Warning: This site documents Glimpse 2.0.0. It is a work in progress, and still has a long way to go. For Glimpse v1 documentation, please visit getglimpse.com/Docs.

For some information about locking Glimpse 2.0.0-beta1 down, check out this gist.

Download insert.json directly.

Contents:
Warning: This site documents Glimpse 2.0.0. It is a work in progress, and still has a long way to go. For Glimpse v1 documentation, please visit getglimpse.com/Docs.

To install Glimpse 2.0.0, please read the Installing Glimpse v2 Beta1 post on our blog.
CHAPTER 2

Contributing Documentation

Warning: This site documents Glimpse 2.0.0. It is a work in progress, and still has a long way to go. For Glimpse v1 documentation, please visit getglimpse.com/Docs.

Glimpse documentation is stored in GitHub and is hosted on the ReadTheDocs.org platform at docs.getglimpse.com. Any individual page in the Glimpse documentation can be edited by clicking “Edit on GitHub” link at the top of the page. For larger contributions, or to more quickly work with multiple pages, ReadTheDocs can be run locally.

2.1 Installing ReadTheDocs

ReadTheDocs is based on the popular open source project Sphinx. Sphinx is built in Python and leverages a powerful Markdown-like language called reStructuredText (reST).

The steps to install are:

1. Download and Install Python 3.5 (or later) from the Python Downloads page.

   Tip: To determine the current version of Python installed on your machine, use python --version

   Note: On OS X, the Python 3.x binary is python3.

2. Using git, clone the Glimpse Docs repository locally: git clone git@github.com:Glimpse/Docs.git docs

3. Change into the docs directory: cd docs

4. Using pip, Python’s package manager, install Sphinx: pip install sphinx

   Note: On OS X, the Python 3.x package manager is pip3.

5. Install the ReadTheDocs’s Sphinx theme: pip install -U sphinx_rtd_theme

6. Install sphinx-autobuild: pip install sphinx-autobuild

7. Run sphinx-autobuild with these two options: sphinx-autobuild source build/html

2.2 Editing Documentation

Edit any .rst file in the source directory. When changes are saved, sphinx-autobuild will automatically rebuild the site and refresh your browser.

For help with reStructuredText itself, use the reStructuredText Primer in Sphinx’s documentation.
This document outlines the standard message formats that Glimpse uses to transport and store data.

**Tip:** Messages are documented using JSON Schema. For the uninitiated, there’s a great open source JSON Schema book.

**Tip:** Messages can be loosely validated with online tooling. To author your own schema, be sure to check out the online generator.

### 3.1 Logging

#### 3.1.1 Write

**Type:** `log-write`

**Schema:** (Link)

**Note:** The `pattern` property should be a JavaScript compatible Regular Expression representation of log messages that leverage string interpolation. The purpose of this property is allow Glimpse clients to treat interpolated values specially, as demonstrated on JSFiddle.

Regular Expressions were chosen since they are nearly universal and can represent any interpolation format. (%s, \{0\}, \{1:d\}, \{foo\}, \$\{bar\}, etc.)

### 3.2 MongoDB

#### 3.2.1 Insert

**Type:** `data-mongodb-insert`
3.2.2 Read

Reads tend to be more complex than writes because of cursors.
This annotated time line shows the approach we’ll take to measure duration, for now.

Instrumented Collection Operations
- `count()`
- `findOne()` - In Glimpse for Node.js, messages for `findOne()` will have an operation field with a value of “next”. This is because the `findOne()` implementation ultimately calls Cursor’s `next()` method.

Instrumented Cursor Operations
- `toArray()` - time measured by Glimpse is the time to return all of the results to the client.
• forEach() - time measured by Glimpse is the time to return the first result. Only the first call to forEach() will have a Glimpse message associated with it.
• next() - Same behavior as forEach().
• each() - Same behavior as forEach().
• nextObject() - Same behavior as forEach.

Type: data-mongodb-read

Schema: (Link)

Note: The options may need to be cobbled together based on these possible calls on the cursor itself.

3.2.3 Update

Type: data-mongodb-update

Schema: (Link)

3.2.4 Delete

Type: data-mongodb-delete

Schema: (Link)

3.2.5 Undocumented Methods

The MongoDB Collection Driver API has lots of methods. The above schemas only cover a subset of them. The following methods are yet to be covered:

• aggregate
• bulkWrite
• createIndex
• createIndexes
• distinct
• drop
• dropAllIndexes
• dropIndex
• dropIndexes
• ensureIndex
• findAndModify
• findAndRemove
• findOneAndReplace
• geoHaystackSearch
• geoNear
• group
• indexes
• indexExists
• indexInformation
• initializeOrderedBulkOp
• initializeUnorderedBulkOp
• isCapped
• listIndexes
• mapReduce
• options
• parallelCollectionScan
• reIndex
• rename
• save
• stats
• update

3.3 Web

3.3.1 Request

Type: web-request
Schema: (Link)

3.3.2 Response

Type: web-response
Schema: (Link)
The primary purpose of the Glimpse server is to store messages generated by Glimpse agents, to be queried for by Glimpse clients.

