in gradle-local.properties file (The path of input data)

testEntityDataFolder=/C:/Users/C58851/testentitytestData/

* input\_file\_pattern = "TestData"

input\_file\_pattern will be the pattern(name) of the source data

* options\_file = "ingestion-options/IngestTestEntityDataSSL.txt"

IngestTestEntityDataSSL.txt file is present in ingestion-options directory under the project which is having the specifications for input data.

**6.ADD THE TASK IN dmfGenerateInputSDM TO GENERATING SDM.JSON IN DATA FOLDER.**

Go to dmfGenerateInputSDM task in build.gradle :

Add “ EntityName;Inputflowname “

Eg:

"TestEntityNew;IngestTestEntityNew"

**7.DEPLOYMENT**

To deploy the codes – run commands in terminal window

Gradle mlDeploy (or)

Gradle storeDeploy

To generate sdm.json – run commands in terminal window

Gradle dmfgenerateInputSDM

Gradle dmfCompileRules

Now, check :

1. Entity folder is created in project\data\entities directory and it is having the directory for input.

Eg : \credistore-dh3\data\entities\TestEntityNew

2. sdm.json and mapping.json files are created in input flow directory in the data folder.

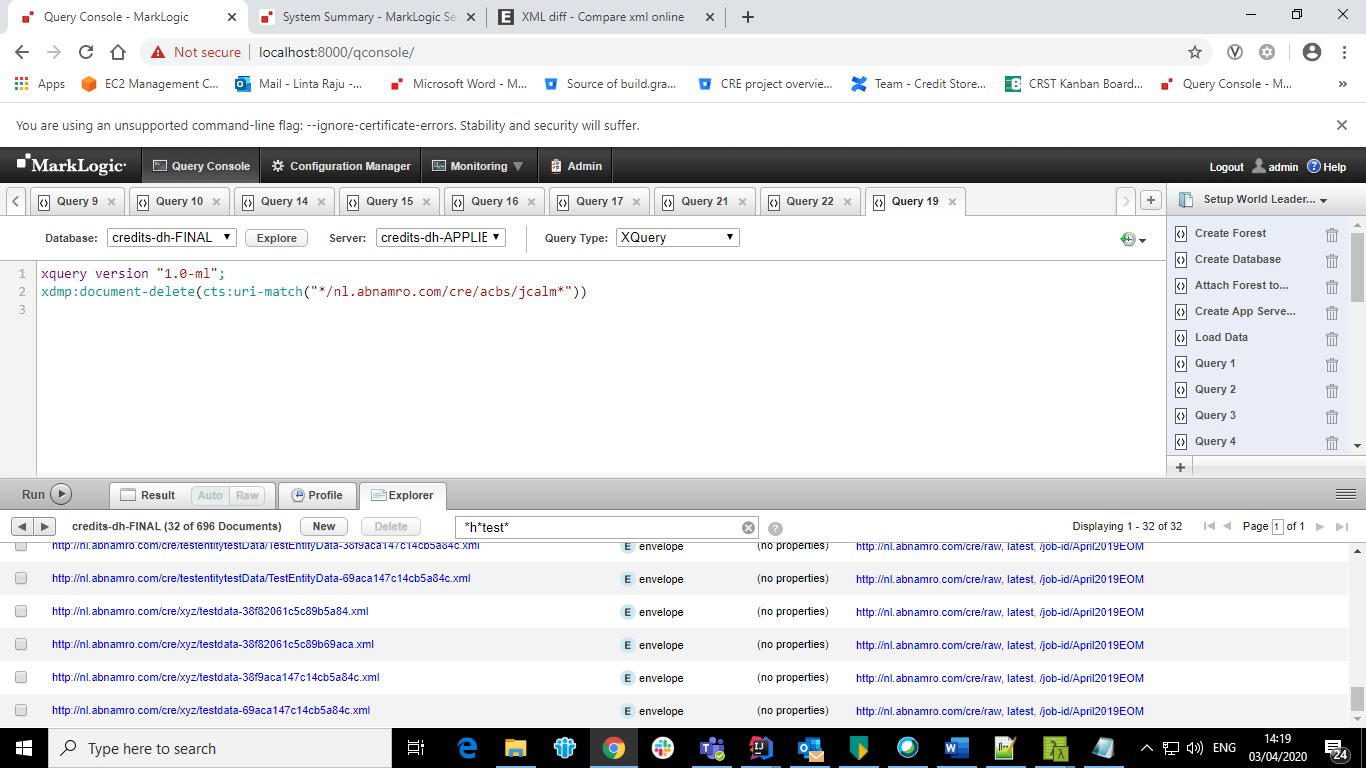
3. gradle task loadTestEntityNew is added to the list of tasks.

