|  |
| --- |
| CLOUD PSS |
| Project Tortoise |
| REST API Documentation |
|  |
| **Pritam Nikam** |
| **6/24/2015** |

|  |
| --- |
| Building a back-end API layer introduces a whole new layer of coordination between Tortoise server and mobile client code. While there are many aspects to this delicate dance of communication, one key ingredient to minimizing back-and-forth-confusion-about what-call-does-what, is consistently communicating about your API endpoints. This document provides in depth discussion on REST API documentation used in Project Tortoise. |

1. **INTRODUCTION**

The interface between the Tortoise cloud server and the client application is based on REST architecture principle. This necessitates that all APIs defined are RESTful, in the sense that the client views the accessible information as a collection of objects or resources that can be uniquely addressed and accessed. In this document we define all entities on the server as resources or objects rather than as functions or methods.

The following example illustrates the difference between a function based approach and a resource based approach.

Assume that we have a database of schools together with details of students enrolled in each school. Each school has unique id and so does each student. Let’s say we want to fetch details about a student with student\_id=1500 enrolled in a school with school\_id=25.

1.1 Function Based Approach:

In function based approach, a HTTP GET method could be defined as follows:

|  |
| --- |
| GET /getStudentDetails/?school\_id=25&student\_id=1500 |

The above is an implicit reference to a function named getStudentDetails() that takes as parameters school\_id and student\_id.

1.2 Resource Based Approach:

In resource based approach, the HTTP GET method would be defined as follows:

|  |
| --- |
| GET /school/25/student/1500 |

The above is a request to a unique resource and that resource is a student whose id is 1500 and enrolled with a school with id 25.

While both approaches are valid and would fetch the same information, the resource oriented approach is the only approach deemed RESTful. So we follow the Resource oriented approach.

The client would work with each resource based on CRUD (Create, Retrieve, Update, Delete) operations. The verbs provided by HTTP are associated in the following way with the CRUD operation set:

|  |  |
| --- | --- |
| **Operation** | **HTTP verb** |
| Get details about specific resource. | GET |
| Create resource on the server. The server automatically associates a reference (id) to the resource. | POST |
| Update a resource with the specified id or if a resource with that particular id does not exist, creates the resource with that id. | PUT |
| Delete a resource. | DELETE |

1.3 Content Type Negotiation:

The same API in cloud return a response in XML (or potentially JSON) format. So the format of the response is negotiated by including the preferred format in the “Accept:” field in the HTTP header.

A sample request is provided below:

|  |
| --- |
| GET /users HTTP/1.1  User-Agent: Tortoise/Android  Accept: application/xml  Accept-Charset: utf-8  Connection: close |

1. **FAULT HANDLING**

If any error occurs during the processing of a client request, the server sends back a fault which is indicated by a combination of:

1. Relevant HTTP Error code
2. Detailed description of the fault in XML (or JSON) format that is present in the body of HTTP response.

The following indicates the schema of an error response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <error>  <code> {Error code} </code>  <description> {Detail error description} </description>  <module> {Server module that raised the fault} <module>  </error> |

Common HTTP Error status codes:

|  |  |
| --- | --- |
| Status Code | Description |
| 200 | Operation successful (Success code, not a error) |
| 403 | Forbidden (Authentication failed) |
| 404 | Not found ( Resource that is being accessed does not exist) |
| 400 | Bad request |

All APIs mentioned in this document reports error status in the above format. In this document the HTTP status code as well as the parameters in the XML response will be indicated.

1. **USER AGENT FOR DEVICE/CLIENT**

Certain functionalities provided by the Tortoise cloud server depend on the value of the “User-Agent” field in the HTTP request header sent by client. Such functionalities manly pertain to services that are specific to a device.

The following “User-Agent” values MUST be used for client of the corresponding type:

|  |  |
| --- | --- |
| **Client** | **User-Agent String** |
| Mobile | Tortoise/Android-{version} |
| Web-browser | Tortoise/Browser-{version} |

1. **AUTHENTICATION AND SESSION MANAGEMENT**

Authentication and session management are provided through a single set of interfaces. The client requests the server to create a “session” resource by supplying the username and password. If successfully authenticated, the server returns authentication/session token which will need to be included with every request to the server.

|  |
| --- |
| NOTE:  There are several security consideration that need to be considered in the light of access by multiple devices. Those are not considered in this version of the document. |

4.1 Creation Of Session Object

Request:

|  |  |
| --- | --- |
| URI | /auth/session |
| Request Type | POST |
| Content Transfer | Application/x-www-form-urlencoded |
| Encoding | UTF-8 |
| User-Agent |  |
| Parameter | username={username}&password={password}&deviceid={deviceid} |

The ‘deviceid’ parameter is sent here is as optional \*if and only if\* provided with “username” and “password” parameters and vice-versa. The server will use combination of “deviceid” and User-Agent to uniquely identify the device. The “deviceid” must be registered first using the procedure mentioned in next section.

