MongoDB Workloads with Control-M

Seamlessly embed MongoDB queries and operations into business workflows

INTRODUCTION

MongoDB Meets Application Workflow Orchestration
MongoDB describes itself as “the most popular database for modern apps” so chances are high that you probably have tons of “modern” data sources like your company’s websites, social media accounts, inventory records, sales transactions, IoT devices, etc. And if you are like most enterprises, it’s an ever-growing list that includes lots of traditional data sources too. You probably also have an arsenal of tools to help you ingest, extract, transform, process, and report on all this data.

A major challenge today is how to take all these data sources and data management tools and build an automated data pipeline. That’s the job of sophisticated and powerful application workflow orchestration.

In this technical note, we’ll demonstrate how you can quickly connect MongoDB, a leading document database solution, with Control-M, a powerful application workflow orchestration product. The result, you get enterprise-grade operational control over your data – from end-to-end workflow visibility to logs and output capture and management. Control-M can help you weave MongoDB (and all your other key technologies) into your entire data pipeline, in the cloud, on-premises, or both.

Automated MongoDB operations using the best practices established for the organization

- Immediately leverage their SQL expertise for MongoDB
- Manage MongoDB operations within the context of a complete business service

These goals are achieved by accessing MongoDB via the JDBC programming interface as implemented by a JDBC-compliant driver.

This document describes one such implementation, describing the steps required to achieve this integration.

SOLUTION COMPONENTS

Control-M for Databases

Control-M for Databases is an application plug-in that enables you to do the following:

- Connect to any supported database from a single computer with secure login
- Define and monitor Stored Procedure, SQL Script, SQL Server Integration Services (SSIS) Package, and Embedded Query database jobs
- Integrate Database jobs with other Control M jobs into a holistic business application workflow
- Introduce all Control-M capabilities to database processing, including advanced scheduling criteria, complex dependencies, quantitative and control resources, and variables

MongoDB

MongoDB is a document database designed for ease of development with the desired scalability and flexibility and the required querying and indexing.

Unity JDBC Driver

UnityJDBC provides simple data virtualization through support of any Java query and reporting software to combine data from multiple databases without requiring any new systems or server modifications.

Audience

This document is intended for anyone familiar with use of Control-M and has general knowledge of MongoDB. There is no attempt to provide in-depth guidance for these topics.

Additional information about the information presented here can be found in the links below:

General Control-M Documentation

Control-M Automation API

UnityJDBC Driver

MongoDB

DEFINE MONGODB TO CONTROL-M

This implementation uses the Unity JDBC driver although others are available. This is a common choice made by customers who have implemented this solution.

A free download and trial is available. If you decide to use this product, there will likely be some minimal additional charges. You can download the driver from:

http://unityjdbc.com/download.php

The downloaded jar file is an installation package. It is not a JDBC driver. Check with Unity for any updates however at the time this document was written, the installation process required java 1.8 or above and consisted of running the downloaded jar file like this:

```
java -jar UnityJDBC_Trial_Install.jar
```

This starts a graphical installation dialog which lets you select the target location.
The default path on a Windows machine is
C:\Program Files\UnityJDBC\
In this directory, find jar file “Unityjdbc.jar”
In the “drivers\Mongo\” directory, find file “mongo-java-driver-3.0.3.jar”.

Copy these two files:

- mongo-java-driver-3.0.3.jar
- Unityjdbc.jar

to a folder of your choosing, that is accessible to Control-M for Databases. This new folder will be specified as the “Path to Driver Folder” in section “Define Mongo Database Type”. Note: these files can be used on Windows or Linux.

MongoDB

This document assumes that a MongoDB instance is available. The only requirement for Control-M is an operational database instance with a username and password that can be used to connect to and operate on databases and collections.

