FactExtractCloud

A Watson Explorer Content Analytics UIMA analysis engine for simplified information extraction.



**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Version** | **Status** | **Description** | **Author** |
| 02/06/2017 | 1.0 | Release | Initial | Martin Saunders |
|  |  |  |  |  |

**Contents**

1. Purpose 3

2. Installation 5

2.1 Pre-Requisites 5

2.2 Obtain service credentials 5

2.3 Content Analytics Studio 5

2.4 Content Analytics Server 8

3. Configuring the extraction 9

3.1 What gets persisted 9

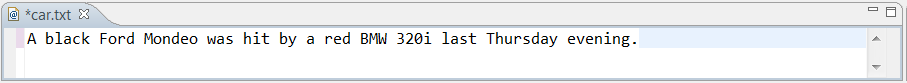
4. Running Content Analytics Collections for Information Extraction Only. 10

4.1 Turning off document indexing 10

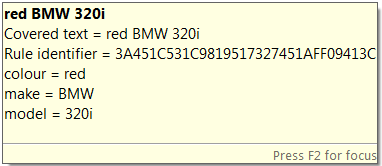
# Purpose

After text mining, information extraction projects are probably the most common usage for Watson Explorer Content Analytics. To do information extraction an annotator is configured in Content Analytics Studio to identify facts of interest and when this custom annotator is run in an Content Analytics server collection all the matching facts are extracted from the collection's corpus of documents and added to the index. These can then be exported to a relational database using standard features in the Content Analytics server. In effect Content Analytics has converted unstructured text to structured data which is typically consumed by 3rd party applications or other IBM tools such as Cognos, SPSS, i2 and DataStage. If your information extraction needs are for entities that are identified with a single value such as names, email addresses and credit card numbers these standard capabilities work very well.

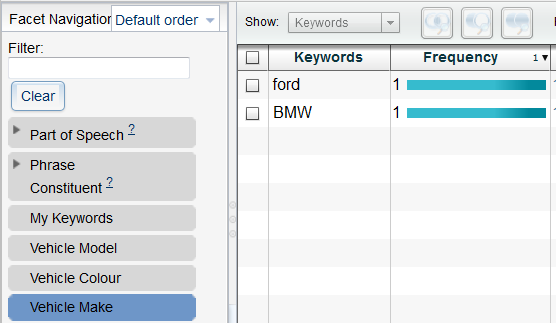
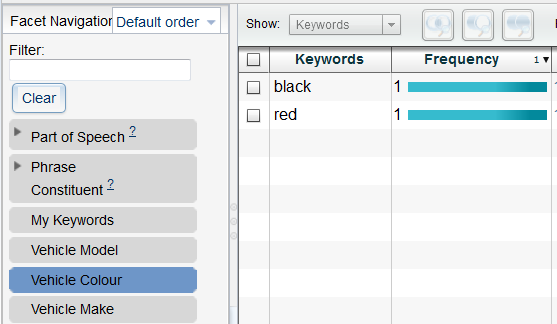
In many instances the entities you want to extract are more complex and have multiples attributes that are also in the text; think of things like a vehicle, these may have a make, model and colour . These can be modeled in Content Analytics Studio using an annotation with multiple features. The instances of annotated text together with all the feature values can also all be stored in Content Analytics server indexes (and hence exported) but the text identifying the entity and each feature value are stored separately. Effectively the implied relationship between them is lost. For example, consider a document containing the following text

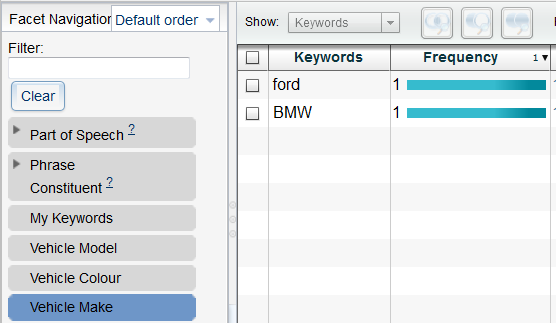
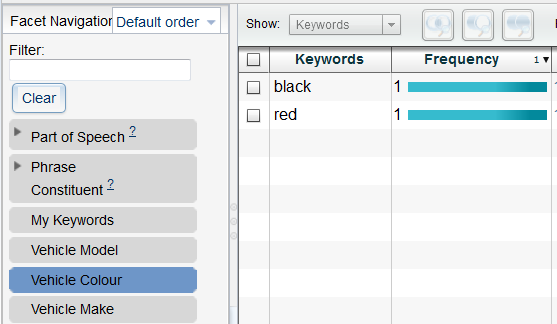


If your interest was in identifying vehicles you might well configure a com.ibm.Vehicle annotation in Content Analytics Studio that would create the following two instance with that text.

