FactExtract

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



**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Version** | **Status** | **Description** | **Author** |
| 19/03/2014 | 1.0 | Release | Initial | Martin Saunders |
| 24/04/2014 | 1.1 | Release | Bug Fixes & Documentation updates | Martin Saunders, Mark Rice |
| 08/10/2015 | 3.1 | Release | Port to UIMA 2.5, JDK 7, MSQLServer and several new features | Martin Saunders |
| 23/03/2017 | 3.1.3 | Release | Port to UIMA 2.8.1, JDK 8 Minor tweaks to AE parameters Internal refactoring | Martin Saunders |
| 13/09/2017 | 3.2.0 | Release | Added lazymode for easy configuration | Martin Saunders |

**Contents**

1. Purpose 3

2. Installation 5

2.1 Pre-Requisites 5

2.2 Content Analytics Studio 5

2.3 Content Analytics Server 8

3. Configuring the extraction 10

3.1 Configuring with lazymode 10

3.2 Configuring using the configuration database table 11

3.3 Annotation Tables 13

3.4 The Documents Table 13

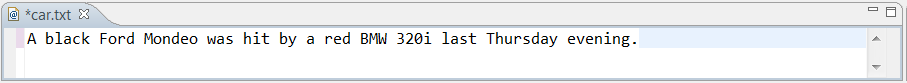
4. Running Content Analytics Collections for Information Extraction Only. 14

4.1 Turning off document indexing 14

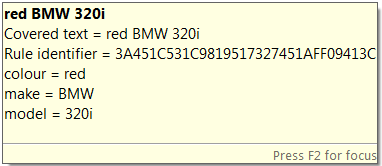
# Purpose

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.

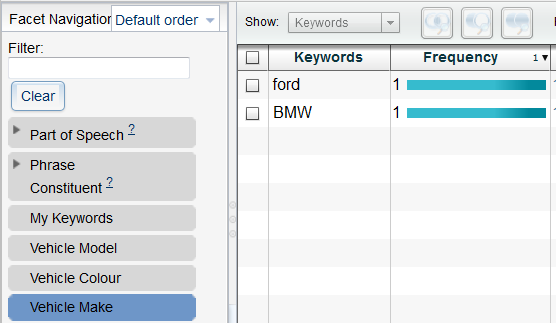
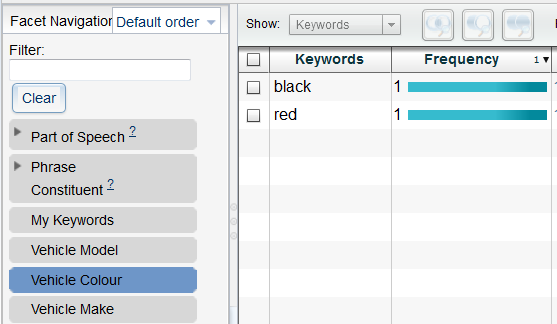
In many instances the entities you want to extract are more complex and have multiples attributes that are also in the text; think of things like a vehicle, these may have a make, model and colour . These can be modeled in Content Analytics Studio using an annotation with multiple features. The instances 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

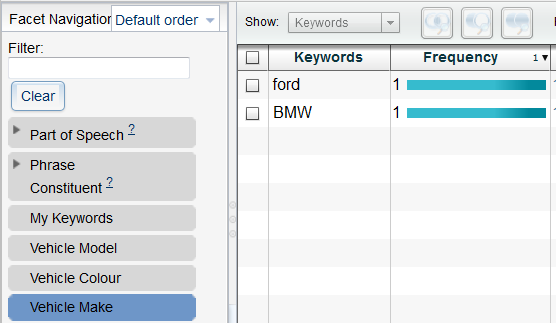
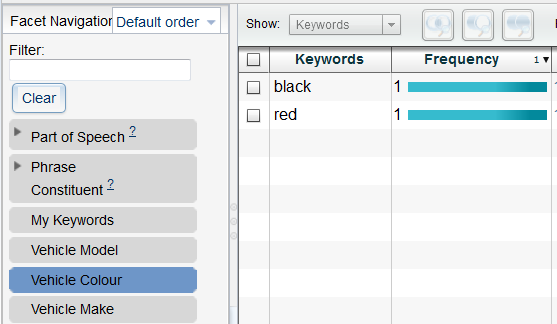


If your interest was in identifying vehicles you might well configure a com.ibm.Vehicle annotation in Content Analytics Studio that would create the following two instance with that text.

