# API Monitoring

## Introduction

API Monitoring is a way to provide the Component level monitoring and to check the connectivity between the different components for a particular Customer given a time interval and raise a mail if an issue to the given address registered with the program.

## Tools Used

API Monitoring uses an individual client machine setup with two applications installed:

* Hawk Eye
* Couch DB

Hawk Eye: The Application is deployed on Tomcat Server which uses Couch DB as a BackEnd to Store the valid requests sent to validate if the components working status are good and parses the response to check whether it’s a valid response or not (for now the application only validates a 200 OK response).

Notes : Tomcat Installed – What version, directory, imp commands. Installation instructions for tomcat.

Hawkeye war – version of the war. Directory. ssh command to push the war. Whose ssh key?

Couch DB: Apache CouchDB, commonly referred to as CouchDB, is an open source database that focuses on ease of use and on being "a database that completely embraces the web". It is a NoSQL database that uses *JSON* to store data, JavaScript as its query language using MapReduce and HTTP for an API.

Notes: installation instrutions. Basic commands .

GOC UI:

UI :

Notes: runs on tomcat. Installation instruction for application. Version.

## Using the Tool

The Hawk Eye application is hosted on a Tomcat sever on the client instance which rarely requires any change, the only part the end-user with be looking at will be Registering/de-registering the request i.e. monitoring requests for the components.

### Working with Couch DB

Applications interact with CouchDB via HTTP. The following demonstrates a few examples using [cURL](http://en.wikipedia.org/wiki/CURL" \o "CURL), a command-line utility. These examples assume that CouchDB is running on [localhost](http://en.wikipedia.org/wiki/Localhost" \o "Localhost)(127.0.0.1) on port 5984.

|  |  |  |
| --- | --- | --- |
| **Action** | **Request** | **Response** |
| Accessing server information | curl http:**//**127.0.0.1:5984**/** | {  "couchdb": "Welcome",  "version":"1.1.0"  } |
| Creating a database named **monitor** | curl -X PUT http:**//**127.0.0.1:5984**/**monitor | {"ok": **true**} |
| Attempting to create a second database named **monitor** | curl -X PUT http:**//**127.0.0.1:5984**/**monitor | {   "error":"file\_exists",   "reason":"The database could not be created, the file already exists." } |
| Retrieve information about the **monitor**database | curl http:**//**127.0.0.1:5984**/**monitor | {  "db\_name": "wiki",  "doc\_count": 0,  "doc\_del\_count": 0,  "update\_seq": 0,  "purge\_seq": 0,  "compact\_running": **false**,  "disk\_size": 79,  "instance\_start\_time": "1272453873691070",  "disk\_format\_version": 5  } |
| Delete the database **monitor** | curl -X DELETE http:**//**127.0.0.1:5984**/**monitor | {"ok": **true**} |
| Create a document, asking CouchDB to supply a document id | curl -X POST -H "Content-Type: application/json" -d  '{ "text" : "Monitor on CouchDB", "rating": 5 }'  http:**//**127.0.0.1:5984**/**monitor | {  "ok": **true**,  "id": "123BAC",  "rev": "946B7D1C"  } |

### Given below is the sample JSON request data to be sent to register a request in Couch Db.

Parameters used in JSON data:

* **Name**: Given name to the Request as identification.
* **Emails**: Email address or mailing list to be given here if the issue is raised whom all to send the mail.
* **Apis**: This JSON element holds the list of different api requests to be sent to validate the components.
  + **Name**: Identification name for the API request.
  + **URI**: The valid request to be given here without the header parameters.
  + **VERB**: verb like what kind of request to be sent. Eg: GET,POST,PUT,HEAD etc
  + **Payload**: Any request body to be sent with the request should be given here.
  + **pollInterval** : After what interval the request to be sent ( specified in seconds).
  + **thresholdTries**: The Number of retries after which this should be raised as an issue and sent as a mail.
  + **thresholdTime**: How long the sever has to wait for the response to return with a valid response.
  + **onFailurePollInterval**: After the raising the issue what time should the server wait to check for the status again for the API.

### Sample JSON to be sent as Data to Couch DB Request:

{  
"org": {  
        "name": "ValidName",  
        "emails": "abc@xyz.com",  
        "apis": [  
            {  
                "name": "ContentAPI",  
                "uri":"http://{HOSTNAME}/content/items/v1/?apiKey=7c1a2593",  
                "verb": "GET",  
                "headers": {},  
                "payload": "",  
                "pollInterval": "3",  
                "thresholdTries": "3",  
                "thresholdTime": "30",  
                "onFailurePollInterval": "60"  
            }  
                ]  
          }  
}

## Considerations:

* If any changes have been in Database the Tomcat server must be restarted to reflect the changes as Tomcat only reads the Database on Startup.
* If you need to delete the database, please restart the Tomcat to create database automatically.
* If any request is deleted/ altered in the database, the Hawk Eye must be restarted to reflect the change in the request.
* The Logs are listed on the Tomcat/Logs directory to have a better idea on the issue raised.