# Indexer Configuration

Creating, modifying or deleting resources in the repository generates JMS events. The indexer listens to those events, and retrieves the RDF from the repository. The indexer can be configured to process the event in various ways, such as copying the resource RDF to a triplestore or indexing in Solr.

One of the major goals of this event-based indexing approach is to reduce the impact of indexing on core repository functionality. The repository just creates a JMS event (containing only the resource identifier and the event type, which are already in memory), and does not need to do any extra work for indexing before moving on to its next task. When repository updates happen at a faster rate than the indexer can match, JMS events can wait in the queue until the indexer catches up, and the updates can continue without waiting. When processing large batches of updates, you can even disable the indexer.

The indexer can have any number of workers configured to process the events. So the main indexer process retrieves the resource RDF from the repository, and that content can be reused by multiple workers. If you want to process the events in several ways (triplestore, Solr, archive to disk, update remote repository, etc.), this limits the number of times the metadata has to be retrieved from the repository to once each time the resource is updated.

## Indexer Modules

Several different indexer modules exist for syncing with different systems:

- ElasticIndexer - syncing with Elasticsearch
- FileSerializer - saves Solr document format to disk
- JcrXmlPersistenceIndexer - saves JCR/XML to disk
- RdfPersistenceIndexer - saves RDF to disk
- SolrIndexer - syncs with Solr
- SparqlIndexer - syncs with triplestores using SPARQL Update

## Configuration

The indexer is configured using Spring. Here is a sample configuration fragment showing three workers (saving RDF to disk, persisting jcr/xml, and syncing to a Jena Fuseki triplestore) and the framework for listening to events and connecting them with the workers:

```
  <!-- Worker #1: Copy resource RDF to a Fuseki triplestore using SPARQL Update -->
  <bean id="sparqlUpdate" class="org.fcrepo.indexer.SparqlIndexer">
    <!-- base URL for triplestore subjects, PID will be appended -->
    <property name="prefix" value="http://localhost:${test.port:8080}/rest/objects/"/>
    <property name="queryBase" value="http://localhost:3030/test/query"/>
    <property name="updateBase" value="http://localhost:3030/test/update"/>
    <property name="formUpdates">
      <value type="java.lang.Boolean">false</value>
    </property>
  </bean>

  <!-- Worker #2: Save resource RDF to timestamped files on disk -->
  <bean id="fileSerializer" class="org.fcrepo.indexer.FileSerializer">
    <property name="path" value="./target/test-classes/fileSerializer/"/>
  </bean>

  <!-- jcr/xml persistence Indexer -->
  <bean id="jcrXmlPersist" class="org.fcrepo.indexer.persistence.JcrXmlPersistenceIndexer">
    <constructor-arg value="${fcrepo.jcrxml.storage:fcrepo4-jcrxml}" />
  </bean>

  <!-- Main indexer class that processes events, gets RDF from the repository and calls the workers -->
  <bean id="indexerGroup" class="org.fcrepo.indexer.IndexerGroup">
    <constructor-arg name="repositoryURL" value="http://${fcrepo.host:localhost}:${fcrepo.port:8080}${fcrepo.
context:/}rest" />
    <constructor-arg name="indexers">
      <set>
        <ref bean="jcrXmlPersist"/>
        <ref bean="fileSerializer"/>
        <ref bean="sparqlUpdate"/>
      </set>
    </constructor-arg>
    <!-- If your Fedora instance requires authentication, enter the credentials here. Leave blank if your repo
is open. -->
    <constructor-arg name="fedoraUsername" value="${fcrepo.username:}" />
    <constructor-arg name="fedoraPassword" value="${fcrepo.password:}" />
  </bean>

  <!-- ActiveMQ queue to listen for events -->
  <bean id="destination" class="org.apache.activemq.command.ActiveMQTopic">
    <constructor-arg value="fedora" />
  </bean>

  <!-- Message listener container to connect the JMS queue to the indexer -->
  <bean id="jmsContainer" class="org.springframework.jms.listener.DefaultMessageListenerContainer">
    <property name="connectionFactory" ref="connectionFactory"/>
    <property name="destination" ref="destination"/>
    <property name="messageListener" ref="indexerGroup" />
    <property name="sessionTransacted" value="true"/>
  </bean>
```

To use another triplestore, change the SparqlIndexer bean configuration.  Here is the bean configuration to use with Sesame running on port 8081:

```
  <!-- Worker #1: Copy resource RDF to a Sesame triplestore using SPARQL Update -->
  <bean id="sparqlUpdate" class="org.fcrepo.indexer.SparqlIndexer">
    <!-- base URL for triplestore subjects, PID will be appended -->
    <property name="prefix" value="http://localhost:${test.port:8080}/rest/objects/"/>
    <property name="queryBase" value="http://localhost:8081/openrdf-sesame/repositories/test"/>
    <property name="updateBase" value="http://localhost:8081/openrdf-sesame/repositories/test/statements"/>
    <property name="formUpdates">
      <value type="java.lang.Boolean">true</value>
    </property>
  </bean>
```

## Extending the Indexer

To implement a new kind of indexer:

1. Implement the indexing functionality using the org.fcrepo.indexer.Indexer interface, which consists of only two methods (one to handle new /updated records, and another to handle deleted records). Any configuration required should be done using Java bean setter methods.
2. Update the Spring configuration to add a bean referencing the new class and providing the configuration properties needed.
3. Add the bean to the list of workers invoked by the indexer.