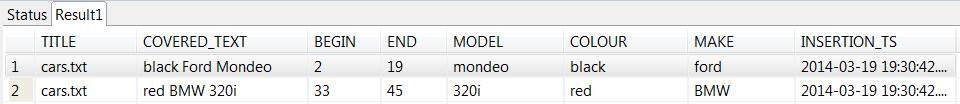


When this is deployed in an Content Analytics collection with a document containing the same text the text miner facets show:





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 FactExtract overcomes this limitation as facts with multiple features can be written as a single record to a database. Using FactExtract the above text could produce the following table.



Creating records like this with multiple columns in an annotation table 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
* Installation of DB2 9.7+ or MSSQLServer
  + A target database in the server.
  + Access via user account with sufficient privileges to create and use schemas and tables within the target database.
* FactExtract files:
  + Core files (included in distribution):
    - FactExtract-ae.xml
    - FactExtract-n-n-n.jar
    - ICAUIMAUtils-n-n-n.jar
  + Db2 support (included in distribution):
    - db2jcc.jar
    - db2jcc\_license\_cu.jar
  + Microsoft SQLServer support (not included in distribution:
    - sqljdbc41.jar

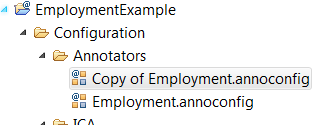
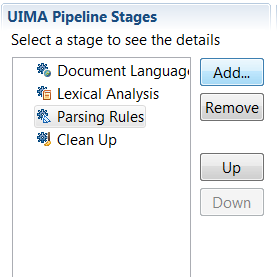
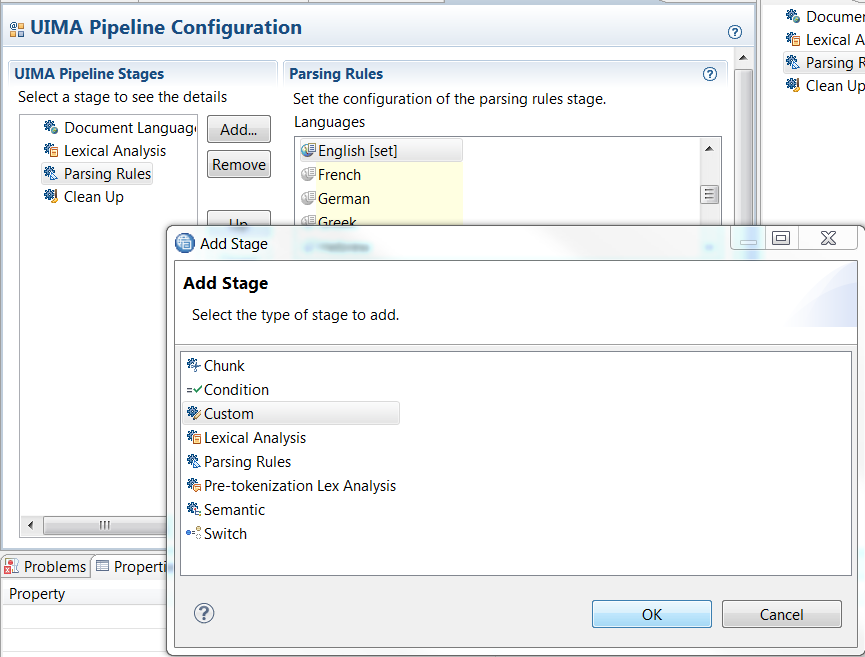
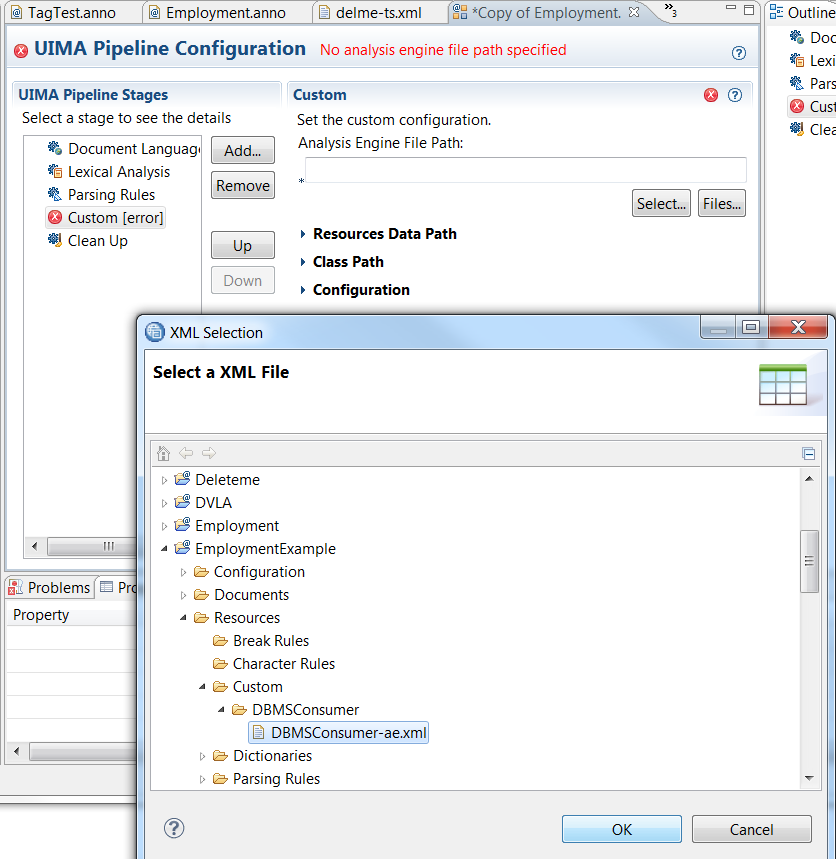
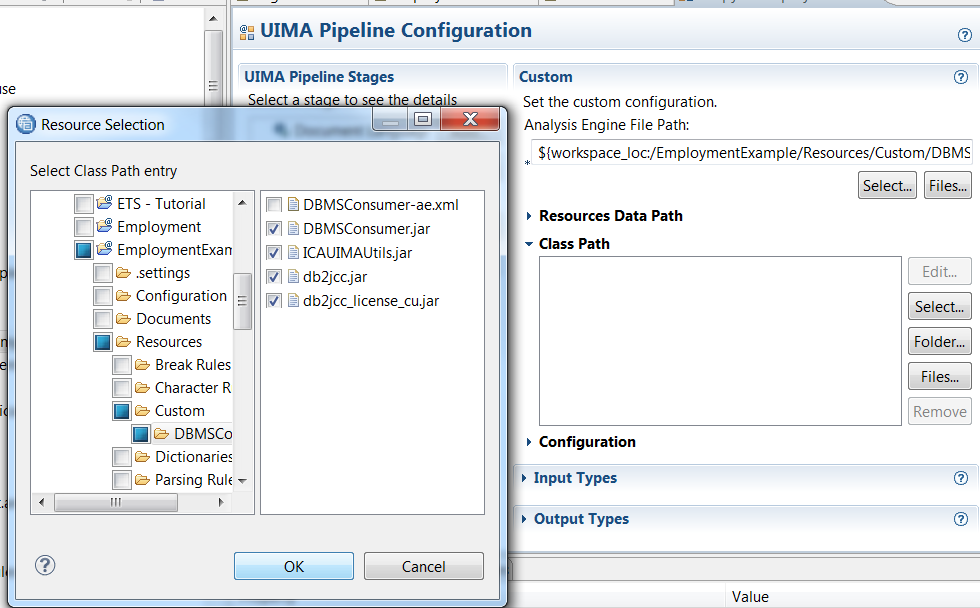
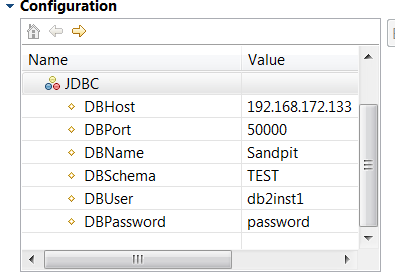
## Content Analytics Studio

### 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/FactExtract** would be a good choice.

### Copy the annotation engine configuration file (**FactExtract-ae.xml**) and all the jar files that make up FactExtract into this new folder. If you’re using a database which doesn’t have the jdbc driver jar files included in the FactExtract distribution then source these separately and copy them in here too.

