MongoDB Settings

Beginning with release 3.0, Aspire uses an external MongoDB instance for connectors built with the new NoSQL Connector Framework. The database is used to keep crawl metadata and allow processing and scanning to be distributed. All MongoDB configuration is done in the **settings.xml** file.

Basic Example:

```
<!-- noSql database provider for the 3.1 connector framework -->
<noSQLConnectionProvider connectionsPerHost="10" sslEnabled="false" sslInvalidHostNameAllowed="false">
  <implementation>com.searchtechnologies.aspire:aspire-mongodb-provider</implementation>
  <dropOnClear>false</dropOnClear>
  <servers>mongodb-host:27017</servers>
</noSQLConnecitonProvider>
```

Attribute	Value	Description
`connectionsPerHost`	integer	Number of simultaneous connections to MongoDB, it will add up to the two (2) mandatory connections for Aspire and Console
`dropOnClear`	true /false	Select if Aspire should "drop" MongoDB collections instead of "clear". Dropping collections will increase time performance significantly if amount of documents are over one million (1M). Otherwise if less "clear" is recommended (option in false)

Aspire will create one MongoDB database for each content source configured. When the content source is deleted, the database will be dropped. The database name will the taken from the normalised value of the content source name. Starting in Aspire 3.1, the database names will be prefixed with "aspire-" to avoid possible conflicts of name. If you wish to change the prefix, add a "namespace" to the configuration:

```
<!-- noSql database provider for the 3.1 connector framework -->
<noSQLConnectionProvider connectionsPerHost="10" sslEnabled="false" sslInvalidHostNameAllowed="false">
  <namespace>myNamespace</namespace>
  <implementation>com.searchtechnologies.aspire:aspire-mongodb-provider</implementation>
  <dropOnClear>false</dropOnClear>
  <servers>mongodb-host:27017</servers>
</noSQLConnecitonProvider>
```

Connect to a Multi-node MongoDB Installation

To connect to a multi-node MongoDB installation, you just need to provide a comma-separated list of hostname:port of the MongoDB nodes in the cluster.

Example:

```
<!-- noSql database provider for the 3.1 connector framework -->
<noSQLConnectionProvider connectionsPerHost="10" sslEnabled="false" sslInvalidHostNameAllowed="false">
  <implementation>com.searchtechnologies.aspire:aspire-mongodb-provider</implementation>
  <dropOnClear>false</dropOnClear>
  <servers>mongodb-host1:27017,mongodb-host2:27017,mongodb-host3:27017,mongodb-host4:27017</servers>
</noSQLConnecitonProvider>
```

Using TLS/SSL

If you need to connect to a MongoDB configured to Use TLS/SSL you need to set the following attributes into the **noSQLConnectionProvider** tag:

Attribute	Value	Description
sslEnabled	true	Enables the ssl on the Aspire MongoDB client

sslInvalidHostNameAllowed	true/false	Disables the hostname verification from the SSL validation

For using TLS/SSL you need to make sure the Certificate Authority (CA) that signed the server certificate that MongoDB is using (server.pem) is a trusted certificate, or that its trust chain can lead to one. If you are using a self signed Certificate Authority to sign your server certificate, you need to add it into the java truststore.

To use a java truststore that you need the Certificate Authority certificate (.cert) and import it using the following command

```
$ keytool -import -trustcacerts -alias slc -file <your-CA-certificate.cert> -keystore truststore.jks -storepass
<your-truststore-password> -noprompt
```

After importing it into a truststore you need to add it into the Aspire startup script, read Crawling via HTTPs for more instructions on how to add the truststore into the startup script.

X.509 Authentication

Aspire 3.1 only supports authenticating to MongoDB using X.509.

The X.509 mechanism authenticates a user whose name is derived from the distinguished subject name of the X.509 certificate presented by the driver during SSL negotiation. This authentication method requires the use of SSL connections with certificate validation.

To configure it, add the following to your **settings.xml** file:

```
  <!-- noSql database provider for the 3.1 connector framework -->
  <noSQLConnectionProvider connectionsPerHost="10" sslEnabled="true" sslInvalidHostNameAllowed="false">
    <implementation>com.searchtechnologies.aspire:aspire-mongodb-provider</implementation>
    <dropOnClear>false</dropOnClear>
    <servers>mongodb-host:27017</servers>
    <x509username>CN=user,OU=OrgUnit,O=myOrg</x509username>
  </noSQLConnecitonProvider>
```

If you don't know what to use into the <x509username> field execute the following command using the x509 client certificate:

```
$ openssl x509 -in client.pem -inform PEM -subject -nameopt RFC2253 | grep subject
subject= CN=aaguilar-lptp.search.local,OU=demouser,O=Search Technologies S.A.,ST=Limon,C=CR
```

For using x509 authentication you need to import the client x509 certificate into a java keystore for Aspire to be able to present it to the server for authentication. (The truststore should already be set in the startup script for self signed certificates)

For importing the x509 certificate (client.pem) into a java keystore you need to execute the following commands:

```
$ openssl pkcs12 -export -out client.pkcs12 -in client.pem
Enter Export Password: <your-password-here>

$ keytool -importkeystore -srckeystore client.pkcs12 -srcstoretype PKCS12 -destkeystore client.jks -
deststoretype JKS
Enter destination keystore password:
Re-enter new password: <your-password-here>
Enter source keystore password: <your-password-here>
Entry for alias 1 successfully imported.
Import command completed:  1 entries successfully imported, 0 entries failed or cancelled
```

After importing the client's certificate into a java keystore, you need to include it into the Aspire startup script (aspire.bat) :

```
-Djavax.net.ssl.keyStore=C:\pathToKeyStore\client.jks
-Djavax.net.ssl.keyStorePassword=password
```

Encrypt sensitive fields in MongoDB

If you want to be extra safe and encrypt the URLs, IDs, or any other metadata stored in MongoDB, you can do by specifying the name of the fields to encrypt:

```
<!-- noSql database provider for the 3.1 connector framework -->
<noSQLConnectionProvider connectionsPerHost="10" sslEnabled="false" sslInvalidHostNameAllowed="false">
  <implementation>com.searchtechnologies.aspire:aspire-mongodb-provider</implementation>
  <dropOnClear>false</dropOnClear>
  <servers>mongodb-host:27017</servers>
  <encryptFields>
    <field>_id</field> <!-- Encrypts all the IDs -->
    <field>url</field> <!-- Encrypts the url fields -->
    <field>fetchUrl</field>
    <field>parentId</field>
  </encryptFields>
</noSQLConnecitonProvider>
```