

August 13, 2014

**Developer Resource Redesign - Backend  
Integration  
Requirements Specification DRAFT  
Task Order 6.28**

Version 1.4



**Prepared by:**

Leidos



1000 Broadway, Suite 680, Oakland, CA 94607

**Prepared for:**

Metropolitan Transportation Commission



101 8<sup>th</sup> Street, Oakland, CA94607

# Requirements Specification – Version 1.4

---

## Revision Chart

Document Description	Date	Version Number
First Internal Draft	10/15/2013	1.0
Updated as per comments from MTC	03/28/2014	1.1
Added requirements for Transit APIs	05/23/2014	1.2
Updated requirements as per comments from MTC	8/13/2014	1.4

# Requirements Specification – Version 1.4

---

## Contents

Revision Chart .....	2
Purpose and Scope of Requirements.....	4
1. API Requirements .....	5
1.1. Facilitate User Signups/Requests .....	5
1.2. Facilitate User Verification after Signups/Requests .....	6
1.3. Facilitate API User Account Management.....	6
1.4. Facilitate API User API/tool selections provided by 511 .....	7
1.5. Facilitate Admin Level functionalities.....	7
1.6. Facilitate Communication between backend systems across modes. ....	9
1.7. Facilitate Access of Transit GTFS APIs .....	10
2. System Requirements .....	12
2.1. Access Control .....	12
2.2. General .....	12
2.3. Testing.....	13
3. Transit Clean Interface Package delivery.....	14
4. Attachment 1 – Glossary .....	14

## Purpose and Scope of Requirements

The purpose of this document is to communicate the requirements associated with backend integration and coordination efforts to be performed during the redesign of the developer resource section (also referred as developer portal) of 511.org. The backend integration for the developer portal has been divided into three phases, based on the type of data/tools and the required user agreements between MTC and end-users. Phase I will include backend integration for Data APIs (bulk/adhoc) signups (requests) and will allow the user to manage their API account and choices, and will allow API admins the ability to manually or automatically send API/data updates and to communicate with developers. Phase II will include backend integration to record and track users who download widget type tools such as the 511 Transit Clean Interface and Regional Rideshare Carpool Calculator. This second phase will allow admins to notify users of any future updates to these widgets. Phase III will include backend integration to record and facilitate user signups/requests for those who are interested in custom interfaces such as a branded 511 Traffic map or the 511 RideMatch Service, as well as admin communication with users for initial set up and update of these services.

The backend integration and coordination effort will entail creating new APIs that can facilitate communication between the front-facing developer portal, which will be developed by the 511 web contractor, and the backend database and 511 modal contractors providing APIs/data/tools. In addition those APIs will work as gateway/proxy to Transit.511 APIs/data/tools serving as a single access point to all 511 data. Most of the work will be performed by reusing components and services that were developed under Open511 API development task. The requirements have been broken down into two core groups, listed below.

- API Requirements - These cover requirements for the APIs that will be required to support various use cases such as user signups/requests and notifications.

# Requirements Specification – Version 1.4

---

- System Requirements – These cover requirements for how these APIs shall be developed and deployed, the type of database and development frameworks to be used, and any security measures that need to be followed.

## 1. API Requirements

### 1.1. *Facilitate User Signups/Requests*

- 1.1.1.1. The system shall expose an API over HTTP/HTTPS that will allow 511.org to submit user registration information. The API shall accept the following information:
  - 1.1.1.2. First name
  - 1.1.1.3. Last name
  - 1.1.1.4. Company/Organization
  - 1.1.1.5. Phone number
  - 1.1.1.6. Email address
  - 1.1.1.7. Password
- 1.1.2. Upon successful submission, the API shall create a new user under 'unverified' status in the API database. The API shall then respond back with appropriate messages/code that can be used by 511.org to display whether the registration was successful or not.
- 1.1.3. The system shall create a TOMS user account (if requested) using management APIs provided by Sonic JMS and then preserve them into the centralized database.
- 1.1.4. Internally, the system shall send a verification email to the email address used by the user during signup. The email shall contain a link pointing to UI to be used for verification.
- 1.1.5. On error or exceptions, the API shall return an appropriate error message.

### **1.2. Facilitate User Verification after Signups/Requests**

- 1.2.1. The system shall expose an API over HTTP/HTTPS that will allow the UI to submit a verification request after users open the link sent to their personal email and verify themselves.
- 1.2.2. Upon successful verification, the API shall change the user status in the database from an unverified to an active state. The API shall then respond back with appropriate messages/code that can be used by 511.org to redirect user to a page on 511.org showing a successful verification.
- 1.2.3. On error or exceptions, the API shall return an appropriate error message.