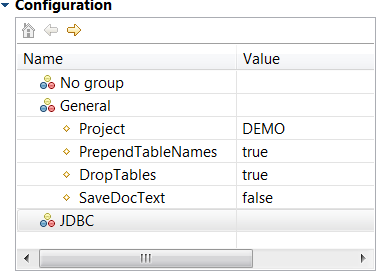
### In the relevant UIMA pipeline configuration file in your project add a custom stage as the penultimate stage of the pipeline. To do this:

* 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 2.2.2 and select the ***FactExtract-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 right arrow to open the ***JDBC*** Configuration group.   
     
    
  Set the JDBC parameters appropriately according to the guidance in the table below.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Default value** | **Note** |
| **DBHost** | 192.168.172.133 | \* hostname or IP address of your DB2 server |
| **DBPort** | 50000 | Port number DB2 server is listening on. |
| **DBName** | Sandpit | \* Database name, this must exist. |
| **DBSchema** | Test | \* The schema to use in the DBName database, if this does not exist it will be created for you the first time FactExtract runs. |
| **DBuser** | db2inst1 | \* User to authenticate with db2 server |
| **DBPassword** | Password | \* Password for authorised user |
| **DB** | DB2 | DB2 or MSSQL |

\* At a minimum these parameters should be changed.

* Close the ***JDBC*** group and click the right arrow to open the ***General*** group,

  
  
Set the parameters according to the guidance in the table on the next page.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Default value** | **Note** |
| **Lazymode** | false | If set to true this switches to lazymode for simple and quick configuration of the extracted annotations. See section 3 Configuring the extraction |
| **Project** | DEMO | \* A unique name for your project. This does not have to be the same as the Content Analytics Studio project name or the Content Analytics collection name though it can be. Any unique reference will suffice. It is limited to ten characters. |
| **PrependTableNames** | true | If set to true (the default) tables created to hold annotation values will have the project name prepended. This allows a single database and schema to be shared across multiple pipelines with FactExtract installed and to avoid any conflicts between tables with the same name in different projects. |
| **DropTables** | true | If set to true (the default) FactExtract will attempt to drop any tables configured to hold annotations when it initialises and then recreate them from the configuration. This allows for easy schema changes in the configuration. In fact if you do change the schema of the tables to which facts are being extracted you must set this to true as the tables need to be dropped and recreated to add or remove any columns. It’s still useful in deployment if you don’t wish to retain the history of extracted facts over multiple pipeline indexing sessions. |
| **SaveDocText** | false | If set to true the full text of all documents analysed is saved in the DOCUMENTS table in the target database. Consequently set to true with care. |
| **KeyField** | default | By default the annotator generate surrogate primary key values for the DOCUMENTS table (and these are foreign keys in the tables for individual annotations. If this parameter is set to a non default value then it is taken as the name of a source metadata field that contains unique key values to be used for the documents instead. This allows preservation of the same keys used in source systems and Content Analytics server index where appropriate. |

\* At a minimum these parameters should be changed.

* Save the UIMA pipeline configuration file

## Content Analytics Server

Once you have defined, developed and tested your project that includes FactExtract, 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 FactExtract custom stage to the server just like you would with any other project.

Should you wish to edit any of the configuration settings in a deployed annotator you can either change them and re-deploy from the Studio project or edit the annotator’s xml configuration file directly on the server. You’ll need to stop and restart the parse and index stage of your collection. To determine the location of the annotator’s configuration file:

* Click the ***System*** tab in the Content Analytics administration console
* Click the sub-tab labeled ***Parse***
* Click ***Configure text analysis engines***
* Click the ***magnifying glass*** icon next to the FactExtract annotator
* Make a note of the directory or folder listed under ***Installed directory:***
* Click ***Return twice***

In your operating system navigate to the directory or folder noted, then to the ***desc*** sub-directory. In there will be a file named custom*N*.xml that contains the configuration values.

