# Koockoo RESTful API

## Service Url:

GET: <http://chatservicelocator.appspot.com/services/any>

Response: {url:serviceUrl} - the base url for all api URLS

For example: {url:http://102.103.105.105:7070/koockoo-services/api}

## Resources

All responses are JSON and wrapped as

{

"data":{return data},

type":"event",

"name":"message",

"code":"return code",

"message":"return message",

"success":"true|false"

}

Format dictated by ExtJS.

Property data contains actual return data which is described in response field. Other fields are for service purpose. If request successfully received by the end point the HTTP code is always 200.

But **response.code** may be different.

Supported codes

|  |  |  |
| --- | --- | --- |
| **code** | **message** | **success** |
| 200 | “” | true |
| 500 | Internal Error: Detailed message will be provided | false |
| 400 | Bad Request: Detailed message will be provided | false |
| 401 | Unauthorized: The request requires user authentication | false |
| 4011 | Invalid Credentials | false |

### Auth

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **URL** | **Params** | **Response** | **Description** |
| GET | ***/auth/signin/operator***  Signin with credentials | login=email address  password=paswd | Return Token  data:{token:$token} |  |
| GET | ***/auth/signin/guest***  Sign in as guest | displayName=name | Return Token  data:{token:$token} |  |
| POST | ***/auth/signout***  Invalidate token | N/A | Generic Response  data:”” |  |
| POST | ***/auth /register/operator***  Create new credentials | email=emailaddr  password=paswd | Generic Response  data:”” |  |
| POST | ***/auth /register/account***  Create new credentials | email=emailaddr  password=paswd | Generic Response  data:”” |  |
| POST | ***/auth/recover***  Retrieve credentials over email | N/A | Generic Response  data:”” |  |
| POST | ***/auth/delete***  Delete credentials.  User must be signed in. | N/A | Generic Response  data:”” |  |

All the resources below require connection to be authenticated first.

User app get a security token. The token get included in header or cookie of each server request.

**TODO: figure the best way to set and pass token.**

Token must be validated before request gets to controller. Most likely in filter.

### ChatSession

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **URL** | **Params** | **Response** |
| GET | ***/sessions/user***  Get sessions associated with the current user | status=Active|Pending|Closed  to filter the response | data: [{ChatSession}] |
| GET | ***/sessions/{sessionId}***  Retrieve session by id | status=Active|Pending|Closed | data: {ChatSession} |
| POST | ***/sessions/group/{groupId}***  Create new ChatSession | N/A | data: {ChatSession} |
| POST | ***/sessions/{sessionId}***  Update status of existing session | status=Active|Pending|Closed | data: {ChatSession} |

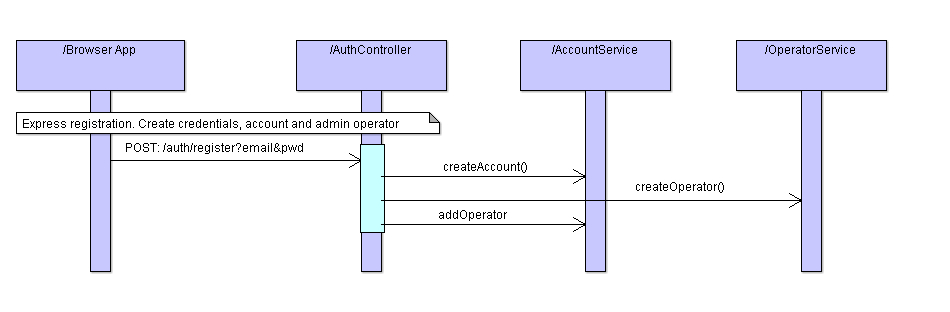
### ChatMessage

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **URL** | **Params** | **Response** |
| GET | /messages/session/{sessionId} | Status=Active|Pending|Closed |  |
| POST | /messages/session/{sessionId} | N/A |  |
| PUT | /sessions/{sessionId} | Status=Active|Pending|Closed |  |

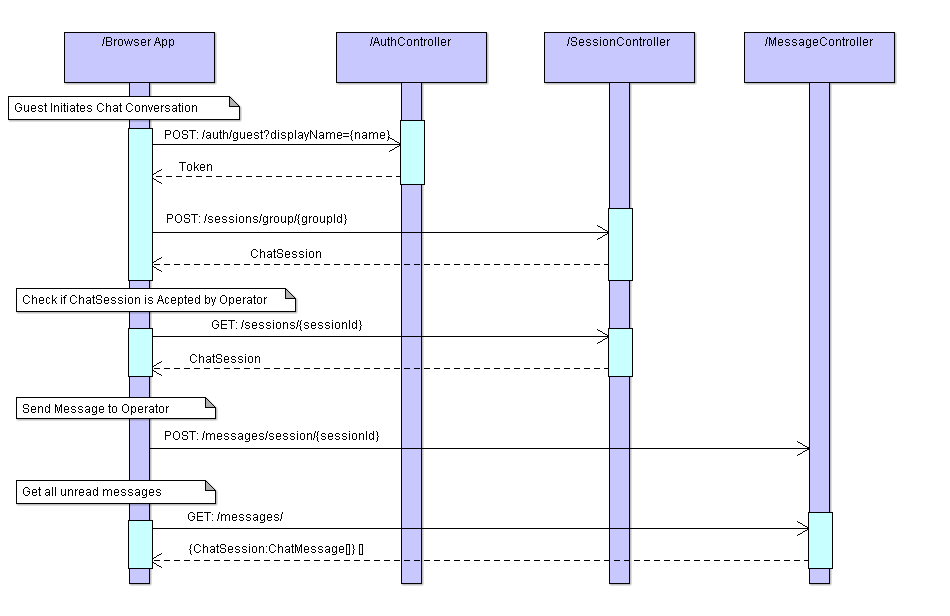
## Sequence Diagrams

Diagrams demonstrate the use of REST API functions.

### Account registration



### Guest initiates chat with operator



### Operator signing in to App and services customers