Sample response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <session>  <authtoken expiers=”{expiry}”> {Authentication token} </authtoken>  <user id=”789876” screenname=”Joe” />  </session> |

The id of the user is a unique id assigned by the server and will be used to reference a particular user. The client needs to save the id that is returned by this API.

Expiry time is absolute time expressed in seconds since UNIX epoch. If expiry time equal to 0, the token will not expire unless the session object is destroyed.

Error:

Invalid user credentials:

|  |  |
| --- | --- |
| HTTP Status Code | 403 |
| Error Code | 100 |
| Error Text | Incorrect Username or Password. |

Unable to create session:

|  |  |
| --- | --- |
| HTTP Status Code | 500 |
| Error Code | 102 |
| Error Text | Unable to create session. |

Invalid Session/Authentication token:

|  |  |
| --- | --- |
| HTTP Status Code | 403 |
| Error Code | 103 |
| Error Text | Invalid Session/Authentication token. |

User-Agent Not Found:

|  |  |
| --- | --- |
| HTTP Status Code | 500 |
| Error Code | 104 |
| Error Text | User-Agent not found. |

Unregistered Device:

|  |  |
| --- | --- |
| HTTP Status Code | 500 |
| Error Code | 105 |
| Error Text | Unregistered device. |

User-Agent Not Supported:

|  |  |
| --- | --- |
| HTTP Status Code | 500 |
| Error Code | 106 |
| Error Text | User-Agent not supported. |

4.2 Deletion of a Session Object

Request:

|  |  |
| --- | --- |
| URI | /auth/session |
| Request Type | DELETE |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Content Transfer Encoding | N/A |
| Parameters | None |

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <operation result=”success” /> |

Error:

Invalid Session/Authentication token:

|  |  |
| --- | --- |
| HTTP Status Code | 403 |
| Error Code | 103 |
| Error Text | Invalid Session/Authentication token. |

1. **USER PROFILE MANAGERMENT**

Users are considered as valid resources that are managed by the server and that are uniquely addressable.

5.1 Access User Information

Request:

|  |  |
| --- | --- |
| URI | /users/{user id} {or} /users/me |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | None |

“user id” is the id returned by the Session creation API. The client can also use the string “me” which will fetch the details of the user that the Authentication token is associated with.

Sample response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <user id=”789876”>  <userinfo>  <firstname> John </firstname>  <lastname> Smith </lastname>  <screenname> Joe </screenname>  <email> john.smith@gmail.com </email>  <phone> 09686829029 </phone>  <services>  <service type=”donar” attrib=”default”>  <name> {name of the service} </name>  <serviced> {service id } </serviceid>  </services>  <medicalhistory>  <dob format=”DD-MM-YYYY”> 01-01-1970 </dob>  <height unit=”centimetre”> 176 </height>  <weight unit=”kilogram”> 76 </weight>  <bloodpressure>  <date format=”DD-MM-YYYY”> </date>  <systole> </systool>  <diastole> </diastole>  </bloodpresure>  <Diabetic>  <date format=”DD-MM-YYYY”> </date>  <reading> </reading>  </Diabetic>  <Thyroid>  <date format=”DD-MM-YYYY”> </date>  <report>  <service type=”docs”>  <name> Google Docs </name>  <serviced> google </serviceid>  <location> {actual location} </location>  </uri>  </report>  </Thyroid>  </medicalhistory>  </userinfo>  </user> |

This list seems exhaustive and can be potential candidate to break up in multiple requests in later version on this document.

Error:

Invalid Session/Authentication token:

|  |  |
| --- | --- |
| HTTP Status Code | 403 |
| Error Code | 103 |
| Error Text | Invalid Session/Authentication token. |

Invalid user:

|  |  |
| --- | --- |
| HTTP Status Code | 404 |
| Error Code | 200 |
| Error String | Unser not found. |

Do not have credentials to access information pertaining to this user:

|  |  |
| --- | --- |
| HTTP Status Code | 403 |
| Error Code | 202 |
| Error String | Access forbidden. |

5.2 Obtain List of Services that the User has subscribed to

Request:

|  |  |
| --- | --- |
| URI | /users/{user id}/services {or} /users/me/services |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | None |