4.1 Contexts

A “context” represents a related set of messages. Each context has a unique ID and a type. While the Glimpse server can store arbitrary contexts, it specifically tracks a type of context called a request (i.e. an individual HTTP request).

4.1.1 Requests

The Glimpse server tracks a specific set of “indices” for each request to enable efficient filtration by a Glimpse client through the request-history resource.

<table>
<thead>
<tr>
<th>Index</th>
<th>Type</th>
<th>Behavior</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>request-duration</td>
<td>number</td>
<td>Overwritten</td>
<td>The total length of time taken by the request (in milliseconds)</td>
</tr>
<tr>
<td>request-url</td>
<td>string</td>
<td>Overwritten</td>
<td>The URL of the request</td>
</tr>
<tr>
<td>request-method</td>
<td>string</td>
<td>Overwritten</td>
<td>The HTTP method of the request (e.g. GET, POST, etc.)</td>
</tr>
<tr>
<td>request-userId</td>
<td>string</td>
<td>Overwritten</td>
<td>The ID of the user associated with the request</td>
</tr>
<tr>
<td>request-statuscode</td>
<td>number</td>
<td>Overwritten</td>
<td>The HTTP status code returned by the request (e.g. 200, 401, etc.)</td>
</tr>
<tr>
<td>request-datetime</td>
<td>string</td>
<td>Overwritten</td>
<td>The time at which the request began</td>
</tr>
<tr>
<td>request-tags</td>
<td>array</td>
<td>Aggregated</td>
<td>A set of tags associated with the request (case-sensitive)</td>
</tr>
</tbody>
</table>
The `Behavior` column indicates how the store handles multiple messages of a given context containing values for the same index. `Overwritten` means that later messages overwrite the indices of earlier messages. `Aggregated` means that the indices of all messages for a given context are aggregated.

**Note:** The index names are case-sensitive.

## 4.2 Message Store

The Glimpse server groups stored messages by their context. That is, messages with the same context ID are grouped together.

### 4.2.1 In-memory Store

By default, the Glimpse server will use an in-memory message store. Accumulated messages will be lost if/when the server is shut down. To limit the amount of memory used by the store, the store will retain, at most, 500 individual contexts.

**Note:** Currently, there is no limit on the number of messages associated with any given context.

When that limit is reached, contexts will be discarded in “least-recently-used” fashion. That is, each time a message is stored by the Glimpse server, its associated context is moved to the beginning of the context list. The context at the end of that list will be the first to be discarded.

## 4.3 Message Query

The Glimpse server allows clients to query the message store in one of two ways: by context ID or by a combination of request indices (for contexts of type `request`). In both cases, messages can be further filtered by type.

### 4.3.1 Query Messages

The entire message store can be queried by context ID and/or type:

<table>
<thead>
<tr>
<th>Query Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context ID</td>
<td>string</td>
<td>If specified, only messages related to this context will be returned.</td>
</tr>
<tr>
<td>Type</td>
<td>array</td>
<td>If specified, only messages matching any of these types will be returned.</td>
</tr>
</tbody>
</table>

### 4.3.2 Query Requests

Request contexts can be queried by a combination of request indices and/or type:
<table>
<thead>
<tr>
<th>Query Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration Minimum</td>
<td>number</td>
<td>If specified, messages of requests of a shorter duration will not be returned.</td>
</tr>
<tr>
<td>Duration Maximum</td>
<td>number</td>
<td>If specified, messages of requests of a longer duration will not be returned.</td>
</tr>
<tr>
<td>URL Contains</td>
<td>string</td>
<td>If specified, only messages of requests with URLs matching this fragment will be returned (case-sensitive).</td>
</tr>
<tr>
<td>Method List</td>
<td>array</td>
<td>If specified, only messages of requests with one of these method types will be returned (case-sensitive).</td>
</tr>
<tr>
<td>Status Code Minimum</td>
<td>number</td>
<td>If specified, messages of requests with smaller status codes will not be returned.</td>
</tr>
<tr>
<td>Status Code Maximum</td>
<td>number</td>
<td>If specified, messages of requests with larger status codes will not be returned.</td>
</tr>
<tr>
<td>Tag List</td>
<td>array</td>
<td>If specified, only messages of requests with one of these tags will be returned (case-sensitive).</td>
</tr>
<tr>
<td>Request Time Before</td>
<td>string</td>
<td>If specified, only messages of requests starting before this time will be returned.</td>
</tr>
<tr>
<td>User ID</td>
<td>string</td>
<td>If specified, only messages of requests associated with this user will be returned (case-sensitive).</td>
</tr>
</tbody>
</table>
CHAPTER 5

Indices and tables

- genindex
- modindex
- search