**8.DATA INGESTION**

To load the data , trigger the gradle task which is added in build.gradle – run command in terminal

Gradle loadTestEntityNew

Once this is completed, check the ingested data in Marklogic query console Final database.



**Ingested Raw data will Point towards following collections :**

<http://nl.abnamro.com/cre/raw>

latest

/job-id/April2019EOM

**9.INGESTED SAMPLE RAW DATA :**

<?xml version="1.0" encoding="UTF-8"?>

<envelope xmlns="http://marklogic.com/entity-services">

<headers>

<ns:ingestTime xmlns:ns="http://nl.abnamro.com/cre">

2020-04-03T13:08:56.1481447+02:00

</ns:ingestTime>

<ns:source xmlns:ns="http://nl.abnamro.com/cre">

XYZ

</ns:source>

<ns:checksum xmlns:ns="http://nl.abnamro.com/cre">

38f82061c5c89b5a84

</ns:checksum>

<ns:onboardingVersion xmlns:ns="http://nl.abnamro.com/cre">

</ns:onboardingVersion>

</headers>

<triples>

</triples>

<instance>

<ns:TESTDATA xmlns:ns="http://nl.abnamro.com/cre/xyz">

<ns:EmployeeID>

B785663

</ns:EmployeeID>

<ns:employeeName>

Chris

</ns:employeeName>

<ns:DOB>

11/02/1990

</ns:DOB>

<ns:Account>

ABN

</ns:Account>

<ns:ManagerName>

Thomas

</ns:ManagerName>

<ns:ACTION>

N

</ns:ACTION>

</ns:TESTDATA>

</instance>

<attachments>

</attachments>

</envelope>

**HARMONIZE FLOW and HARMONIZATION**

**10. HARMONIZATION REQUIREMENT**

Harmonization is the process of creating a canonical model of your data using only the parts you need and leaving the rest as is. Harmonization can be as simple as keeping the data as-is or as involved as you want to make it. Here are the few requirements for the harmonization of raw data which are ingested for TestEntityNew.

**Expected Template for Harmonization :**

> EmployeeID  - change the filed name as 'clientID' and add

"ABN" with each ID if Account = ABN

“CITI” with each ID If Account = citi

> EmployeeName - EmployeeFirstName

> ManagerName - supervisername

> DOB- change the date format

> if

                Accout = ABN , change it to ABN AMRO

   if

                Account = citi , change it to Citi Bank.

**11. HARMONIZE FLOW**

Harmonize flows is to harmonize the ingested data and are meant to run in batches. The primary purpose of the Data Hub Framework is to run harmonize flows.

**Create the directory** for the harmonize flow inside \plugins\entities\TestEntityNew\harmonize\

(Eg: credistore-dh3\plugins\entities\TestEntityNew\harmonize\TestEntityNewHarmonize)

Files to be added in harmonize flow:

Properties file (eg : TestEntityNewHarmonize.properties)

Mapping .json (eg : TestEntityNewHarmonize.mapping.json)

TestEntityNewHarmonize.properties -

#  
#Tue Apr 10 09:51:05 CEST 2018  
mainModule=../../../../com.marklogic.dmf/builtins/steps/harmonize/main.xqy  
collectorCodeFormat=xqy  
mainCodeFormat=xqy  
codeFormat=xqy  
collectorModule=../../../../com.marklogic.dmf/builtins/steps/harmonize/collector.xqy  
dataFormat=xml

TestEntityNewHarmonize.mapping.json – as per the above mentioned harmonization template.

{  
 "info": {  
 "title": "TestEntityNewHarmonize",  
 "version": "1.0.0",  
 "description": "Mapping information for the harmonize model."  
 },  
 "pathNamespaces": [  
 {  
 "prefix": "es",  
 "namespaceUri": "http://marklogic.com/entity-services"  
 },  
 {  
 "prefix": "xyz",  
 "namespaceUri": "http://nl.abnamro.com/cre/xyz",  
 "default": true  
 },  
 {  
 "prefix": "fn",  
 "namespaceUri": "http://www.w3.org/2005/xpath-functions"  
 },  
 {  
 "prefix": "fcm",  
 "namespaceUri": "http://abnamro.nl/data-hub/common"  
 }  
 ],  
 "collector": {  
 "TestEntityNew": {  
 "$ref": "#/contexts/TESTDATA"  
 }  
 },  
 "contexts": {  
 "TESTDATA": {  
 "path": "/TESTDATA",  
 "collections": [  
 {  
 "collectionsAnd": [  
 "latest",  
 "http://nl.abnamro.com/cre/raw",  
 "/job-id/April2019EOM"  
 ]  
 }  
 ]  
 }  
 },  
 "definitions": {  
 "TestEntityNew": {  
 "headers": {  
 "hid": {  
 "expression": "fcm:md5(fn:string-join((EmployeeID,employeeName), '-'))"  
 }  
 },  
 "properties": {  
 "clientID": {  
 "enabled": true,  
 "coalesce": [  
 {  
 "expression": "fn:string-join(('ABN',EmployeeID), ':')",  
 "condition": "upper-case(xs:string(Account)) = 'ABN'"  
 },  
 {  
 "expression": "fn:string-join(('CITI',EmployeeID), ':')",  
 "condition": "upper-case(xs:string(Account)) = 'CITI'"  
 }  
 ]  
 },  
 "EmployeeFirstName": {  
 "enabled": true,  
 "expression": "employeeName"  
 },  
 "supervisername": {  
 "enabled": true,  
 "expression": "ManagerName"  
 },  
 "DOB": {  
 "enabled": true,  
 "expression": "fn:tokenize(xdmp:parse-yymmdd('MM/dd/yyyy', xs:string(DOB)),'T')[1]"  
 },  
 "Account": {  
 "enabled": true,  
 "coalesce": [  
 {  
 "expression": "xs:string('ABN AMRO')",  
 "condition": "upper-case(xs:string(Account)) = 'ABN'"  
 },  
 {  
 "expression": "xs:string('Citi Bank')",  
 "condition": "upper-case(xs:string(Account)) = 'CITI'"  
 }  
 ]  
 }  
 }  
 }  
 }  
}