Sample response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <user id=”789876”>  <services>  <service type=”donar” attrib=”default”>  <name> {name of the service} </name>  <serviced> {service id } </serviceid>  </services>  </user> |

Error:

Invalid Session/Authentication token:

|  |  |
| --- | --- |
| HTTP Status Code | 403 |
| Error Code | 103 |
| Error Text | Invalid Session/Authentication token. |

Invalid user:

|  |  |
| --- | --- |
| HTTP Status Code | 404 |
| Error Code | 200 |
| Error String | Unser not found. |

Do not have credentials to access information pertaining to this user:

|  |  |
| --- | --- |
| HTTP Status Code | 403 |
| Error Code | 202 |
| Error String | Access forbidden. |

1. **CONTENT MANAGEMENT**

This section provides a set of APIs that provides the client with the information pertaining to the files that the server manages for the user. These could be of any types documents and images.

Files of specific types can be hosted and edited by only one specific third party service. For example, medical prescription can be hosted in either Google Docs or Microsoft Office Live but not both.

6.1 Authentication with 3rd Party Service (Such As Google)

When a call to a specific content management API is made, then server attempts to retrieve information from the third party service. For illustrative purposes let us assume that 3rd party service is Google. If authentication credentials with Google do not exist or if the validity of existing credentials have expired, then the server returns an error response as below:

|  |  |
| --- | --- |
| HTTP Status Code | 401 |
| Error Code | 700 |
| Error Text | Unable to authenticate with the subscribed services |
| Module Name | {Serviceid} |
| Auth Type | {client} or {oauth} |
| Oauth-url | url {parameter sent only in the case of oauth} |
| Oauth-callback | Callback URL {Parameter sent only in the case of oauth} |

The authentication mechanism with Google could either be client authentication in which case the Tortoise android client (web browser client) sends the username and password and uses the API and sends the authentication parameter to create a session object.

In the case of oauth, the mobile client opens the URL in a webview.

Ex:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <error>  <code> 700 </code>  <description> Unable to authenticate with subscribe service </description>  <module> Google <module>  <auth\_type> oauth </auth\_type>  <oauth\_url> <http://xxxxxxxxxxxx> </oauth\_url>  <oauth\_callback> <http://yyyyyyyyyyy> </oauth\_callback>  </error> |

Upon receiving the error response, the client shall prompt the user for credentials that the server can use authenticate with the 3rd party service.

6.2 Creation of a 3rd Party Session Object (Client Login)

Request:

|  |  |
| --- | --- |
| URI | /services/{serviceid}/session |
| Request Type | POST |
| Content Transfer | Application/x-www-form-urlencoded |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| User-Agent |  |
| Body | username={username}&password={password} |

This request has to be sent over HTTPS only. And the server will not cache the username and password. The session object that is created need not explicitly be destroyed.

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <operation result=”success” /> |

Error:

Invalid 3rd Party Authentication Credentials

|  |  |
| --- | --- |
| HTTP Status Code | 530 |
| Error Code | 700 |
| Error Text | Unable to authenticate with the subscribed service |

Invalid Session/Authentication token:

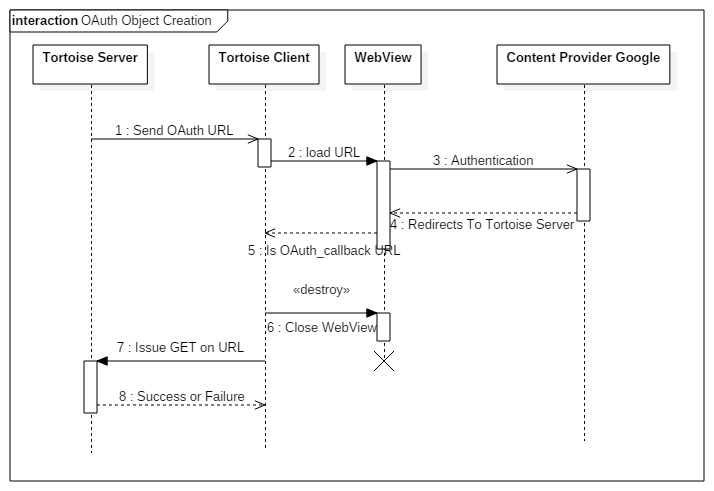
|  |  |
| --- | --- |
| HTTP Status Code | 403 |
| Error Code | 103 |
| Error Text | Invalid Session/Authentication token. |

6.3 Creation on 3rd Party Session Object (OAuth)

In the case of OAuth, the error message that is sent as described would indicate the Auth Type as oauth and pass the oauth url.