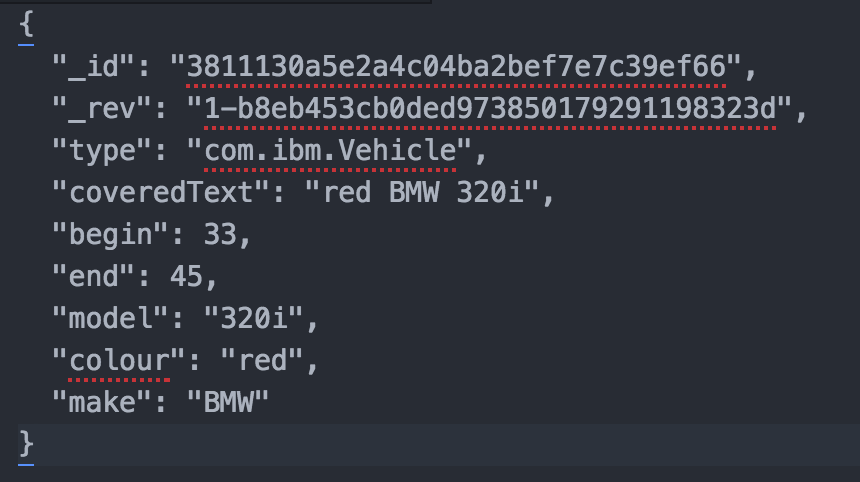
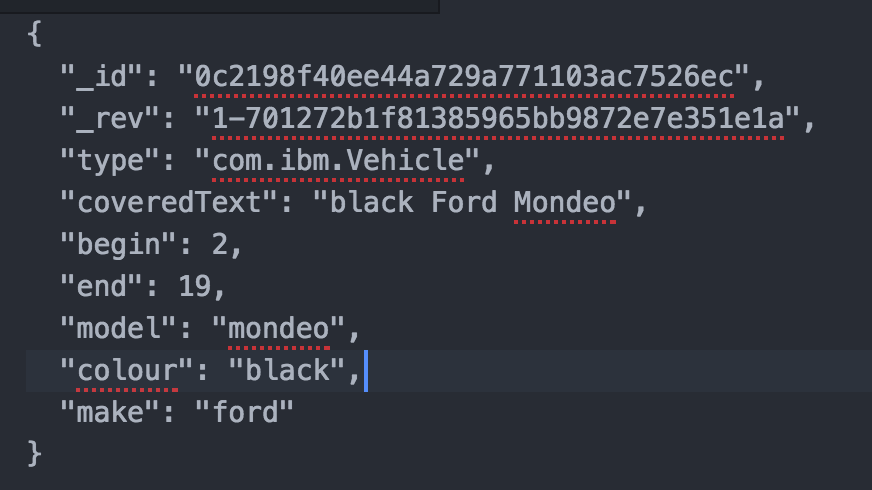


When this is deployed in an Content Analytics collection with a document containing the same text the text miner facets show:





Which vehicle is black? You cannot drill-down on ***black*** in the **Vehicle Colour** facet and expect to see one **Vehicle Make** as the answer; you’ll see both, because both **Vehicles Makes** are in are in the same document as the **Vehicle Colour *black***. If we export these facts we’ll see we’ve got a BMW, a Ford, a black vehicle and a red vehicle, and we’ll also know which document they occur in. But the only way to answer the question is to read the text. Using the FactExtractCloud overcomes this limitation as facts with multiple features can be written as a single json document to a database. Using FactExtractCloud the above text could produce the following two documents.

Creating records like this allows the relationships between features and their associated annotated text be maintained and allows for easy integration with downstream applications.