# Configuring the extraction

The FactExtract 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.

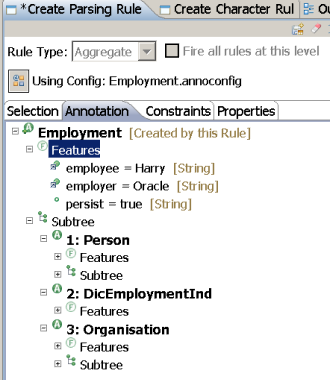
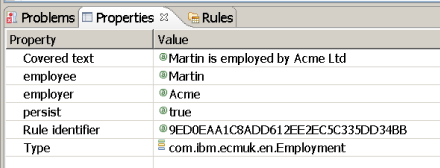
There are two different and mutually exclusive ways to configure the extractions. The quickest and simplest to use is “lazymode”; with lazymode the FactExtract is simply configured by the naming convention you use in annotations. If you add a feature with the correct trigger name to your annotation it will be persisted, if you don’t, it won’t. The alternative method is to explicitly define what you want extracted in the configuration table FactExtract creates in the database. Using the configuration table gives you more control of exactly what gets extracted and how the tables and columns are named. The table below compares the two techniques.

|  |  |  |
| --- | --- | --- |
|  | **Lazymode** | **Configuration Table** |
| **Configuration** | Automatic | Configured manually by entering rows in the configuration table in the target database. |
| **What get extracted?** | Any annotation with a String feature called persist that is set to true. | Defined by the configuration stored in the table. |
| **Table names** | Automatically created with the same name as the annotation. | Full control over each annotation’s table name. |
| **Columns** | A column is automatically added to the table for every feature in the annotation. | Full control over which features are persisted to columns in the target annotation table. |
| **Column names** | Automatically created with the same name as the feature they are holding | Full control over each column’s name |

*Note: Lazymode and configuration table entries are mutually exclusive. If you switch on lazymode anything in the configuration table is ignored.*

## Configuring with lazymode

All annotations that have a type which includes a String feature named “persist” that has a value “true” will be be extracted and persisted in the database. See this annotation rule and and properties of an instance of it as an example.

This annotation type is **com.ibm.ecmuk.Employment** and the presence of a feature called **persist** will result in a database table being created named **Employment**. The annotation has features of **employee** and **employer** and so the table will have columns with those names and values too. The final table schema will look like this:

|  |  |
| --- | --- |
| **EMPLOYMENT** holds extracted **com.ibm.ecmuk.Employment** annotations | |
| **DOC\_ID** | Foreign key to the **DOCUMENTS** table the holds information about which document these facts were extracted from. |
| **COVERED\_TEXT** | The annotated text of the **com.ibm.Vehicle** annotations in the document |
| **BEGIN\_OFFSET** | The character offset in the document where the annotated text begins. |
| **END\_OFFSET** | The character offset in the document where the annotated text ends. |
| **EMPLOYEE** | The value of the **employee** feature from the **com.ibm.ecmuk.Employment** annotations. |
| **EMPLOYER** | The value of the **employer** feature from the **com.ibm.ecmuk.Employment** annotations. |
| **INSERTION\_TS** | The timestamp when the facts were extracted. |

Note that the feature persist=true must be set on all instances of annotations you want extracted. This means that if you have two rules creating the same annotation type and one has the persist feature set to true and one doesn’t then only annotations triggered by the first rule will be persisted.

## Configuring using the configuration database table

The first time the FactExtract is run it will connect to the database configured when you installed it in the pipeline, then create the configured schema (if it doesn’t exist), and then finally create the minimum necessary tables in that schema. No extraction is ever done this first time because nothing is configured to be extracted, and the tables that would hold the configuration don’t exist. Consequently, you can run the pipeline the first time against any document in Studio to just get the schema and tables created. You can then enter values into the configuration table. The next time the pipeline with the FactExtract is run it will read the values from the configuration table and then assuming these are valid will extract these annotations from all documents it processes and write them to the requested target tables.