The mobile client should open a browser or webview instance and direct it to the outh url. After the user logs in to the third party service, he will be redirected back to a Tortoise service URL. This URL is provided to the client in the outh\_callback URL, then it should close the webview or browser instance and issue GET request to the URL by itself. The response to this GET request would be either a success as indicated by the success manages indication or would be an error.

The following sequence illustrates the oauth authentication process as will be followed for Tortoise server.



1. **CONTENT CREATION APIs**

Most of the use cases are content driven. For example, A-Z medicine, pharmacy services, health check up packages or even the invoice copy etc… These need either doctor’s prescriptions or recommended health test document. It would be recommended that Tortoise cloud server instead of maintaining its copy keep it with user’s cloud content provider like Google Docs, Drive, Dropbox etc… and make use of the metadata i.e. shared links instead. That way user’s data privacy would be maintained and Tortoise cloud server will not be burden with additional memory storage requirements. Below set of APIs outlines this with the help of Google Document service used in backend.

* 1. Get Supported Content Types.

Request:

|  |  |
| --- | --- |
| URI | /content/supportedcontent |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | None |

List the supported formats that user can upload or download.

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <content id=”{user id}”>  <type>  <extension> {doc | pdf | html | odt | bmp | gif | png} </extension>  <mimetype> {MIME type} </mimetype>  <description> {Format description } </description>  </type>  </content> |

* 1. Get Document List.

Request:

|  |  |
| --- | --- |
| URI | /content/docs/folders/{folder-id}/files |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | None |
| Comments | To query the root folder for the documents, just omit the folder id and just send /content/docs |

The user is identified based on the session/authentication token that is sent as a part of HTTP header request.

This API fetches document object only from the service that is marked as default service for the particular object type.

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <service serviceid=”google” userid=”47834”>  <documents xmlns=”<http://tortoise/content/docs>” folder\_id=”3923893”>  <doc>  <name> {document name }</name>  <type> {pdf|docx} </type>  <link> {URL} </link>  <id> { document id} </id>  </doc>  </documents>  <service> |

* 1. Get Folder List.

Request:

|  |  |
| --- | --- |
| URI | /content/docs/folders |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | None |

The user is identified based on the session/authentication token that is sent as part of the HTTP request.

This API fetches the document object only from the service that is marked as default service for the particular object type.

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <service serviceid=”google” userid=”47834”>  <folders xmlns=”<http://tortoise/content/docs>” folder\_id=”3923893”>  <folder>  <name> {document name }</name>  <id> { document id} </id>  </ folder>  </folders>  <service> |

* 1. Get Folder and Document List Together.

Request:

|  |  |
| --- | --- |
| URI | /content/docs/all |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | None |

The user is identified based on the session/authentication token that is sent as part of the HTTP request.

This API fetches the document object only from the service that is marked as default service for the particular object type.

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <service serviceid=”google” userid=”47834”>  <documents xmlns=”<http://tortoise/content/docs>” folder\_id=”3923893”>  <doc>  <name> {document name }</name>  <type> {pdf|docx} </type>  <link> {URL} </link>  <id> { document id} </id>  </doc>  </documents>  <folders xmlns=”<http://tortoise/content/docs>” folder\_id=”3923893”>  <folder>  <name> {document name }</name>  <id> { document id} </id>  </ folder>  </folders>  <service> |

* 1. Create A New Folder.

Request:

|  |  |
| --- | --- |
| URI | /content/folders |
| Request Type | POST |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | application/xml |
| Parameter | {Sample below} |

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <folder>  <name> {Folder Name} </name>  <contenttype> {Documents or Photos } </contenttype>  </folder> |

Based on \*contenttype\* that the folder is create in the appropriate service.

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <service serviceid=”google” userid=”{user-id}”>  <folder type=”{documents or photos}”>  <name> {Folder Name} </name>  <id> {Folder Id} </id>  </ folder>  </ service > |

* 1. Move Content To New Folder.

Request:

|  |  |
| --- | --- |
| URI | /content/{docs or photos}/file |
| Request Type | PUT |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | application/xml |
| Parameter | {Sample below} |

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <file>  <id> {file id} </id>  <sourcefolder> {Source folder ID} </ sourcefolder >  <targetfolder> {Target folder ID} </ targetfolder>  </folder> |

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <operation result =”success” / > |

* 1. Content upload

Request:

|  |  |
| --- | --- |
| URI | /content/file |
| Request Type | POST |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | Multipart/form-data; boundary=----uniqueboundary |
| Parameter | {Sample below} |

Content type either a document or an image can be sent and it will be uploaded by the server to the appropriate service of user’s choice.