# Installation

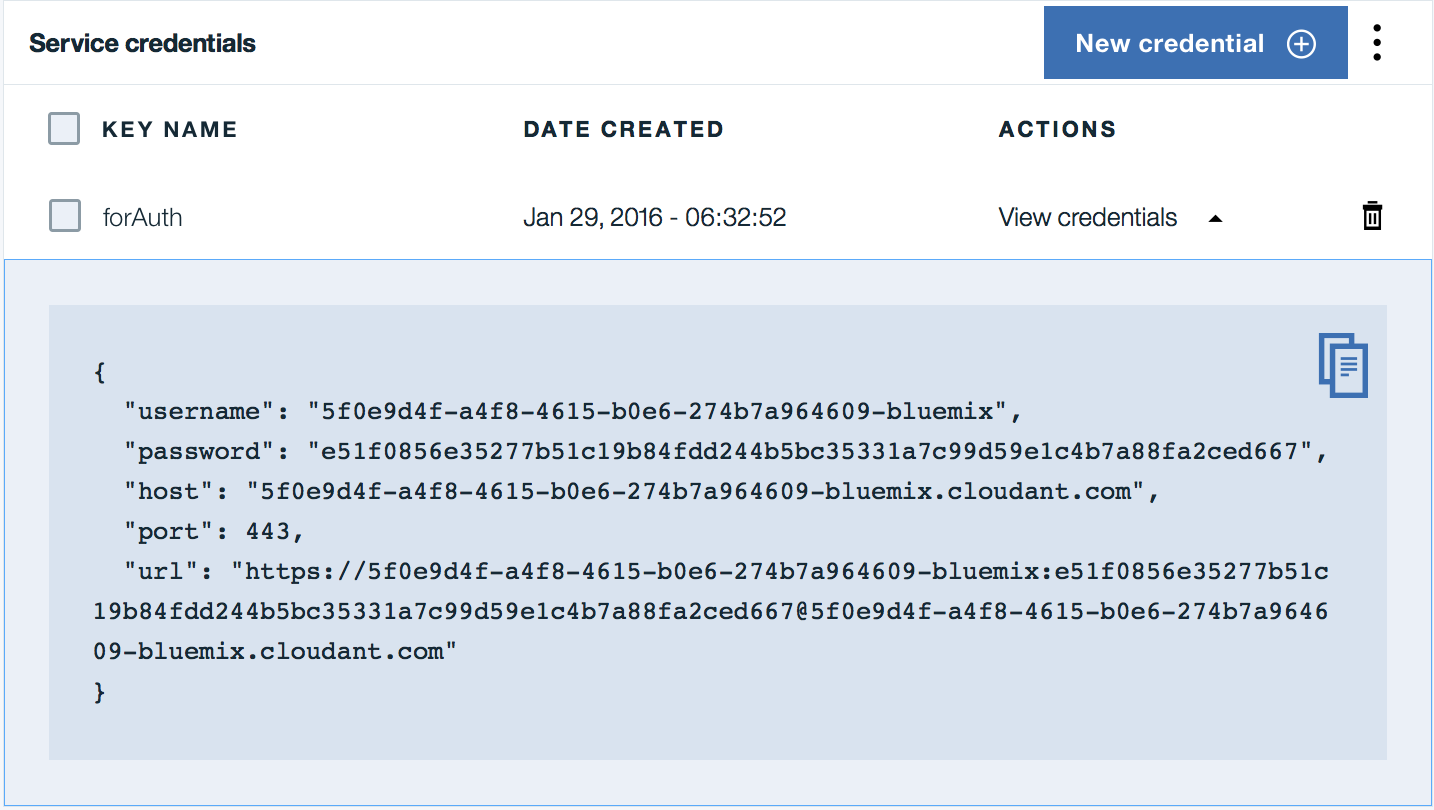
## Pre-Requisites

* Installation of Watson Explorer v11 or above with access to the administrative application via esadmin user.
* Installation of Content Analytics Studio version 11 or above
* An instance of the Cloudant service on Bluemix or equivalent
  + Access to the service credentials.
* FactExtractCloud files:
  + Core files (included in distribution):
    - FactExtractCloud-ae.xml
    - FactExtractCloud-n-n-n.jar
    - ICAUIMAUtils-n-n-n.jar
  + Cloudant and library support (included in distribution):
    - cloudant-client-2.9.0.jar
    - cloudant-http-2.9.0.jar
    - commons-codec-1.6.jar
    - commons-io-2.4.jar
    - gson-2.7.jar