### **1.3. Facilitate API User Account Management**

- 1.3.1. The system shall expose an API over HTTP/HTTPS that will allow 511.org to request details about a user which can then be used by 511.org to build and display a user account/edit screen for returning users upon successful login. API shall return following information:
  - 1.3.1.1. First name
  - 1.3.1.2. Last name
  - 1.3.1.3. Company/Organization
  - 1.3.1.4. Phone number
  - 1.3.1.5. Email address
- 1.3.2. The API shall also provide details related to each API(s)/tool(s) requested (registered to use) by the user. The details shall also include current quota usage for each API, where applicable, along with quota and threshold limits.
- 1.3.3. The system shall expose an API over HTTP/HTTPS that will allow 511.org to submit updates to personal/individual information on the user's behalf to the backend database.

## Requirements Specification – Version 1.4

---

- 1.3.4. The system shall expose an API over HTTP that will allow 511.org to submit requests to delete a user account on the user's behalf to the backend database.
- 1.3.5. Upon successful submission, the API shall respond back with user information that can be shown on 511.org.
- 1.3.6. On error or exceptions, the API shall return an appropriate error message.

### **1.4. Facilitate API User API/tool selections provided by 511**

- 1.4.1. The system shall expose an API over HTTP/HTTPS that will allow 511.org to list all the tools and APIs/bulk data that are currently available from 511.
- 1.4.2. The system shall expose an API over HTTP/HTTPS that will allow 511.org to submit user selections of tools and APIs/bulk data to the backend database.
- 1.4.3. The API shall also record acceptance of terms and conditions by the user into the backend database as requested by UI. Upon successful submission, the API shall respond back with user information that can be shown on 511.org.
- 1.4.4. On error or exceptions, the API shall return an appropriate error message.

### **1.5. Facilitate Admin Level functionalities**

- 1.5.1. The system shall expose an API over HTTP/HTTPS that will allow UI to search for API/tool/bulk data users either by email, first name, last name, status, or token. This API will be used by an admin interface to look up user information from 511.org. The API shall accept the following parameters in any combinations:
  - 1.5.1.1. User email address

## Requirements Specification – Version 1.4

---

- 1.5.1.2. User first name
- 1.5.1.3. User last name
- 1.5.1.4. User status
- 1.5.1.5. User token
- 1.5.2. The system shall expose an API over HTTP/HTTPS to allow UI to notify users about changes to API/tool/Bulk data. This API will be used by 511.org to send email messages to users. Each message shall be saved to the database as email templates which can be retrieved later to edit and create new messages. API shall accept following parameters:
  - 1.5.2.1. Previously saved email template ID (optional)
  - 1.5.2.2. New email template
  - 1.5.2.3. List of applications/data/widgets/APIs affected (only users subscribed to those services will be notified)
- 1.5.3. The System shall expose an API over HTTP/HTTPS to list all email templates, retrieve and update previously created email templates that were used in sending messages to users.
- 1.5.4. Admin users shall be able to gather the following statistics from the API database:
  - 1.5.4.1. Number of verified and unverified users in a given time period (day, week and month).
  - 1.5.4.2. Number of user signups by API type and sub types, for example number of users that have signed up for SIRI or GTFS-RT API in transit in a given time period (day, week and month).
  - 1.5.4.3. Number of active/inactive (not used API in last 30 days) users in a given time period (day, week and month)
  - 1.5.4.4. Date of registration and login/API access history by each account.
- 1.5.5. On error or exceptions, the API shall return an appropriate error message.

### **1.6. Facilitate Communication between backend systems across modes.**

1.6.1. The system shall expose an API to save/update API/tool specific information into a centralized database. This API will be used either by an automated tool and/or will be exposed via a UI to allow API admins to save information. This API shall accept following parameters:

- 1.6.1.1. Public API/tool/widget ID
- 1.6.1.2. Type of credentials (token, username/password, etc.)
- 1.6.1.3. Connection details (textual description)

*For example, in the case of the TOMS data, this API can be used by the fulfillment tool that creates the required connection details in JMS and then saves those connection details to the centralized database. Similarly, this API can be exposed via a web form to the Rideshare team and or a prospective employer to enter details that are specific to the RideMatch service. After a prospective employer has been entered into the system ether by the Rideshare team or an employer, the system shall send an alert to the RRP that there is a new entry/customer. The Rideshare team can also use this form to add additional details such as the custom ridematch URL for the employer after it has been configured.*

1.6.2. The system shall provide an HTTP/HTTPS based API to save/update API access options provided by each modal system to the centralized database. For example, Transit will require a mechanism to list all the available agencies for which 511.org can provide data to end users. This API shall accept following parameters:

- 1.6.2.1. Public API/tool/widget ID
- 1.6.2.2. Public API/tool/widget access option ID
- 1.6.2.3. Public API/tool/widget access option name
- 1.6.2.4. Public API/tool/widget access option description

## Requirements Specification – Version 1.4

---

1.6.2.5. Public API/tool/widget specific option properties

*Note: this functionality will be used for GTFS dataset URL storage using two character agency IDs and agency specific Transit API endpoints (in addition to other functionalities)*

1.6.3. The system shall provide an HTTP/HTTPS based API to validate a token and get user information upon verification. This API can be used by modal systems to obtain information about data sets/tools the user has signed up for and then restrict access to only those data and tools. This API shall provide additional verification functionality for Transit GTFS packages requests. In addition to the token verification it shall validate that the user has access to the requested agency and also the quota and throttle limits, if any. Upon successful validation, it shall pass on the download request to Transit.