## Trying Out the Indexer

To get hands-on experience with the indexer and see updates synced with an external triplestore, you need three components. Each component will potentially run in its own application container. The three components are:

1. Triplestore (Fuseki or Sesame)
2. Fedora 4 Repository
3. JMS event listener/indexer

The triplestore and Fedora4 do not need to be aware of each other or of the JMS listener. However, the event-listener needs to know the web-endpoints of both the triplestore and Fedora 4. It is therefore important that you start the three components on different ports.

Instructions on how to start up and configure the three components follows:

#### 1. Triplestore

- The easiest to setup is Jena Fuseki (Fuseki setup instructions).
- Alternatively, you can setup Sesame (Sesame setup instructions).

#### 2. Fedora Repository

You can deploy Fedora4 either by downloading the latest war file and dropping it into an application container (e.g. Tomcat7). Or you can clone the Git fcrepo4 project and run the fcrepo-webapp directly within the code base.

See the following pages for details on either approach:

- Deploying Fedora 4 Complete Guide
- First Steps

#### 3. JMS Event Indexer

You can deploy the JMS event listener/indexer by downloading the latest war file and dropping it into an application container (e.g. Tomcat 7). Or you can clone the fcrepo-message-consumer project and run the fcrepo-message-consumer-pluggable directly within the code base. Building the project from source will likely make it easier to configure the JMS event listener/indexer.

You can specify the connection to either Fuseki or Sesame in the following configuration file.

- By default, Fuseki is expected
- To connect to Sesame instead, comment out the "queryBase", "updateBase", and "formUpdates" XML elements associated with Fuseki, and uncomment the corresponding Sesame XML elements in the configuration file mentioned above.

To configure the JMS indexer to connect to the Fedora Repository, you can set the following system variables

```
-Dfcrepo.host=<defaults.to.localhost>
-Dfcrepo.port=<defaults.to.8080>
```

To configure the JMS indexer to connect to the triplestore, you can set the following system variables

```
-Dfuseki.host=<defaults.to.localhost>
-Dfuseki.port=<defaults.to.3030>
```

... or if you are using Sesame:

```
-Dsesame.host=<defaults.to.localhost>
-Dsesame.port=<defaults.to.8081>
```

Finally, you will potentially need to set the output directory for the FileSerializer (which is a testing class for showing what is being indexed)

```
-Dfile.serializer.dir=<defaults.to.webcontainer.target>
```

Below is an example of how to download, build, and start the JMS indexer.

```
$ git clone https://github.com/fcrepo4/fcrepo-message-consumer.git

$ cd fcrepo-message-consumer
$ mvn install
$ cd fcrepo-message-consumer
$ mvn -Dfcrepo.host=localhost -Dfcrepo.port=8080 -Dfuseki.host=localhost -Dfuseki.port=3030 -Djetty.port=8082
jetty:run
```

If the Fedora Repository is be running at http://localhost:8080/rest/ – you can create, update and delete resources using your browser, or using the REST API (see SPARQL Recipes ). Each event will trigger the indexer and be synced to Fuseki (or Sesame), which you can access at http://localhost:3030/ (if you have Fuseki running on its default port).

- **Reindexing**

If you have a repository with existing content that you want to index, or have changed your indexing logic and want to reindex content, you can use the reindex REST API call in the indexer webapp.

To reindex the resource `http://localhost:8080/rest/objects/` and all of its children:

```
$ curl -X POST -d baseURI=http://localhost:8080/rest/objects/ http://localhost:8082/reindex
```

To reindex just the resource `http://localhost:8080/rest/objects/foo/`, but not recursively reindex its children, add the `recursive=false` parameter:

```
$ curl -X POST -d baseURI=http://localhost:8080/rest/objects/ -d recursive=false http://localhost:8082/reindex
```

- **Indexing Multiple Repositories to a single Triplestore**

In some situations it is desirable to have multiple Fedora repositories all feeding into a single external triplestore. In order to accomplish this, we need to install and setup the three components (Triplestore, Fedora 4 Repository and JMS event listener/indexer) as follows:

- Follow the instructions above to install the triplestore (Fuseki or Sesame) in one machine and start it.
- Follow the instructions above to install two or more Fedora 4 Repositories in different machines and start them.
- Install JMS event listener/indexer (https://github.com/fcrepo4/fcrepo-message-consumer) for each Fedora 4 repository installation and start the indexer with the following command:

  ```
  $ mvn -D jetty.port=9999 -Dfuseki.host=<triplestore.host.name> -Dfcrepo.host=<repository.host.name>
  jetty:run
  ```

- Notes
  - To make a resource indexable in the triplestore, the resource needs to include Indexable mixin type: http://fedora.info/definitions/v4 /indexing#Indexable, which can be inserted through a SPARQL insert:

    ```
    INSERT {<> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://fedora.info/definitions/v4
    /indexing#Indexable> }.
    ```

  - Start the triplestore first. If the triplestore is restarted, then the JMS event listener/indexer needs to be restarted, too.