The configuration table created in the target schema is named CONFIGSCHEMA and has the following structure:

|  |  |  |
| --- | --- | --- |
| **CONFIGSCHEMA** | | |
| **PROJECT** | mandatory | The project name for this entry. This is matched against the project name specified in the configuration of the FactExtract into the UIMA pipeline in step 2.2.4. It allows for multiple deployments of FactExtract in multiple pipelines to share this single configuration table. |
| **TABLENAME** | mandatory | The name of the table to be created to hold annotation values |
| **ANNOTYPE** | mandatory | The fully qualified annotation name to extract, this is case sensitive. |
| **COLUMNNAME** | optional | The name of the column to be created in the target TABLENAME to hold the FEATURE value |
| **FEATURE** | optional | The name of the feature in the ANNOTYPE feature structure to store in COLUMNNAME |

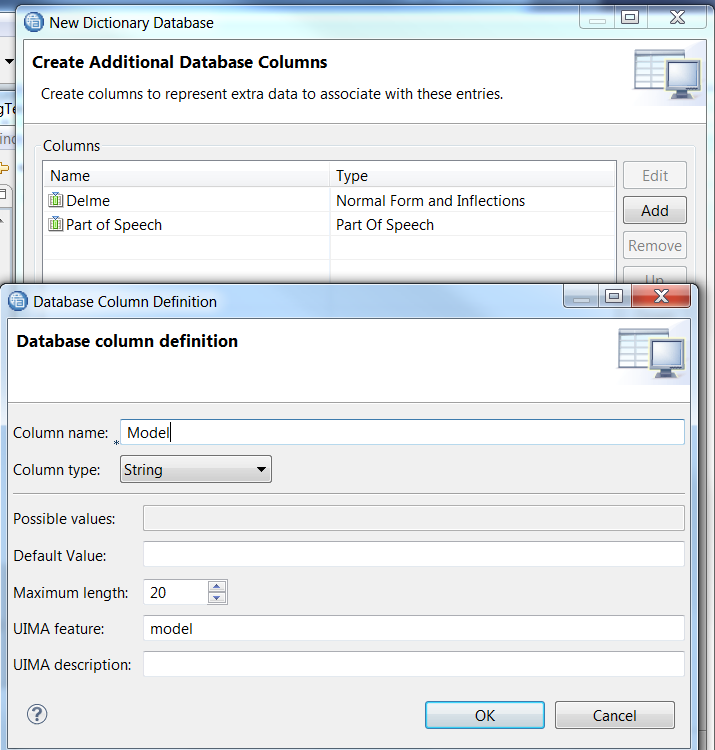
For example, in section 1 we had an annotation ***com.ibm.Vehicle*** that had three features: colour, make and model.  
  
**com.ibm.Vehicle**  


If we wanted to extract these vehicle facts into a table called ***CAR***  and were only interested in the make and model and wanted these in columns named ***BRAND*** and ***MODEL*** respectively we would achieve this with the following entries in ***CONFIGSCHEMA***.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PROJECT** | **TABLENAME** | **ANNOTYPE** | **COLUMNNAME** | **FEATURE** |
| Demo | CAR | com.ibm.Vehicle | BRAND | make |
| Demo | CAR | com.ibm.Vehicle | MODEL | model |

Where you just want to extract the text of annotation and it doesn’t have any features or you don’t want them just leave the COLUMNNAME and FEATURE columns empty.

***A note on dictionaries***

When you create dictionaries in Content Analytics Studio that have additional columns the wizard asks you for the column name and automatically generates the internal UIMA feature name from the label you specify (though you can override this). The UIMA feature names created always starts with a lower case letter irrespective of the capitilisation used in the column name label. It is this UIMA feature name that must be specified in the FEATURE column not the Column name that will appear in the GUI.   