## Obtain service credentials

### Login to Bluemix and create or find your existing Cloudant service instance.

### Open the credentials for that instance. They’ll look something like this



### Make a note of the **username** and **password**

### Make a note of the **host** parameter excluding the “.cloudant.com” suffix. This is your Cloudant **account** name.

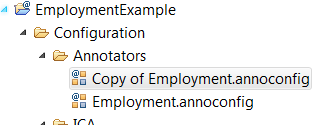
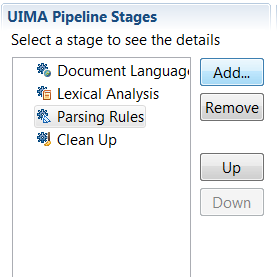
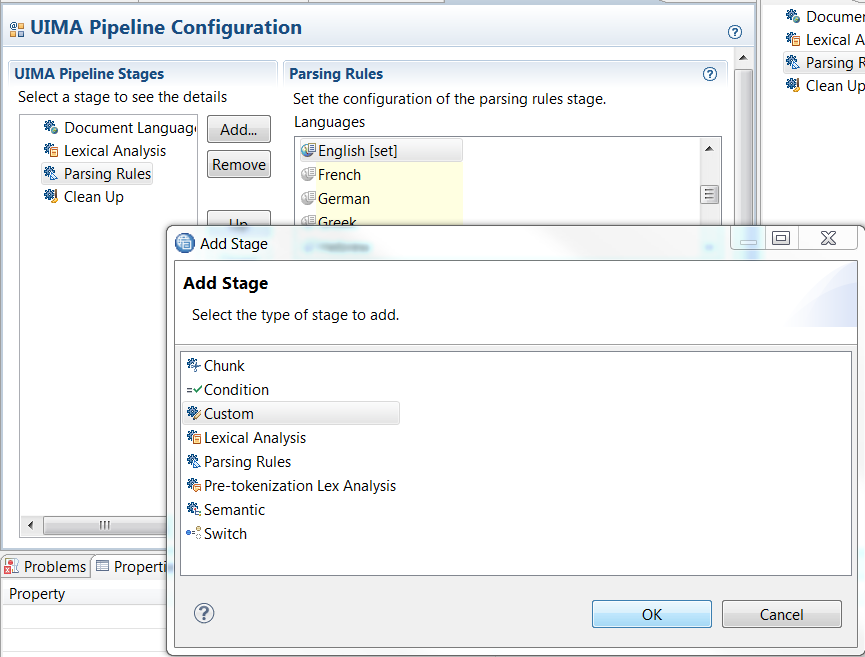
## Content Analytics Studio

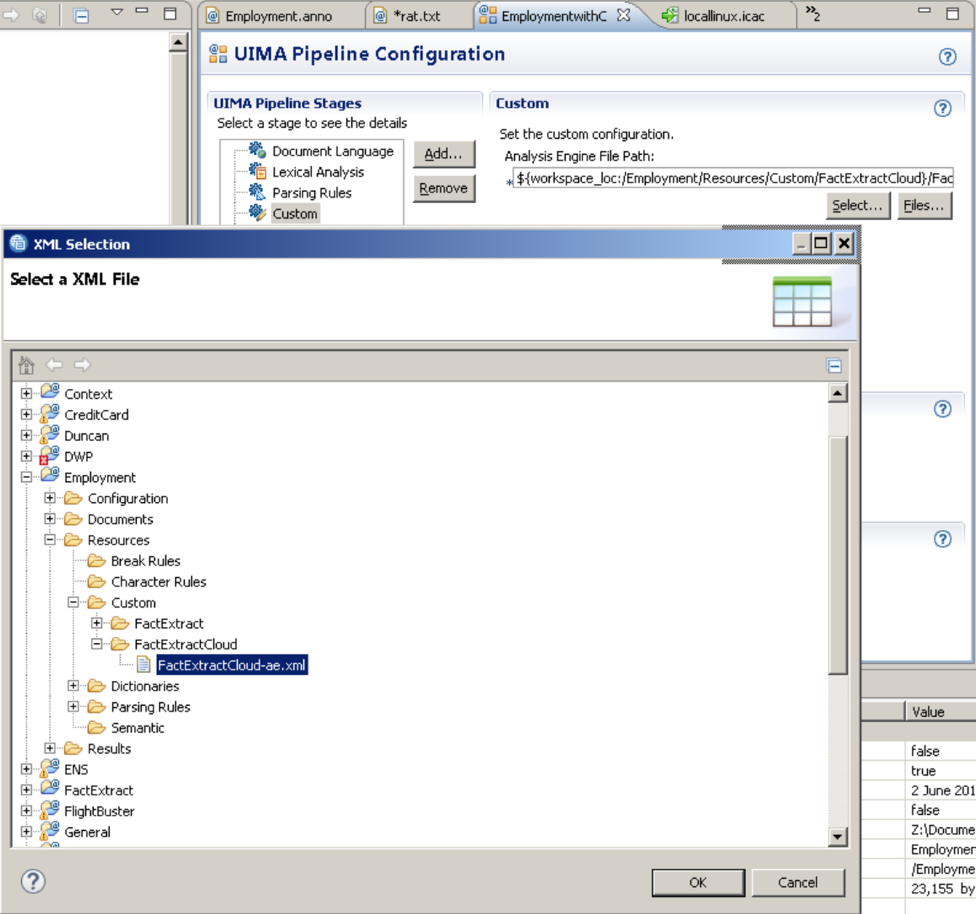
### Create a project and configure your annotations to identify relevant facts.

