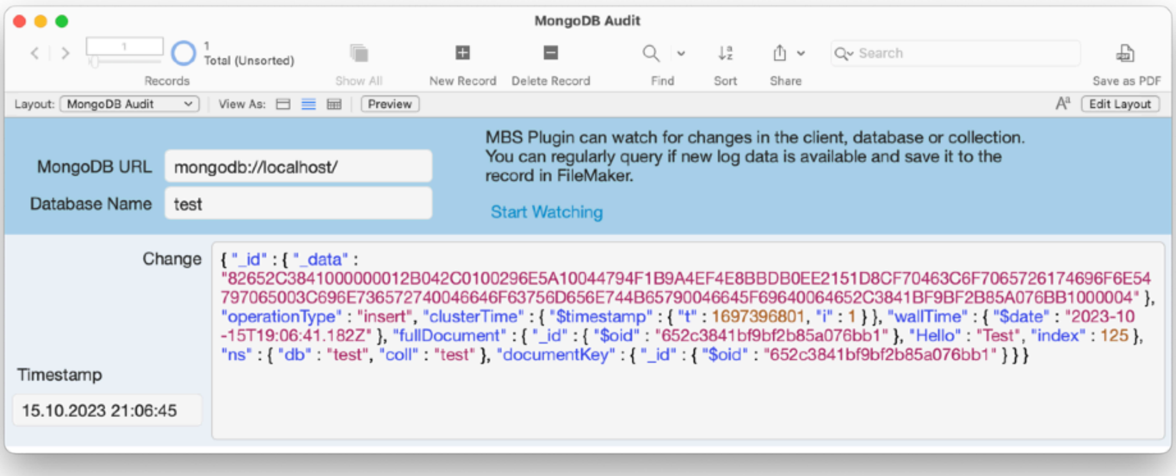
Watch MongoDB Database

Did you know that MongoDB has a watch feature (change stream) to see what happens in the database?



MBS Plugin can watch for changes in the client, database or collection. You can regularly query if new log data is available and save it to the record in FileMaker.

MongoDB URL: mongodb://localhost/

Database Name: test

Change:

```
{ "_id" : { "_data" : 
"82652C3841000000012B042C0100296E5A10044794F1B9A4EF4E8BBDB0EE2151D8CF70463C6F7065726174696F6E54
797065003C696E736572740046646F63756D656E744B65790046645F69640064652C3841BF9BF2B85A076BB1000004" }, 
"operationType" : "insert", "clusterTime" : { "$timestamp" : { "t" : 1697396801, "i" : 1 } }, "wallTime" : { "$date" : "2023-10
-15T19:06:41.182Z" }, "fullDocument" : { "_id" : { "$oid" : "652c3841bf9bf2b85a076bb1" }, "Hello" : "Test", "index" : 125 }, 
"ns" : { "db" : "test", "coll" : "test" }, "documentKey" : { "_id" : { "$oid" : "652c3841bf9bf2b85a076bb1" } } }
```

Timestamp: 15.10.2023 21:06:45

You can watch a client (your connection) with MongoDB.WatchClient, a database with MongoDB.WatchDatabase or just a collection with MongoDB.WatchCollection function. Once watch is active, you can query MongoDB.NextChange regularly and see if you get a JSON with a new change. In our example database, we run a schedule to look for new changes every 5 seconds and this shows the change as new record in our example. You can use this feature for various purposes, including:

1. **Real-time Data Synchronization:** You can use watch feature to keep data in sync across different parts of your application or between different services. When a change occurs in a MongoDB collection, you can capture that change and react accordingly to keep your data up-to-date, e.g. copy values to FileMaker.
2. **Notification Systems:** You can implement real-time notification systems, such as sending alerts, emails, or messages to users or applications when specific changes or events happen in the database. This is useful for building chat applications, social networks, or any application that requires real-time updates.
3. **Data Replication:** Change Streams can be used to replicate data between MongoDB instances or databases. For example, you can replicate data from a primary MongoDB server to one or more secondary servers for fault tolerance and load distribution.
4. **Auditing and Logging:** You can track and log changes made to your data for auditing purposes. This can help you maintain a record of who made changes, when they were made, and what the changes were.
5. **Triggers and Workflow Automation:** You can use Change Streams to trigger specific actions or workflows when certain events occur in your

database. For example, you can automatically update related documents or trigger external services in response to changes.