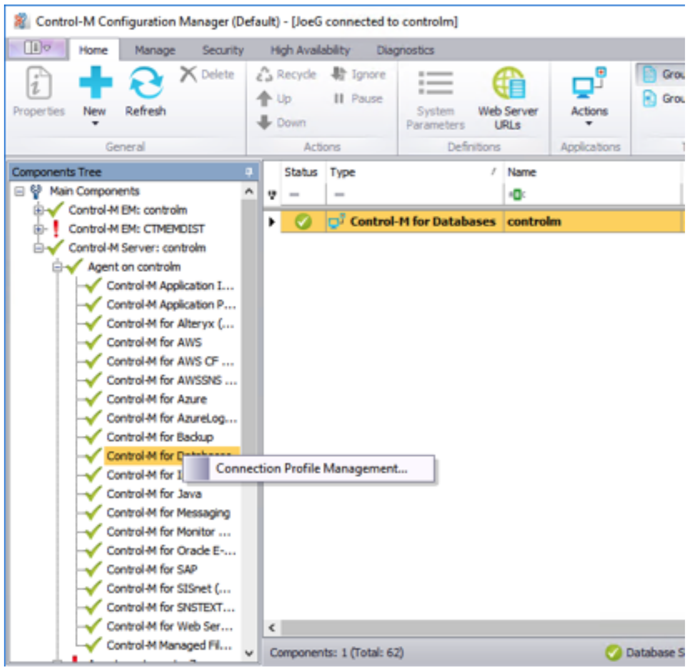
The assumed configuration in this document is that the MongoDB port is 27017 and connection from other hosts is enabled (for example by coding Bind = 0.0.0.0 in the mongodb.conf configuration file).

Define Mongo Database Type
Control-M for Databases provides out-of-the-box support for Oracle, MS SQL Server, DB2, Sybase and PostgreSQL. Any other database can also be supported via a JDBC driver. Once you have the driver, a simple configuraiton defines the database to Control-M and all further activities are then available for the newly-defined database just like for natively supported ones.

The definition can be performed either via graphical tools or via a code-like approach using JSON and Automation API.

Use Control-M Configuration Manager (CCM)
The CCM is the Control-M administrative console for managing your environment.

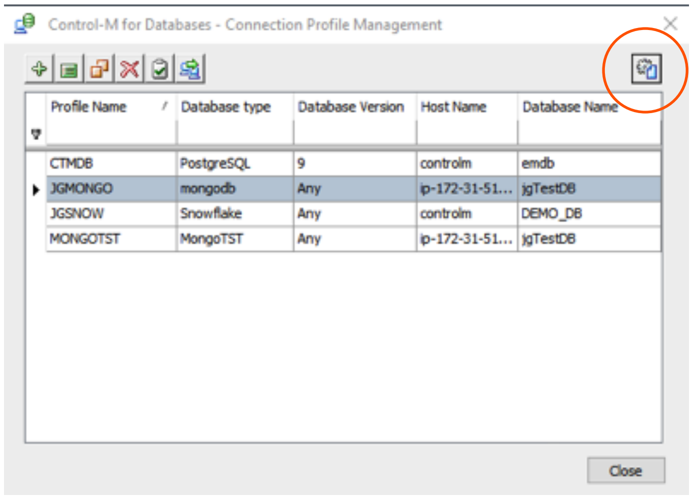
To define a Mongo database, navigate to the agent where Control-M for Databases is installed, right-click on that branch and select "Connection Profile Management" as shown in the image below:



Main Components
- Control-M EM: controlm
- Control-M EM: CTMEMDIST
- Control-M Server: controlm
  - Agent on controlm
    - Control-M Application I...
    - Control-M Application P...
    - Control-M for Alteryx (...
    - Control-M for AWS
    - Control-M for AWS CF ...
    - Control-M for AWSSNS ...
    - Control-M for Azure
    - Control-M for AzureLog...
    - Control-M for Backup
    - Control-M for Databases
    - Control-M for I
    - Control-M for Java
    - Control-M for Messaging
    - Control-M for Monitor ...
    - Control-M for Oracle E-...
    - Control-M for SAP
    - Control-M for SISnet (...
    - Control-M for SNSTEXT...
    - Control-M for Web Ser...
    - Control-M Managed Fil...