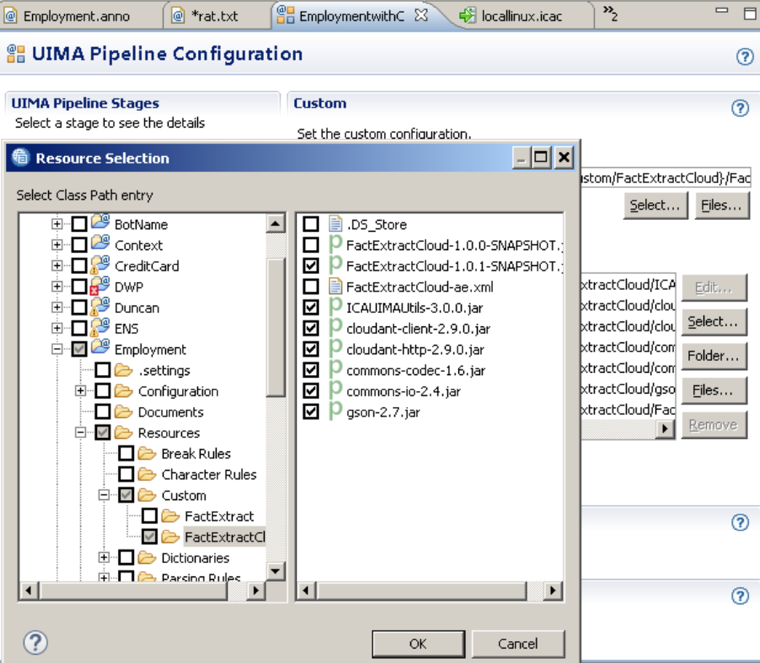
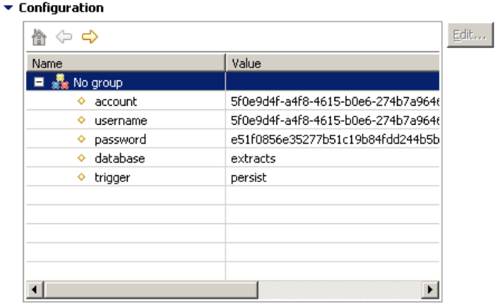
### In this project create a folder to hold the FactExtract resources. This can be anywhere in the workspace but something like **Resources/Custom/FactExtractCloud** would be a good choice.

### Copy the annotation engine configuration file (**FactExtractCloud-ae.xml**) and all the jar files that make up FactExtractCloud into this new folder.

### In the relevant UIMA pipeline configuration file in your project add a custom stage as the penultimate stage of the pipeline. To do this:

* Double click on the relevant pipeline configuration file under your ***Configuration/Annotators*** folder to open and edit it.  
  
* select the existing penultimate stage  
  click the “Add” button  
  
* select “Custom” in the popup window  
  click “OK”  
  
* Click in the text box of the Custom panel under where it says ***Analysis Engine File Path:*** and click the ***Select*** button. Navigate to the folder created in step 2.3.2 and select the ***FactExtractCloud-ae.xml*** annotation engine configuration file.



* Back in the ***Custom*** panel click the small right arrow to open the ***Class Path*** panel and click ***Select***. Again navigate to the new folder and this time click the check boxes to select the jar files.  
  Click ***OK***  
  
* Back in the ***Custom*** panel click the small right arrow to open the ***Configuration*** panel. then click the small plus sign to open the ***No group*** Configuration group.   
     