This also applies to the built-in Lemma feature in dictionary types. In the Studio GUI when text is annotated from dictionary entries these show as “**Lemma**” with a capital **L**, but the underlying UIMA features are actually named “**lemma**” with a lower case **l**. If you want to extract lemma feature values from dictionary types you must specify lowercase **lemma** in the FEATURE column.

## Annotation Tables

Individual tables are created for each annotation extracted as specified in the **CONFIGSCHEMA** table. These new tables always have five columns created (DOC\_ID, COVERED\_TEXT, BEGIN\_OFFSET, END\_OFFSET and INSERTION\_TS), additional columns are created according to the specification. The configuration example shown in section 3.2 above would result in a table named **DEMO\_CAR** being created (assuming PrePendTablenames is set to true) with the following columns to hold the extracted facts.

|  |  |
| --- | --- |
| **DEMO\_CAR** holds extracted **com.ibm.Vehicle** annotations | |
| **DOC\_ID** | Foreign key to the **DOCUMENTS** table the holds information about which document these facts were extracted from. |
| **COVERED\_TEXT** | The annotated text of the **com.ibm.Vehicle** annotations in the document |
| **BEGIN\_OFFSET** | The character offset in the document where the annotated text begins. |
| **END\_OFFSET** | The character offset in the document where the annotated text ends. |
| **BRAND** | The value of the **make** feature from the **com.ibm.Vehicle** annotations. |
| **MODEL** | The value of the **model** feature from the **com.ibm.Vehicle** annotations. |
| **INSERTION\_TS** | The timestamp when the facts were extracted. |

## The Documents Table

When the initial schema is created a **DOCUMENTS** table is created too. This holds references to the documents that have been processed. The table has the following structure.

|  |  |
| --- | --- |
| **DOCUMENTS** | |
| **DOC\_ID** | Primary key for each document. Either an annotator generated surrogate key or the value of the document's Metadata field specified with the KeyField configuration property. |
| **URL** | The source url of the document being processed. In the case of documents in Content Analytics server this will be taken from the crawler. In the case of documents in Studio this will be set to “unknown” |
| **TITLE** | The title of the document being processed. In the case of documents in Content Analytics server this will be taken from the crawler. In the case of documents in Studio this will be set to “unknown” |
| **DATASOURCE\_NAME** | The name given to the crawler in Content Analytics server that retrieved this document. In the case of documents in Studio this will be set to “unknown” |
| **DOC\_DATE** | Not used. |
| **DOCTEXT** | If the SaveDocText configuration parameter is set to true in the FactExtract custom stage in the UIMA Pipeline configuration file then this column will hold the full text of the document being processed, otherwise it will be null. |
| **UPDATE\_TS** | The timestamp when the document was processed. If a document is reprocessed (re-crawled or re-indexed) then this timestamp is updated. |