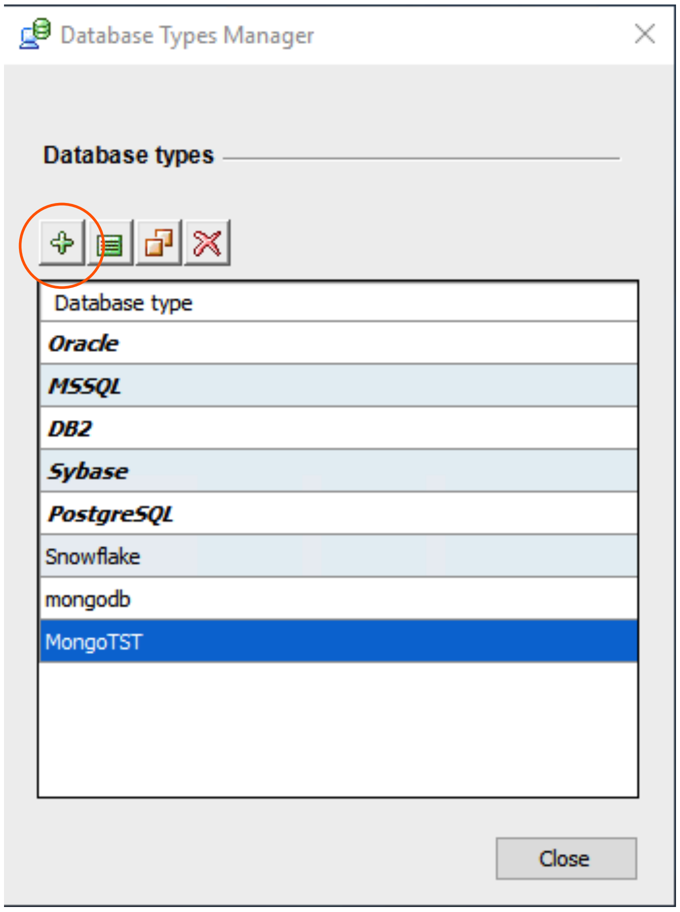
Status	Type	Name
✓	Control-M for Databases	controlm

Click the icon on the right to open the Database Types Manager.



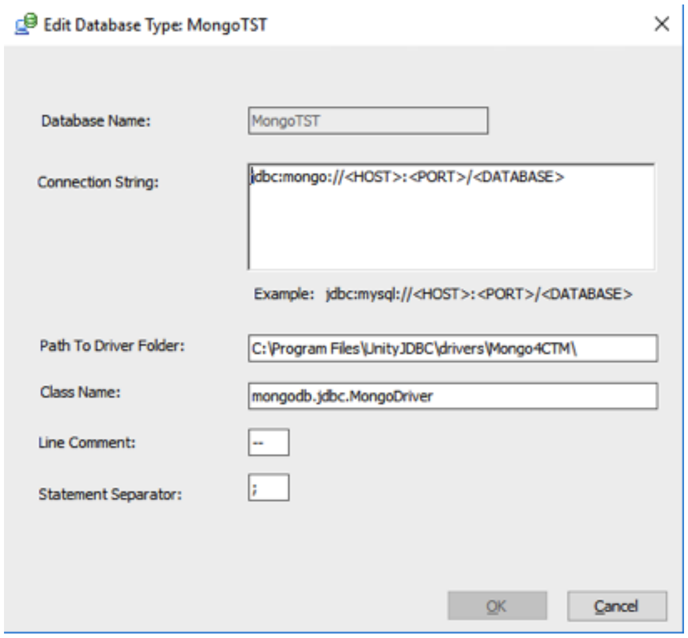
Profile Name	Database type	Database Version	Host Name	Database Name
CTMDB	PostgreSQL	9	controlm	emdb
JGMONGO	mongodb	Any	ip-172-31-51...	jgTestDB
JGSNOW	Snowflake	Any	controlm	DEMO_DB
MONGOTST	MongoTST	Any	ip-172-31-51...	jgTestDB

Select the "+" to add a new Database Type on the Database Types Manager form below:



Database type
Oracle
MSSQL
DB2
Sybase
PostgreSQL
Snowflake
mongodb
MongoTST

This action opens the Edit dialogue below.



Edit Database Type: MongoTST

Database Name:	MongoTST
Connection String:	jdbc:mongo://<HOST>:<PORT>/<DATABASE>
	Example: jdbc:mysql://<HOST>:<PORT>/<DATABASE>
Path To Driver Folder:	C:\Program Files\UnityJDBC\drivers\Mongo4CTM\
Class Name:	mongodb.jdbc.MongoDriver
Line Comment:	--
Statement Separator:	;

Database name: A logical name this database type should be referred to, such as Mongo or Snowflake, etc.