    
  Set the parameters appropriately according to the guidance in the table below.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Default value** | **Note** |
| **account** | none | \* Cloudant service account from 2.2.4 |
| **username** | none | \* Cloudant service username from 2.2.3 |
| **password** | none | \* Cloudant service password from 2.2.3 |
| **database** | extracts | The Cloudant database to write to. It will be created if it does not exist |
| **trigger** | persist | The name of the feature in annotations that will be used to signal annotations of this type should be saved. |

\* At a minimum these parameters should be changed.

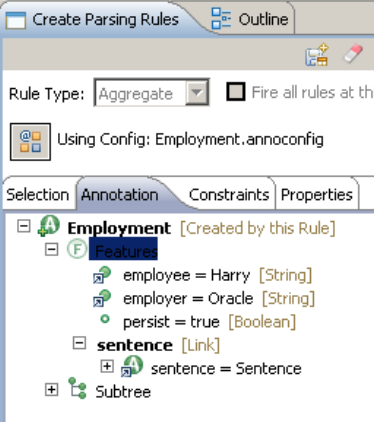
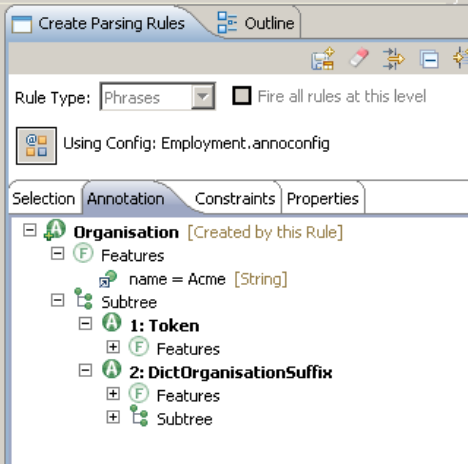
## Content Analytics Server

Once you have defined, developed and tested your project that includes FactExtractCloud, you must export it to the Content Analytics server so that it can be used to analyze documents. This is done by simply deploying your Studio project that contains the FactExtractCloud custom stage to the server just like you would with any other project.

# Configuring the extraction

The FactExtractCloud can be run in pipelines in Content Analytics Studio and in pipelines deployed to the Content Analytics Server. It’s often easiest to first run it in Studio to test the extractions and then run it deployed into your server to extract facts from your corpus of documents in a collection.

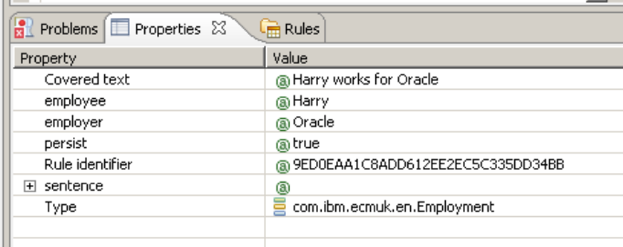
FactExtractCloud uses the presence of a named feature in your annotations to decide if an annotation is going to be persisted or not. The name of this feature is configured in the custom stage (see 2.3), the default name is “persist”. So given the two annotation definitions below the first one will be persisted and the second one won’t.