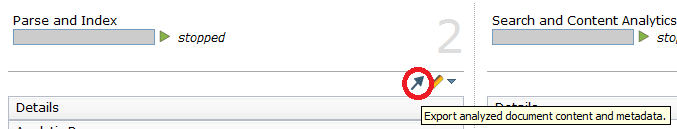
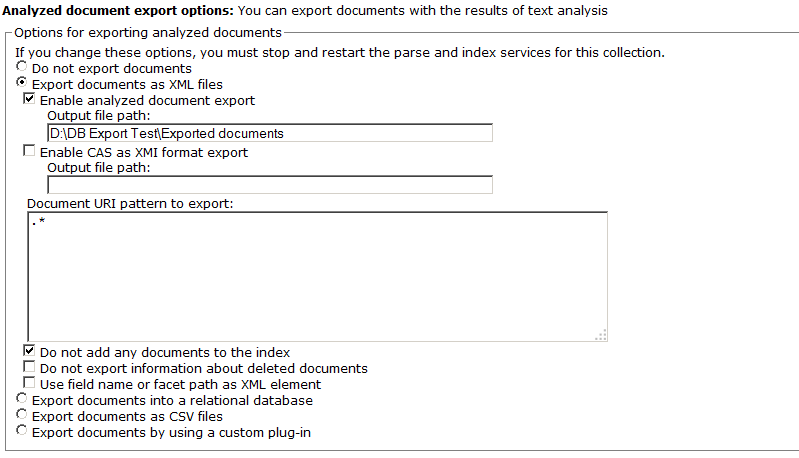
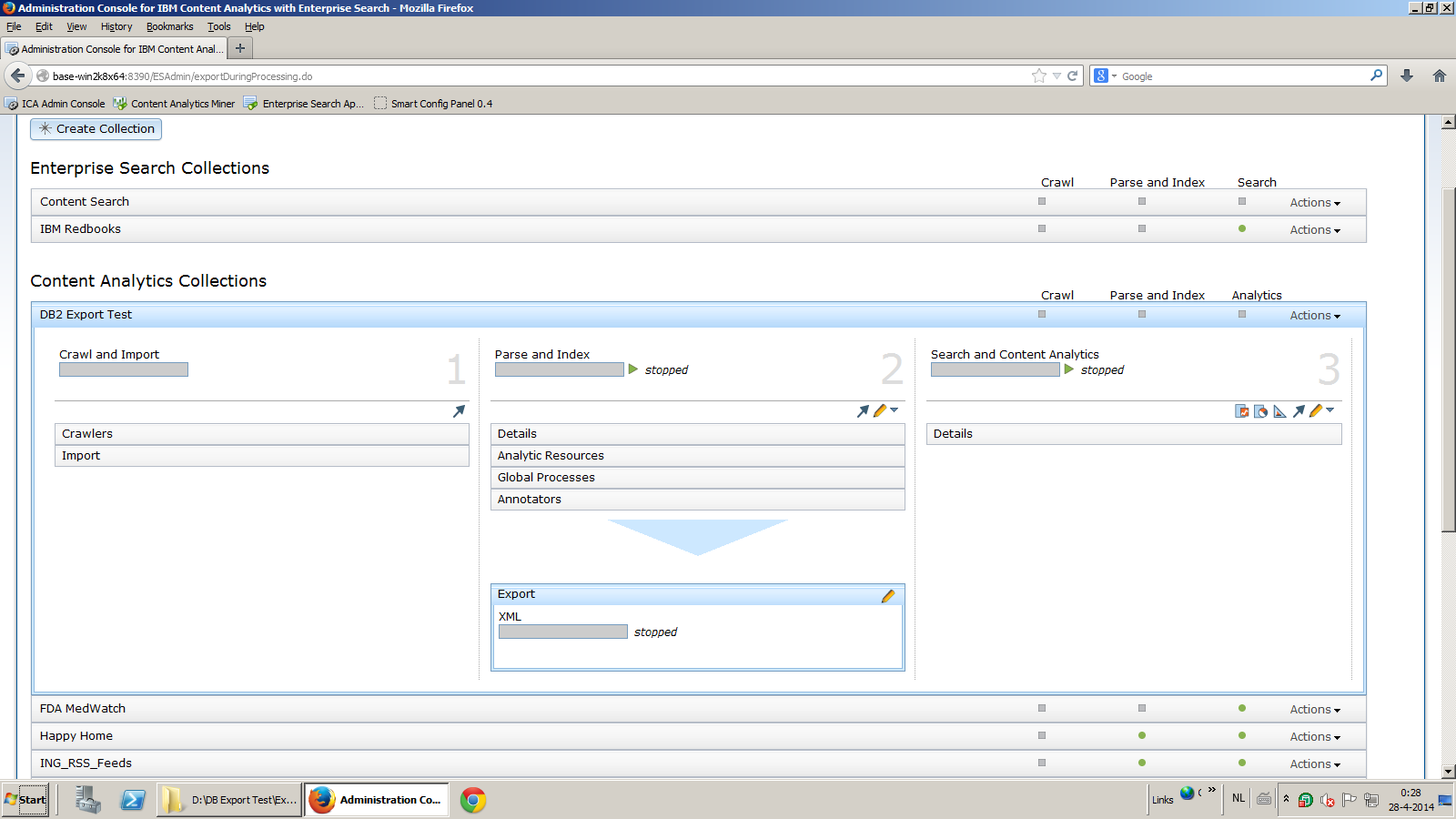
# Running Content Analytics Collections for Information Extraction Only.

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.