Connection String: This is a TEMPLATE for the connection string. It is used by Control-M to formulate the specific connection string based on information in the Connection Profile (see below). This approach enables a single Database Type definition to be used to connect to multiple databases.

<HOST> replaced by the hostname or IP Address specified in the connection profile

<PORT> replaced by the hostname or IP Address specified in the connection profile

<DATABASE> replaced by the database name specified in the connection profile

Path to Driver Folder: The fully-qualified path to the folder which contains the JDBC driver jar files (see "Prepapre UnityJDBC Driver")

Line Comment comments within SQL queries are prefixed by this character, for example:

-- This is a comment

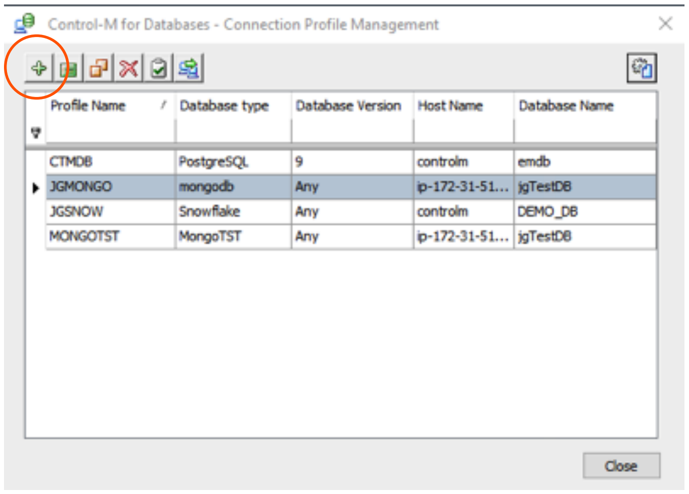
select * from inventory;

Statement Separator statements must be terminated by this character, for example:

select * from inventory;

Press OK to save the definition and Close to close the Database Types Manager.

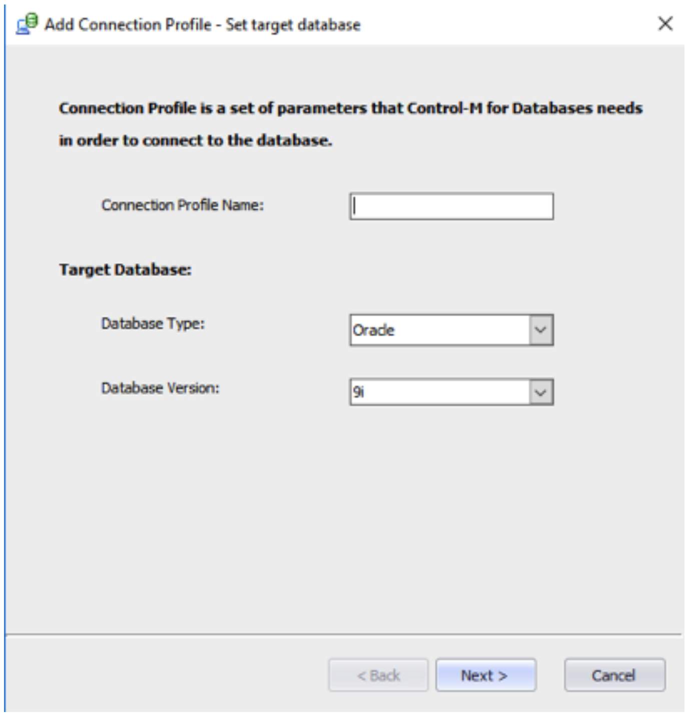
This brings you back to the Connection Profile Management form where you select "+", to define a new connection



Control-M for Databases - Connection Profile Management

Profile Name	Database type	Database Version	Host Name	Database Name
CTMDB	PostgreSQL	9	controlm	emdb
JGMONGO	mongodb	Any	ip-172-31-51...	jgTestDB
JGSNOW	Snowflake	Any	controlm	DEMO_DB
MONGOTST	MongoTST	Any	ip-172-31-51...	jgTestDB

profile in a form similar to the image below:



Add Connection Profile - Set target database

Connection Profile is a set of parameters that Control-M for Databases needs in order to connect to the database.