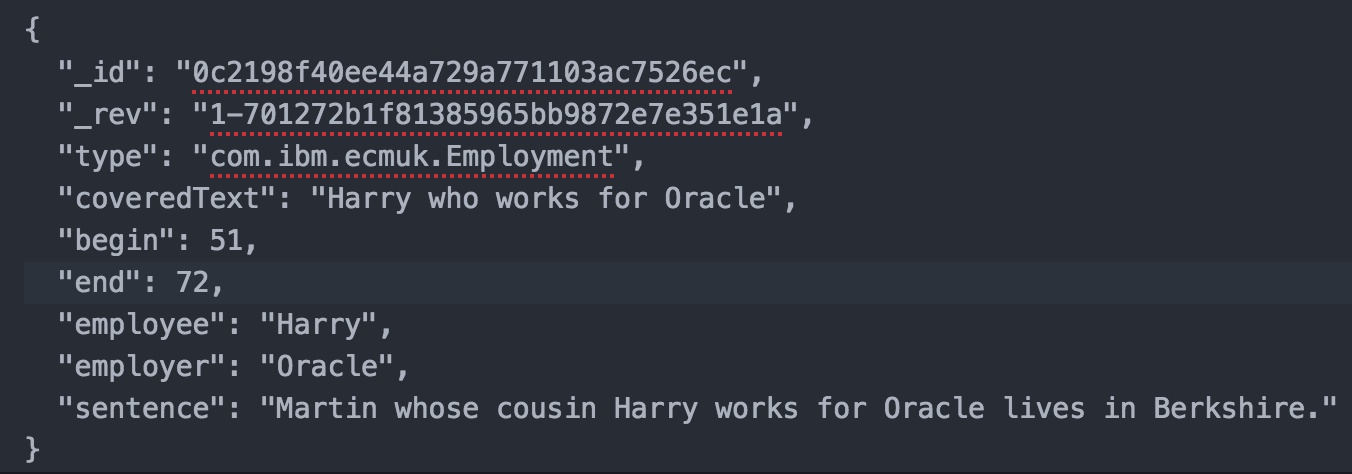
 

The value of the feature is irrelevant, the mere presence is enough to trigger the save.

## What gets persisted

All primitive features with the exception of the rule identifier get persisted, as does the covered text, the character offsets of the annotation and the annotation type name. If you’ve associated either the sentence or paragraph with the annotation (as in the above Employment example) then the text of that sentence or paragraph will get persisted too. Here’s an instance of that example Employment annotation in Studio:

 and here’s the final document in Cloudant:



There is currently no support in the annotator for updates or deletes. So re-running the annotator will create more documents in Cloudant each with a unique **\_id** and **\_rev**.

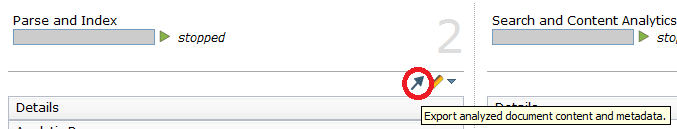
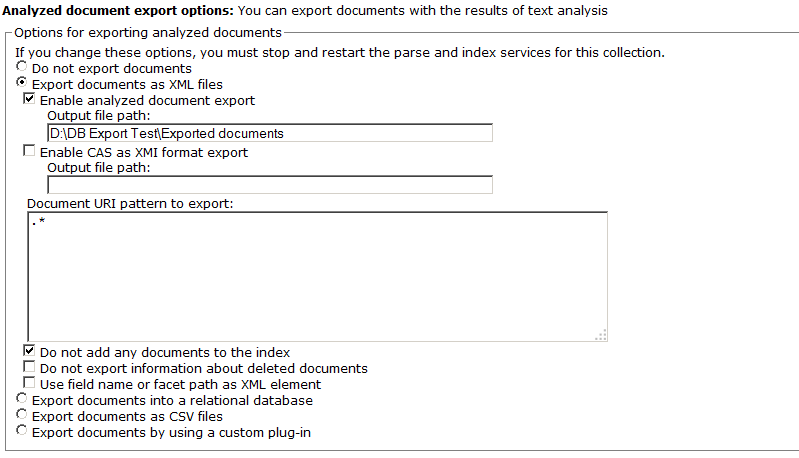
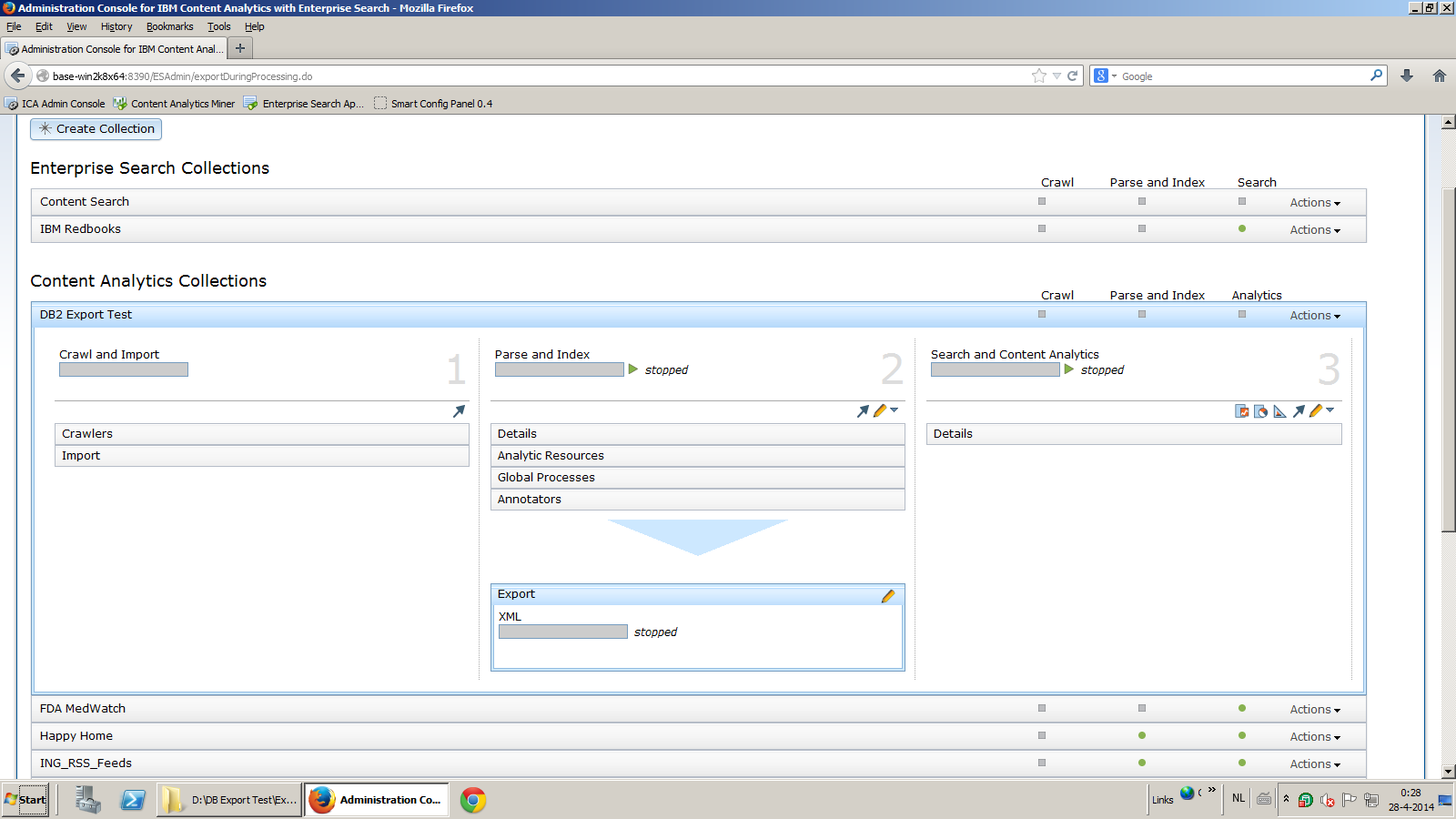
# Running Content Analytics Collections for Information Extraction Only.