The content uploaded using this API will be stored in a specific folder called “Tortoise Upload” in the appropriate service. In case of Google DOCs, a new folder with the aforementioned name will be created if it does not already exist. In case of photo hosting service, a new album will be created with the aforementioned name if t already does not exist.

The request is sent as multipart/form-data separated by defined boundary.

|  |
| --- |
| POST /content/file HTTP/1.1  User-Agent: Tortoise/Android  Accept: application/xml  X-Tortoise-Session: {Authentication Token}  Accept-Charset: utf-8  Connection: close  Content-Type: multipart/form-data; boundary=--uniqueboundary  Content-Length: 58024  ----uniqueboundary  Content-Disposition: form-data; name=”file”;filename=”sample.pdf”  Content-Type: application/pdf  {binary data goes here}  ----uniqueboundary |

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <service serviced=”google” userid=”{user id}”>  <doc>  <name> sample.pdf </ name >  <type> pdf </type>  <link> {URL for the uploaded file} </link>  <id> {document ID} </id >  </folder> |

* 1. Upload Content To A Specific Folder.

Request:

|  |  |
| --- | --- |
| URI | /content/folders/{folder\_id}/file |
| Request Type | POST |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | Multipart/form-data; boundary=----uniqueboundary |
| Parameter | {Sample below} |

Content type either a document or an image can be sent and it will be uploaded by the server to the appropriate service of user’s choice.

If a folder does not already exist, it first has to be created using the folder create API before using this API to upload content to the specific folder.

The request is sent as multipart/form-data separated by defined boundary.

|  |
| --- |
| POST /content/folders/{folder-id}/file HTTP/1.1  User-Agent: Tortoise/Android  Accept: application/xml  X-Tortoise-Session: {Authentication Token}  Accept-Charset: utf-8  Connection: close  Content-Type: multipart/form-data; boundary=--uniqueboundary  Content-Length: 58024  ----uniqueboundary  Content-Disposition: form-data; name=”file”;filename=”sample.pdf”  Content-Type: application/pdf  {binary data goes here}  ----uniqueboundary |

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <service serviced=”google” userid=”{user id}”>  <doc>  <name> sample.pdf </ name >  <type> pdf </type>  <link> {URL for the uploaded file} </link>  <id> {document ID} </id >  </folder> |

* 1. Sample Error Response For All Content APIs

Invalid Session/Authentication Token:

|  |  |
| --- | --- |
| HTTP Status Code | 403 |
| Error Code | 103 |
| Error Text | Invalid Session/Authentication Token |

If user is not having 3rd party services authentication token at Tortoise Cloud server:

|  |  |
| --- | --- |
| HTTP Status Code | 403 |
| Error Code | 700 |
| Error Text | Unable to authenticate with subscribed services |

In case request XML is not proper:

|  |  |
| --- | --- |
| HTTP Status Code | 400 |
| Error Code | 402 |
| Error Text | XML error |

If new folder or document is not created because of 3rd party server error:

|  |  |
| --- | --- |
| HTTP Status Code | 503 |
| Error Code | 710, 711, 712 |
| Error Text | Unable to create document. |

If content list is empty:

|  |  |
| --- | --- |
| HTTP Status Code | 500 |
| Error Code | 601, 602, 603 |
| Error Text | Document | Folder | Photo list is empty. |

1. **APIs PER SCREEN**

8.1 Dashboard

Request:

|  |  |
| --- | --- |
| URI | /services/dashboard/list |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | user={user id} |

Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <services language=”en-us”>  <service type=”{service type}” attrib=”default”>  <name> {name of the service} </name>  <serviced> {service id } </serviceid>  <icon> {icon URL} <icon>  <label> {Service Label} </label>  <pending-notice> {Count} </pending-notice>  <service>  </services> |

Dashboard listings are dynamic and can vary based on user subscription. For example, “Donar” service may not be available for all users, based on subscribed package one can be avail this service.

Furthermore, in future the service list may be increase and modify. Client application shall not hard code it, instead templatise so that list can be generic, scalable can be adopted for other solutions.

8.2 Child Vaccination

This API fetches the “Child Vaccination” information listing based on the information passed in as parameter.