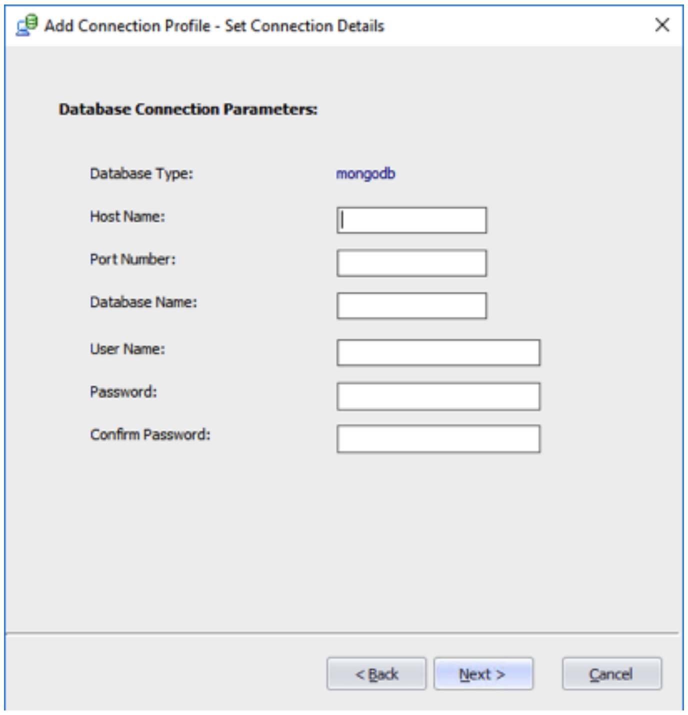
Connection Profile Name:	
Target Database:	
Database Type:	Oracle
Database Version:	9i

Connection Profile Name: A logical name for this profile that will be logical to users, like "MongoTest" or :"MongoProd"

Database Type: the name used for the Database Type definition in the previous set of steps, like "Mongo"

Database Version: choose "Any"

Choose "Next" to go on to the next form:



Database Connection Parameters:

Database Type:	mongodb
Host Name:	
Port Number:	
Database Name:	
User Name:	
Password:	
Confirm Password:	

Host Name: The hostname of the machine on which MongoDB is running.
Port Number: The Port defined in mongodb.conf for connections to Mongo.

The default is 27017.
Database Name: The nae of the database you wish and are authorized to connect to.
User Name: the user name defined in MongoDB authorized to use this database.
Password: the password for the above user.
Confirm Password: repeat the password for confirmation.
Select "Next" to page through the remaining forms.
The values are the same as any other database (see documentation or press F1).
Press Finish to complete and save the Connection Profile.

Use Automation API

This is an alternative method using JSON and either REST APIs or the "ctm" cli which implements those same REST services.
Create a json file. In this example, it is named Mongo_DbDef.json
This is the file:

```
{
 MongoDB": {
   "Type": "Driver:Jdbc:Database",
   "TargetAgent":"controlm",
   "StringTemplate":"jdbc:mongo://<HOST>:<PORT>/<DATABASE>",
   "DriverJarsFolder":"C:\\Mongo4CTM\\",
   "ClassName":"mongodb.jdbc.MongoDriver",
   "LineComment" : "--",
   "StatementSeparator" : ";"
 }
```

MongoDB is the Database name.
Type: Json syntax for this definition type.

TargetAgent: the Control-M Agent where this definition is to be deployed

StringTemplate: the Connection String Template described above.

DriverJarsFolder: location where the Unity JDBC jar files were placed

ClassName: the JDBC class name provided by the JDBC driver documentation

Line Comment: as described above.

StatementSeparator: as described above.

Validate the syntax of your json file using the Automation API "build" function:

```
ctm build Mongo_DbDef.json
```

You should get a response similar to:

```
[
 {
   "deploymentFile": "Mongo_DbDef.json",
   "successfulFoldersCount": 0,
   "successfulSmartFoldersCount": 0,
   "successfulSubFoldersCount": 0,
   "successfulJobsCount": 0,
   "successfulConnectionProfilesCount": 0,
   "successfulDriversCount": 1,
   "isDeployDescriptorValid": false
 }
]
```

Once the syntax is correct, use the "deploy" service to create the entry. The following is using the "ctm" cli:

```
ctm deploy Mongo_DbDef.json
```

