Contents

[Introduction 1](#_Toc478055966)

[Advantages 1](#_Toc478055967)

[Why should use MongoDB 1](#_Toc478055968)

[Where should use MongoDB? 2](#_Toc478055969)

[How to Set-up and Configure Mi4T 2](#_Toc478055970)

[SETUP 2](#_Toc478055971)

[CMS Setup 2](#_Toc478055972)

[MongoDB Setup 3](#_Toc478055973)

[Content Delivery Setup 3](#_Toc478055974)

[MongoDB Index Service 3](#_Toc478055975)

[MongoDB Search Service 3](#_Toc478055976)

[To test the index and search services 4](#_Toc478055977)

[Index service 4](#_Toc478055978)

[Search Service 4](#_Toc478055979)

# Introduction

MongoDB Integration 4 Tridion-Mi4T is intent to provide the Tridion integration with MongoDB

Let's see some of the features advantages of using **MongoDB**

## Advantages

1. Schema less: MongoDB is document database in which one collection holds different different documents. Number of fields, content and size of the document can be differ from one document to another.
2. Structure of a single object is clear
3. Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language

## Why should use MongoDB

1. Document Oriented Storage: Data is stored in the form of JSON style documents
2. Index on any attribute
3. Replication & High Availability
4. Rich Queries

## Where should use MongoDB?

1. Big Data
2. Content Management and Delivery
3. Mobile and Social Infrastructure
4. User Data Management

**How to Set-up and Configure Mi4T**

We have five different modules in the Mi4T

1. **MongoDB -**Setup and configuration
2. **Template Bulding Block: -**C# TBB is used to get the component DCP in XMLformat after some changes.
3. **Custom Storage Extension** – A JPAComponentPresentationDAO based custom storage extension to manipulate the dynamic component presentations
4. **MongoDBIndexService-** WCFRestFul Service which get invoked by custom storage extension and take DCP as input
5. **MongoDBSearchService**- WCFRestFul service to get the data from MongoDB and take input query in JSON format.

# SETUP

Below are various setup steps

## CMS Setup

* + Copy and paste the templating building block (TBB) to a location on your Tridion CM Server
  + Upload **MI4TIndexing.Templating.dll** TBB to Tridion CMS
  + Create a Component Template with following attributes
    - Output Format – XML Fragment
    - Add **GetComponentAsXML TBB, Publish Binaries in Package, Link Resolver and Clean-up Template**

## MongoDB Setup

* + You Can download the MongoDB [link](https://www.mongodb.com/download-center)
  + To start MongoDB
    - Create folder C:/servers and copy the bin from MongoDB installed Path
    - Create data/db folders in C:/servers
    - **Run**>> mongod --dbpath c:\servers\data --port 27017
    - [click here now find out more on MongoDB Installation and setup](http://hemkant-webapps.blogspot.in/2016/05/mongodb-crud-introduction.html)

## Content Delivery Setup

* + Open the cd\_storage\_config.xml Storage Configuration file from the /bin/config folder and add following node under the Storages section:
    - <StorageBindings><Bundle src="CustomStorageDAOBundles.xml"/></StorageBindings>
  + Copy and paste CustomStorageConfig.xml file to change the value of following nodes
    - ServiceEndPoint - URL of the IndexService
    - TemplateIdToIndex - Tcm Id Of component Template which we have created in step 1 CMS setup.
  + Copy the CustomStorageDAOBundles.xml XML file in the Content Delivery /bin/config folder

## MongoDB Index Service

* + Copy and paste **MongoDBIndexService on your server host it in IIS**
  + Copy the configuration folder as well.
    - You can update the log files path from Logging.config inside the configuration folder
  + Update the path of configuration\logging.config file in web.config of index service

## MongoDB Search Service

* + Copy and paste **MongoDBIndexService on your server host it in IIS**
  + Copy the configuration folder as well
    - Update the log files path in Logging.config inside the configuration folder
  + Update the path of configuration\logging.config file in web.config of index service

# To test the index and search services

## Index service

* 1. You can use fiddler for debugging
  2. Run index service on fiddler
     + http://localhost/Service1.svc/AddDocument
     + Input JSON which will generate by custom storage
     + **{"ServicePayload":{"DCP":"<Content xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' xmlns:xsd='http://www.w3.org/2001/XMLSchema' xmlns:xlink='http://www.w3.org/1999/xlink' xmlns:tcm='http://www.tridion.com/ContentManager/5.0' Title='Copy of Demo Of MongoDB' Id='tcm:2073-13667'><title>Demo Of MongoDB</title><description><![CDATA[Demo of component creation in Tridion using MongoDB]]></description><imageurl>/images/demo.png</imageurl><publication Id='tcm:0-2073-1' Title='03 Content Master' /></Content>","LanguageInRequest":"en"}}**
  3. Execute this request in Fidler and verify the results returned
  4. Check in mongoDB as well inside the collection\document which you have created and entered in the index service
  5. Result as 0 for success and 1 for failure
     + {"ResponseContext":{"EnvironmentContext":null,"FaultCollection":[]},"ServicePayload":{"ErrorMessage":"","Result":0}}
  6. Test this by publishing the Component as well

## Search Service

* 1. You use the fiddler for debugging
  2. Run the search service on fiddler
     + Url:- http://localhost/SearchSvc.svc/GetContentFromMongoDB
     + **{"ServicePayload":{"ContentType":"Content","Filters":[{"Key":"ItemURI","Value":"tcm:2073-13667"},{"Key":"publicationID","Value":"tcm:0-2073-1"}]}**
     + Execute the request in fiddler and verify the results returned
     + This service will get you the result based on Filters you provide and use /**/MAP/REDUCE**
     + **//Map/Reduce**

**var map** =

                        "function() {" +

                        "    for (var key in this) {" +

                        "        emit(key, { count : 1 });" +

                        "    }" +

                        "}";

**var reduce** =

                        "function(key, emits) {" +

                        "    total = 0;" +

                        "    for (var i in emits) {" +

                        "        total += emits[i].count;" +

                        "    }" +

                        "    return { count : total };" +

                        "}";

* + - Output will be in the JSON format
    - **{ "\_id" : ObjectId("5741539eef525465db9eb131"), "title" : "Demo Of MongoDB", "description" : "Demo of component creation in Tridion using MongoDB", "imageUrl" : "/images/demo.png", "ItemURI" :**

* + - **"tcm:2073-13667", "publicationID" : "tcm:0-2073-1" }**
    - You can have**AND, OR and NOT**as a logical operator to query the data from MongoDB
      * **{"ServicePayload":{"ContentType":"Content","Filters":[{"Key":"ItemURI","Value":"tcm:278-13667"},{"Key":"publicationID","Value":"tcm:0-278-1"}],"MongoDatabase":"customerDatabase","Table":"article","QueryType":"OR"}**