Request:

|  |  |
| --- | --- |
| URI | /services/{service id}/vaccination |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | location={location}&age={age} |

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <service type=”{service type}” attrib=”default”>  <name> {name of the service} </name>  <serviced> {service id } </serviceid>  <age-group> {age-group} </age-group>  <vaccination-chart> {URL for vaccination chart} </vaccination-chart>  <packages>  < package type=”standard”>  <name> { package name} </name>  <packageid> {package id} </packageid>  <description> {Description for this package} </description>  </package>  < package type=”premium”>  <name> { package name} </name>  <packageid> {package id} </packageid>  <description> {Description for this package} </description>  </package>  < package type=”highvalued”>  <name> { package name} </name>  <packageid> {package id} </packageid>  <description> {Description for this package} </description>  </package>  </packages>  </service> |

8.2.1 Child Vaccination Package Selection

Request:

|  |  |
| --- | --- |
| URI | /services/{service id}/packages/{packageid} |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | age={age-group}&filter={ brand | name | price | place | pincode } |

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <service type=”{service type}” attrib=”default”>  <name> {name of the service} </name>  <serviced> {service id } </serviceid>  <age-group> {age-group} </age-group>  <filter> {brand | name | price | place | pincode} </filter>  < package type=”{standard | premium | highvalue }”>  <name> { package name} </name>  <packageid> {package id} </packageid>  </package>  <vendors>  <vendor>  <name> {Vendor Name} </name>  <vendorid> { vendors id} </vendorid>  <address> { vendor’s Address } </address>  <price> {Package Price} </price>  <offer> {Offer details} </offer>  <payment-mode> {Card | COD} </payment-mode>  <user-rating total-reviews=”{total review count}” rate-scale=”{scale}”> {User Rating} </user-rating>  <reviews>  <review>  <reviewer>  <name> {Name of user | anonymous } </name>  <userid> {User id} </userid>  <text> {Review in 500 words } </text>  <rating rate-scale=”{scale}”> {rating} </rating>  </reviewer>  </review>  </reviews>  </vendor>  </vendors>  </service> |

8.2.2 Child Vaccination Package Booking

Request:

|  |  |
| --- | --- |
| URI | /services/{serviceid}/packages/{packageid}/vendors/{vendorid}/purchase |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | user={user-id}&payment-mode={card | cod} |

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <operation result=”success” /> |

8.3 A-Z medicine

This API fetches the “A-Z Medicine” information listing based on the information passed in as parameter.

Request:

|  |  |
| --- | --- |
| URI | /services/{serviceid}/medicines/{category-id} |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | {Sample As Below} |

The user is identified based on the session/authentication token that is sent as a part of HTTP header request.

This API fetches document object only from the service that is marked as default service for the particular object type.

Prescription details are provided in document uploaded to 3rd party server (Google Docs in below example). Furthermore, name, brand and prescription fields are optional here as prescription document is uploaded already to “Tortoise Upload” folder or folder visible to cloud server.

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <medicine>  <name> {medicine-name} </name>  <brand> {brand} </brand>  <prescribed-by> {doctor} </ prescribed-by>  <service serviceid=”google” userid=”47834”>  <documents xmlns=”<http://tortoise/content/docs>” folder\_id=”3923893”>  <doc>  <name> {document name } </name>  <type> {pdf|docx} </type>  <link> {URL} </link>  <id> { document id} </id>  </doc>  </documents>  </medicine> |

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <service type=”{service type}” attrib=”default”>  <name> {name of the service} </name>  <serviced> {service id } </serviceid>  <medicines>  <category> { Broad Category } </category>  <categoryid> { Category id } </categoryid>  <name> {name of medicine} </name>  <brand> {name of the phrama brand </brand>  <description> {Description of medicine} </description>  <doses> {recommended-doses} </doses>  <prescribed-by> {name of doctor} </prescribed-by>  </medicines>  <vendors>  <vendor>  <name> {Vendor Name} </name>  <vendorid> { vendors id} </vendorid>  <address> { vendor’s Address } </address>  <price> {Package Price} </price>  <offer> {Offer details} </offer>  <payment-mode> {Card | COD} </payment-mode>  <user-rating total-reviews=”{total review count}” rate-scale=”{scale}”> {User Rating} </user-rating>  <reviews>  <review>  <reviewer>  <name> {Name of user | anonymous } </name>  <userid> {User id} </userid>  <text> {Review in 500 words } </text>  <rating rate-scale=”{scale}”> {rating} </rating>  </reviewer>  </review>  </reviews>  </vendor>  </vendors>  </service> |

Errors:

Invalid Session/Authentication Token:

|  |  |
| --- | --- |
| HTTP Status Code | 403 |
| Error Code | 103 |
| Error Text | Invalid Session/Authentication Token |

If user is not having 3rd party services authentication token at Tortoise Cloud server:

|  |  |
| --- | --- |
| HTTP Status Code | 403 |
| Error Code | 700 |
| Error Text | Unable to authenticate with subscribed services |

In case request XML is not proper:

|  |  |
| --- | --- |
| HTTP Status Code | 400 |
| Error Code | 402 |
| Error Text | XML error |

If document is not available on 3rd party server error:

|  |  |
| --- | --- |
| HTTP Status Code | 404 |
| Error Code | 713 |
| Error Text | Document not found. |

8.4 Bloodbank db

This API fetches the “Blood bank Database” information listing based on the location information passed in as parameter. We can set different filters like place, brand, price and popularity.