The response should be similar to this:

```
[
 {
  "deploymentFile": "Mongo_DbDef.json",
  "successfulFoldersCount": 0,
  "successfulSmartFoldersCount": 0,
  "successfulSubFoldersCount": 0,
  "successfulJobsCount": 0,
  "successfulConnectionProfilesCount": 0,
  "successfulDriversCount": 1,
  "isDeployDescriptorValid": false,
  "deployedDrivers": [
   "MongoDB_API"
  ]
 }
]
```

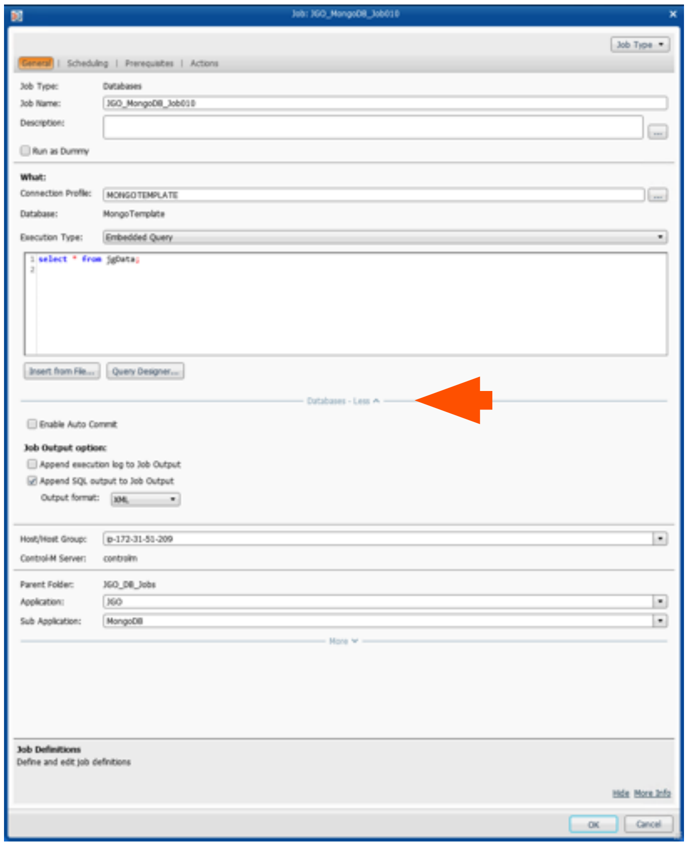
BUILD AND EXECUTE JOBS

The goal of the integration described in this document is to execute Control-M jobs that perform MongoDB tasks.

Graphical Clients

This example uses the Workload Automation client. Control-M/Web Planning can also be used.

Select the Databases job type.



Job Type: Databases
Job Name: JGO_MongoDB_Job010
Description:
Run as Dummy

What:
Connection Profile: MONGOTEMPLATE
Database: MongoTemplate
Execution Type: Embedded Query

```
select * from jgData;
```

Enable Auto Commit

Job Output option:
Append execution log to Job Output
[✓] Append SQL output to Job Output
Output format: XML

Host/Host Group: ip-172-31-51-209
Control-M Server: controlm

Parent Folder: JGO_DB_Jobs
Application: JGO
Sub Application: MongoDB

Job Definitions
Define and edit job definitions

Connection Profile: Choose a profile created in the previous section ("Define Mongo Database Type")
Execution Type: Choose the SQL query you wish to execute. It can be embedded, as above, or a script stored in a file.
Expand the **"More"** section (highlighted by the arrow) to choose the type of output you wish to capture in the Control-M output.

Automation API

Sample JSON to use the connection profile built in the previous sections:

```
{
 "JGO_MongoDB" :
 { "Type" : "Folder",
              "SiteStandard" : "JGO_Std",
              "ControlmServer" : "controlm",
              "OrderMethod" : "Manual",
   "JGO_Mongo_SQL_Job001" :
              { "Type" : "Job:Database:SQLScript",
                            "OutputExcecutionLog": "N",
                            "OutputSQLOutput" : "Y",
                            "SQLOutputFormat" : "XML",
                            "SQLScript" : "/home/ctmagent/script.sql",
                            "ConnectionProfile" : "MONGOCP",
                            "Host" : "ip-172-31-51-209",
                            "SubApplication" : "MongoDB",
                            "Application" : "JGO",
                            "Description" : "MongoDB SQL Query",
                            "RunAs" : "joeuser",
                            "Output": {
 "Destination": "/home/ctmagent/ mongo_
output_%%ORDERID._%%$DATE._%%TIME..xml",
      "Operation": "Copy"
                            }
   }
 }
}
```

Job:Database:SQLScript this job executes a SQL script rather than an in-line embedded query.