6. **Data Analytics and Reporting:** Real-time data feeds from Change Streams can be used for building analytics dashboards and generating real-time reports based on the changes happening in your MongoDB database.
7. **Caching:** You can use Change Streams to update cache data whenever there are changes in the database. This can help improve the performance of your applications by serving frequently accessed data from cache.
8. **Building Live Feeds:** If you're developing applications that require live feeds or activity streams (e.g., social media timelines or news feeds), Change Streams can help you efficiently implement these features.
9. **Data Transformation:** You can use Change Streams to trigger data transformation processes, ensuring that your data remains consistent and conforms to the desired format.
10. **Reactive Programming:** Change Streams can be integrated into reactive programming frameworks or libraries to create responsive and event-driven applications.

MongoDB Change Streams provide a flexible and powerful way to react to database changes in real-time, making them a valuable tool for a wide range of applications that require real-time data processing and synchronization.

Back in FileMaker you would open the database with MongoDB.OpenDatabase and then start watching changes in that database with MongoDB.WatchDatabase and then call MongoDB.NextChange regularly to pick up new data. That may be a script scheduled to run on FileMaker Server in an endless loop to process incoming changes. Or like in the example below use Schedule to evaluate an expression every 5 seconds to check for changes and if a change happened, trigger a script to pick it up.

```
Set Variable [ $r ; Value: MBS( "MongoDB.OpenDatabase"; $Mongo; MongoDB
Audit::Database Name) ]
If [ MBS("IsError") ]
	Set Variable [ $x ; Value: MBS( "MongoDB.Release"; $Mongo ) ]
	Show Custom Dialog [ "Failed to open database." ; $r ]
	Exit Script [ Text Result:    ]
End If
#
Set Variable [ $r ; Value: MBS( "MongoDB.WatchDatabase"; $Mongo; "{}" ) ]
If [ MBS("IsError") ]
	Set Variable [ $x ; Value: MBS( "MongoDB.Release"; $Mongo ) ]
	Show Custom Dialog [ "Failed to watch database." ; $r ]
	Exit Script [ Text Result:    ]
End If
#
```

```
# success
Set Variable [ $$MongoConnection ; Value: $Mongo ]
# schedule an expression to check status regularly every 5 seconds...
Set Variable [ $$MongoSchedule ; Value: MBS( "Schedule.EvaluateAfterDelay";
        5 /* DelaySeconds*/;
        "Let([
        json = MBS( \"MongoDB.NextChange\"; $$MongoConnection );
        r = If(Length(json) > 0; MBS( \"FM.RunScript\"; \"MongoDB Audit\";
\"LogChange\"; json); 0)]; \"\")";
        "" /* ScriptFileName*/;
        "" /* ScriptName*/;
        5 /* RepeatDelay*/ ) ]
```

The LogChange script gets the JSON of the change passed and stores it in a new record using our FM.InsertRecord function with a bit of color thanks to JSON.Colorize function. Since our insert command uses SQL, we don't need to do a layout change and can run with whatever layout is currently showing. Also if FileMaker is busy, the script triggers will queue and execute later when there is time.

```
# we trigger this script to do the logging
Set Variable [ $json ; Value: MBS( "JSON.Colorize"; Get(ScriptParameter)) ]
Set Variable [ $r ; Value: MBS( "FM.InsertRecord"; "MongoDB Audit"; "MongoDB
Audit"; "Change"; $json) ]
```

Please try it and enjoy the new feature for 13.5 plugin version of MBS FileMaker Plugin.