Request:

|  |  |
| --- | --- |
| URI | /services/{serviceid}/bloodbanks/location/{location-details} |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | filter={place | brand | price | popularity}&pin={pincode}&blood-group={blood-group}&RH={+ | -} |

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <service type=”{service type}” attrib=”default”>  <name> {name of the service} </name>  <serviced> {service id} </serviceid>  <filter> { place | brand | price | popularity} </filter>  <bloodgroup> { blood group } </bloodgroup>  <vendors>  <vendor>  <name> {Vendor Name} </name>  <vendorid> { vendors id} </vendorid>  <address> { vendor’s Address } </address>  <price> {Package Price} </price>  <offer> {Offer details} </offer>  <payment-mode> {Card | Cash} </payment-mode>  <user-rating total-reviews=”{total review count}” rate-scale=”{scale}”> {User Rating} </user-rating>  <reviews>  <review>  <reviewer>  <name> {Name of user | anonymous } </name>  <userid> {User id} </userid>  <text> {Review in 500 words } </text>  <rating rate-scale=”{scale}”> {rating} </rating>  </reviewer>  </review>  </reviews>  </vendor>  </vendors>  </service> |

8.5 Health Checkup

This API fetches the “Health Checkup” listing based on the location information passed in as parameter.

Request:

|  |  |
| --- | --- |
| URI | /services/{serviceid}/healthcheckup/all |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | filter={ price | place | city | hospital | popularity}&pin={pincode} |

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <service type=”{service type}” attrib=”default”>  <name> {name of the service} </name>  <serviced> {service id} </serviceid>  <filter> { price | place | city | hospital | popularity} </filter>  <vendors>  <vendor>  <name> {Vendor Name} </name>  <vendorid> { vendors id} </vendorid>  <address> { vendor’s Address } </address>  <price> {Package Price} </price>  <offer> {Offer details} </offer>  <payment-mode> {Card | Cash} </payment-mode>  <user-rating total-reviews=”{total review count}” rate-scale=”{scale}”> {User Rating} </user-rating>  <reviews>  <review>  <reviewer>  <name> {Name of user | anonymous } </name>  <userid> {User id} </userid>  <text> {Review in 500 words } </text>  <rating rate-scale=”{scale}”> {rating} </rating>  </reviewer>  </review>  </reviews>  </vendor>  </vendors>  </service> |

8.6 Health Listing

This API fetches the “Health Listing” information listing based on the location information passed in as parameter. We can set different filters like brand, place, city, pincode, name, department and popularity.

8.6.1 Hospital listings

Request:

|  |  |
| --- | --- |
| URI | /services/{serviceid}/healthlistings/hospitals |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | pin={pincode}&filter={location| Popularity | brand} |

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <service type=”{service type}” attrib=”default”>  <name> {name of the service} </name>  <serviced> {service id} </serviceid>  <location> {location details} </location>  <hospitals>  <hospital>  <name> {Hospital Name} </name>  <id> { Hospital id} </id>  <address> { Hospital’s Address } </address>  <description> {description in 500 words. } </description>  <image> {Hospital image URL. } </image>  <user-rating total-reviews=”{total review count}” rate-scale=”{scale}”> {User Rating} </user-rating>  <reviews>  <review>  <reviewer>  <name> {Name of user | anonymous } </name>  <userid> {User id} </userid>  <text> {Review in 500 words } </text>  <rating rate-scale=”{scale}”> {rating} </rating>  </reviewer>  </review>  </reviews>  </hospital>  </hospitals>  </service> |

8.6.2 Pharmacy listings

Request:

|  |  |
| --- | --- |
| URI | /services/{serviceid}/healthlistings/pharmacies |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | pin={pincode}&filter={location| Popularity | brand} |