By coding "OutputExcecutionLog": "N" and OutputSQLOutput" : "Y"   only the SQL output is written to Control-M output. Then adding "Output": {"Destination": "/home/ctmagent/ mongo_ output_%%ORDERID._%%$DATE._%%TIME..xml", "Operation": "Copy"}, the SQL output is copied ot the output destination and given a file name made unique by Control-M variables containing the current date and time.

Check the JSON syntax with the Automation API "build" service via the ctm cli:

```
ctm build mongodb_sql.json
```

Successful response appears similar to:

```
[
 {
   "deploymentFile": "mongodb_sql.json",
   "successfulFoldersCount": 0,
   "successfulSmartFoldersCount": 1,
   "successfulSubFoldersCount": 0,
   "successfulJobsCount": 1,
   "successfulConnectionProfilesCount": 0,
   "successfulDriversCount": 0,
   "isDeployDescriptorValid": false
 }
]
```

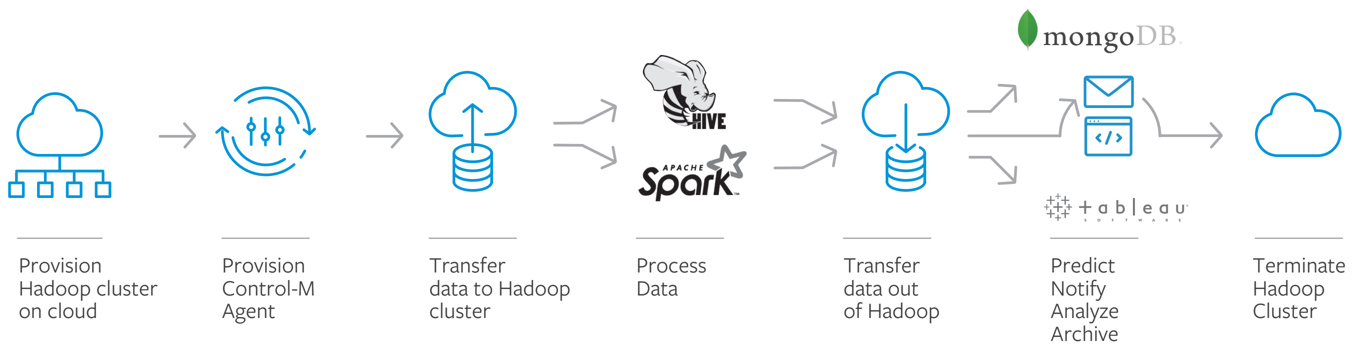
You can then execute the job using the "run service or via the cli as follows:

```
ctm run mongodb_sql.json
```

See documentation for how to monitor job execution status, retrieve the log or output and perform other operations like rerunning the job.

SIMPLIFY AND SCALE COMPLEX DATA PIPELINES

With the above steps, you have successfully connected MongoDB with the power of application workflow orchestration. But, don't stop there; Control-M can automate your entire data pipeline, allowing you to ingest and process data from all kinds of platforms, including Hadoop, Spark, EMR, RedShift and others. Just check out the figure below to see what it looks like.



HIVE
Apache Spark
mongoDB
tableau software
Provision Hadoop cluster on cloud
Provision Control-M Agent
Transfer data to Hadoop cluster
Process Data
Transfer data out of Hadoop
Predict Notify Analyze Archive
Terminate Hadoop Cluster

CONCLUSION

MongoDB has gained popularity and is now a common component of many applications. The ability to execute queries within the workflow of a business application, provides enhanced visibility and manageability for such tasks. Furthermore, MongoDB process lineage can be captured to enhance auditing and simplify governance.With Control-M you will spend more time uncovering actionable intelligence, and less time worrying about access to data. Control-M gives you an end-to-end view of data pipelines at every stage from data ingestion to processing to analytics. You'll be able to manage business SLAs for service delivery and resolve critical issues before deadlines are missed.

FOR ADDITIONAL INFORMATION ON CONTROL-M

Visit our web page