**12. ADD THE ENTITIES IN dmf TASKS: IN build.gradle FILE**

**(a) dmfGenerateHarmonizeSDM –** To generate sdm.json in data/entities/TestEntityNew/harmonize.

Add “EntityName;Harmonizeflowname”

Eg:

"TestEntityNew;TestEntityNewHarmonize"

**(b)dmfGenerateTemplate - –** To generate template.xml in data/entities/TestEntityNew/harmonize.

Eg:

"TestEntityNew;TestEntityNewHarmonize"

**(c)dmfGenerateSchematron –** To generate shcematron.xml in data/entities/TestEntityNew/harmonize.

Eg:

"TestEntityNew;TestEntityNewHarmonize"

**13. XSD VALIDATION FOR HARMONIZE DOCUMENT**

(a) add the entityName in dmfGenerateXSD

Eg:

"TestEntityNew"

(b) run command commands in terminal window

gradle dmfGenerateXSD

- X\_TestEntityNew.xsd will get generated under src/main/ml-schemas/xsd/core/ (based on the entity.json in the plugins/entities/TestEntityNew directory)

(c) Now go to - src/main/ml-schemas/xsd/core/group-extension.xsd

(d) make an entry of xsd file of our entity in the group-extension.xsd

Eg:

- <xs:include schemaLocation="/xsd/core/X\_TestEntityNew.xsd" />

(e) Now run the command in terminal window to load the new schemas in the database

gradle mlLoadSchemas

**14.DEPLOYMENT FOR THE CHANGES.**

gradle storeDeploy

gradle dmfCompileRules

**15.DMF ARTICRAFTS –** run these commands commands in terminal window

(a)gradle dmfCopyEntityModels

(b)gradle dmfGenerateHarmonizeSDM

(c)gradle dmfCopyHarmonizeMapping

(d)dmfDeployEntities

Now, to generate schematron.xml and template.xml in data/entities/TestEntityNew/harmonize/TestEntityNewHarmonize folder, run the commands in terminal window

(e)gradle dmfGenerateTemplate

(f) gradle dmfGenerateSchematron

(g) gradle dmfDeployEntities

Now the data/entities/TestEntityNew/harmonize/ TestEntityNewHarmonize folder will have following 4 files :

Mapping.json

Schematron.xml

Sdm.json

Template.xml

**16.RUN THE HARMONIZE FLOW.**

Command for harmonization :

gradle hubRunFlow -PentityName="TestEntityNew" -PflowName="TestEntityNewHarmonize" -PflowType="harmonize" -PthreadCount=1 -PbatchSize=1 -PsourceDB=credits-dh-FINAL -PdestDB=credits-dh-FINAL -Pdhf.source="XYZ" -PmlUsername=admin -PmlPassword=admin -PmlRestAdminUsername=credits-dh-ingester -PmlRestAdminPassword=1ng3st3r

**Once this is successful , will get similar output :**

Output:

{

"jobId" : "3843ff5d-d786-43ae-849b-788dec12693a",

"flowType" : "harmonize",

"flowName" : "TestEntityNewHarmonize",

"entityName" : "TestEntityNew",

"jobName" : null,

"startTime" : "2020-04-03T11:19:49.324Z",

"endTime" : "2020-04-03T11:19:55.641Z",

"jobOutput" : null,

"status" : "FINISHED",

"successfulEvents" : 4,

"failedEvents" : 0,

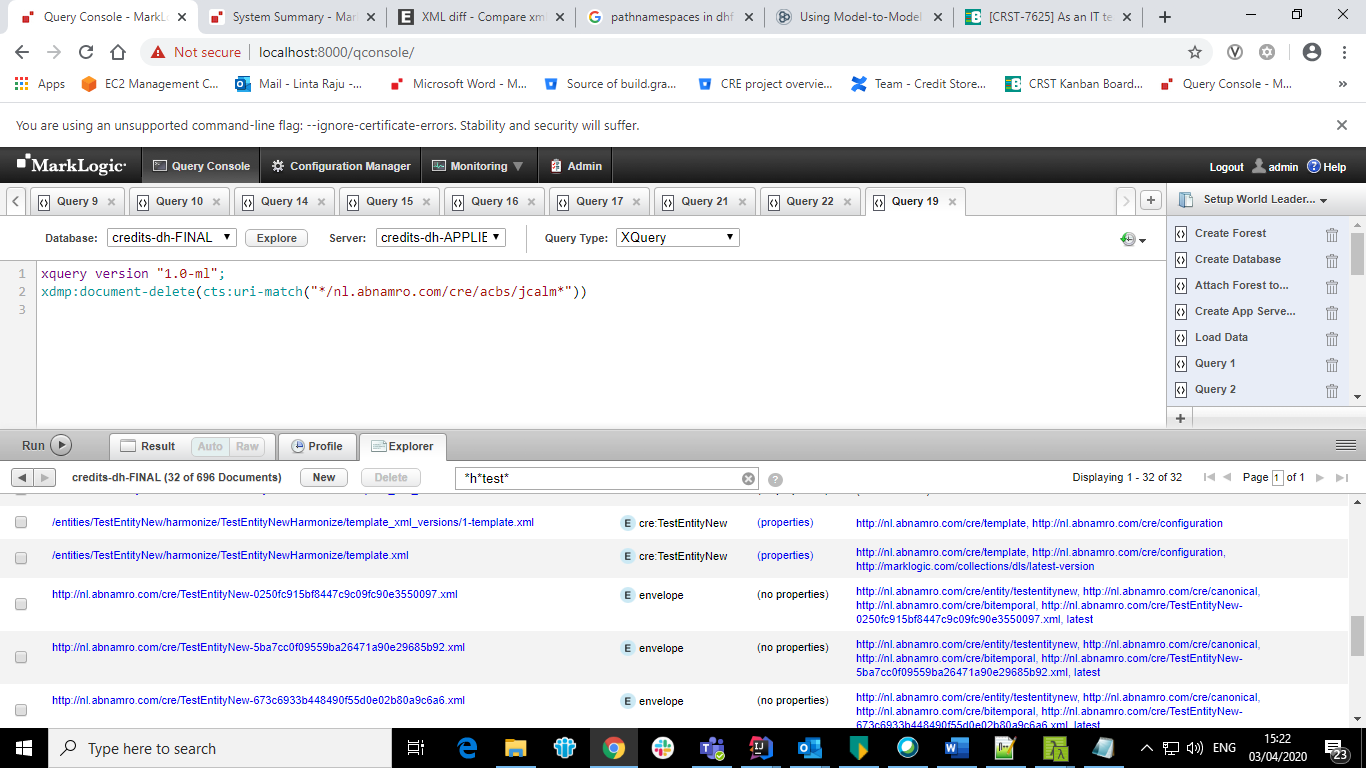
"successfulBatches" : 4,

"failedBatches" : 0

}

**17.VALIDATE THE OUTPUT IN QUERYCONSOLE :**

Once the event is successful, check the harmonized data in Marklogic query console - Final database.



**The Harmonized data will point towards the collections :**

<http://nl.abnamro.com/cre/entity/testentitynew>

<http://nl.abnamro.com/cre/canonical>

<http://nl.abnamro.com/cre/bitemporal>

http://nl.abnamro.com/cre/TestEntityNew-0250fc915bf8447c9c09fc90e3550097.xml

latest

**18. SAMPLE HARMONIZED DATA :**

<?xml version="1.0" encoding="UTF-8"?>

<envelope xmlns="http://marklogic.com/entity-services">

<headers></headers>

<triples></triples>

<instance>

<info xmlns:es="http://marklogic.com/entity-services">

<title>

TestEntityNew

</title>

<version>

1.0.0

</version>

</info>

<cre:TestEntityNew xmlns:cre="http://nl.abnamro.com/cre">

<cre:clientID>

ABN:B785663

</cre:clientID>

<cre:EmployeeFirstName>

Chris

</cre:EmployeeFirstName>

<cre:supervisername>

Thomas

</cre:supervisername>

<cre:DOB>

1990-11-02

</cre:DOB>

<cre:Account>

ABN AMRO

</cre:Account>

</cre:TestEntityNew>

</instance>

<attachments>

</attachments>

</envelope>

Now the document is harmonized as per the template in Step 10 .