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <service type=”{service type}” attrib=”default”>  <name> {name of the service} </name>  <serviced> {service id} </serviceid>  <location> {location details} </location>  < pharmacies>  < pharmacy>  <name> { pharmacy Name} </name>  <id> { pharmacy id} </id>  <address> { pharmacy’s Address } </address>  <description> {description in 500 words. } </description>  <image> { pharmacy image URL. } </image>  <filter> {location | popularity | brand} </filter>  <user-rating total-reviews=”{total review count}” rate-scale=”{scale}”> {User Rating} </user-rating>  <reviews>  <review>  <reviewer>  <name> {Name of user | anonymous } </name>  <userid> {User id} </userid>  <text> {Review in 500 words } </text>  <rating rate-scale=”{scale}”> {rating} </rating>  </reviewer>  </review>  </reviews>  </ pharmacy >  </ pharmacies>  </service> |

8.6.3 Physician listings

Request:

|  |  |
| --- | --- |
| URI | /services/{serviceid}/healthlistings/physicians |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | Category={category}&pin={pincode}&filter={location| Popularity} |

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <service type=”{service type}” attrib=”default”>  <name> {name of the service} </name>  <serviced> {service id} </serviceid>  <location> {location details} </location>  <physicians>  <physician>  <name> { physician Name} </name>  <id> { physician id} </id>  <address> { physician’s Address } </address>  <description> {description in 500 words. } </description>  <image> { physician image URL. } </image>  <filter> {location | popularity | brand} </filter>  <user-rating total-reviews=”{total review count}” rate-scale=”{scale}”> {User Rating} </user-rating>  <reviews>  <review>  <reviewer>  <name> {Name of user | anonymous } </name>  <userid> {User id} </userid>  <text> {Review in 500 words } </text>  <rating rate-scale=”{scale}”> {rating} </rating>  </reviewer>  </review>  </reviews>  </ physician>  </ physicians >  </service> |

8.6.4 Getting it all together

Request:

|  |  |
| --- | --- |
| URI | /services/{serviceid}/healthlistings/all |
| Request Type | GET |
| Header Param | X-Tortoise-Session: {Authentication Token} |
| Encoding | UTF-8 |
| Content Transfer Encoding | N/A |
| Parameter | pin={pincode}&filter={location| Popularity} |

Sample Response:

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8” ?>  <service type=”{service type}” attrib=”default”>  <name> {name of the service} </name>  <serviced> {service id} </serviceid>  <location> {location details} </location>  < physicians>  < physician>  <name> { physician Name} </name>  <id> { physician id} </id>  <address> { physician’s Address } </address>  <description> {description in 500 words. } </description>  <image> { physician image URL. } </image>  <filter> {location | popularity | brand} </filter>  <user-rating total-reviews=”{total review count}” rate-scale=”{scale}”> {User Rating} </user-rating>  <reviews>  <review>  <reviewer>  <name> {Name of user | anonymous } </name>  <userid> {User id} </userid>  <text> {Review in 500 words } </text>  <rating rate-scale=”{scale}”> {rating} </rating>  </reviewer>  </review>  </reviews>  </physician>  </ physicians>  < pharmacies>  < pharmacy>  <name> { pharmacy Name} </name>  <id> { pharmacy id} </id>  <address> { pharmacy’s Address } </address>  <description> {description in 500 words. } </description>  <image> { pharmacy image URL. } </image>  <filter> {location | popularity | brand} </filter>  <user-rating total-reviews=”{total review count}” rate-scale=”{scale}”> {User Rating} </user-rating>  <reviews>  <review>  <reviewer>  <name> {Name of user | anonymous } </name>  <userid> {User id} </userid>  <text> {Review in 500 words } </text>  <rating rate-scale=”{scale}”> {rating} </rating>  </reviewer>  </review>  </reviews>  </ pharmacy >  </ pharmacies>  <hospitals>  <hospital>  <name> {Hospital Name} </name>  <id> { Hospital id} </id>  <address> { Hospital’s Address } </address>  <description> {description in 500 words. } </description>  <image> {Hospital image URL. } </image>  <user-rating total-reviews=”{total review count}” rate-scale=”{scale}”> {User Rating} </user-rating>  <reviews>  <review>  <reviewer>  <name> {Name of user | anonymous } </name>  <userid> {User id} </userid>  <text> {Review in 500 words } </text>  <rating rate-scale=”{scale}”> {rating} </rating>  </reviewer>  </review>  </reviews>  </hospital>  </hospitals>  </service> |

Errors:

Invalid Session/Authentication token:

|  |  |
| --- | --- |
| HTTP Status Code | 403 |
| Error Code | 103 |
| Error Text | Invalid Session/Authentication token. |

TO-DO List

1. health\_products
2. offers
3. pharmacy\_services
4. premium\_services
5. reminders
6. tele\_nutrisionist
7. Search