This is an optional configuration step.

Having successfully followed the steps described in sections 1 – 3 you will have a collection with a custom analysis step included in its processing pipeline. When the collection crawler is configured and started, the source system will be crawled, and the data analyzed using the custom annotator you designed and that annotator will have been will configured using the FactExtract to automatically write relevant annotations to a database. Finally, the server will also write, at a minimum, the standard lexical analysis annotations to the collection index (things like language, parts of speech, sentence, and paragraph annotations) to be used in either the Text Miner or Search applications. In many cases where we are writing extracted data to a database there is no requirement for these applications, and hence no use for an index. Constructing and maintaining these indexes is time consuming and wasteful of resources. In these cases it is possible to configure the pipeline to run the analysis stages only and not build an index for the collection.

The procedure for doing this is not obvious as the relevant options are included in the collection export configuration screens.

## Turning off document indexing

* From the Content Analytics Administration Console, expand the collection and click on the export icon in Parse and Index section.  
  
* On the Export configuration page, you will want to configure an export for Analyzed Documents, as the documents must be analyzed in order for custom annotator to be applied to the documents. There are several options to choose from, each of which is explained in more detail in the on-line help. If you simply want to avoid the creation of an index then exporting the documents as xml files may be a worthwhile option:  
  
* As the above figure shows, the two important parameters are the file location in which the xml files will be written, and the option "Do not add any documents to the index". This ensures that no index is created for the collection.
* One additional point to make here is that the system must write the documents somewhere. If you were to deselect the "Enable analyzed document export" option, you would get an error.
* Having completed the export configuration, you can simply import data or crawl a data sources. the documents will be written to the target you defined and the annotations to the database as you configured in your annotator.   
  
* You can manually delete the xml files, or write a simple utility to do this automatically.