1.6.4. On error or exceptions, the API shall return the appropriate error message.

### **1.7. Facilitate Access to Transit GTFS APIs**

1.7.1. The system shall provide an HTTP/HTTPS based API that will serve as a gateway/proxy while accessing Transit.511 APIs (GTFS and Open511).

1.7.2. Gateway/proxy API shall perform authentication/authorization against common API user database before forwarding request to Transit APIs

1.7.3. Gateway/proxy API shall enforce quotas and throttling configured in common user database

1.7.4. Gateway/proxy will forward user requests that are authenticated and validated against common user database and passed quota/throttling check to Transit.511 APIs for fulfillment. Responses from Transit.511 APIs shall be passed back to the users/requestors.

1.7.5. To enable the user to select the transit agencies that the user is interested in, Transit system shall provide a HTTP/HTTPS based API that

## Requirements Specification – Version 1.4

---

returns the list of active agencies that have downloadable GTFS datasets. This API shall return the agency's two character ID, agency name, and dedicated API endpoint URLs. There may be additional information returned based on the design of the user interface for this functionality. This API shall be used to enable the front-end UI to display the list of active transit agencies with available GTFS datasets.

- 1.7.6. URL endpoints to the GTFS dataset shall be routed internally through the Gateway and will not be exposed at the public level.

*Note: This is to ensure that users retrieve GTFS datasets via the 511.org developer resource portal while applying validation and throttling requests as needed.*

- 1.7.7. A notification service in the Transit system shall run as a background process to notify the gateway whenever a GTFS dataset is generated. The notification service shall call a gateway/proxy API notification endpoint that shall accept the two character agency ID and the date and time the GTFS dataset was generated along with email template ID.

*Note: GTFS datasets are typically generated for every agency sign-up (service change). GTFS datasets may also be generated for other updates such as interim sign-up changes and holiday updates.*

- 1.7.8. The following Open511 Transit APIs shall be accessible through the gateway/proxy API:

- Operator
- Line
- Stop
- Stop Place
- Pattern
- Timetable
- Holiday
- Transit Scheduled Departures for a stop

- Transit Announcement
- General Announcement

**2. Note: The 511 Transit team will provide the internal URLs for all the APIs to the 511 Traffic contractor. The Traffic contractor will integrate each API with its public endpoint (URL). The request will be validated by the gateway/proxy and forwarded to the Transit system if valid. System Requirements**

### **2.1. Access Control**

- 2.1.1. The API shall not allow access to any APIs/tools if the user has not accepted the terms and conditions.
- 2.1.2. The API shall not allow access to any APIs/tools if the user has not been verified.
- 2.1.3. The API shall not fulfill requests from users who have reached their quota or throttling limits and shall return the appropriate error message.
- 2.1.4. All user profile/properties/parameters related communications shall use HTTPS/SSL based communications.
- 2.1.5. None of the user profile/properties/parameters related API endpoints can be accessed from outside the 511 system network.
- 2.1.6. Upon user login the API shall update the user's last login date/time stored in the backend database.

### **2.2. General**

- 2.2.1. The data exchange format between systems shall be JSON based.
- 2.2.2. These APIs shall reuse the same set of servers (Web and Database) as used by the other Traffic.511 applications.

## Requirements Specification – Version 1.4

---

- 2.2.3. During errors or exceptions, the APIs shall respond with an appropriate message that will allow applications consuming these APIs to identify that an error has occurred and the application can then display an appropriate error message to the user.
- 2.2.4. When possible the API shall capture and log all fatal errors during processing of an API request. Additionally email notifications shall be sent out to admins (system administrators and leads) when the API system is detected in a non-operational state. Typical checks would include expecting a HTTP 200 response on HTTP Get operations.
- 2.2.5. API shall use external web analytics system (Google Analytics or similar) for usage tracking.
- 2.2.6. API shall implement number of API request limitations (quotas) on per user basis (each user has individual quota associated). Quotas shall be configurable and updated as needed.
- 2.2.7. API shall implement frequency of API requests limitations (throttling) on per user basis (each user has individual quota associated). Limits shall be configurable and updated as needed.

### **2.3. Testing**

- 2.3.1. These APIs shall be provided with test plans and test procedures for integration cases and the system acceptance test to ensure that each test is comprehensive and verifies all the features of the function to be tested, including interaction with the User Interface on 511.org.
- 2.3.2. Each test procedure shall list the objective of the testing and the specific Developer Resources portal backend system requirement(s) that are being verified along with pass/fail criterion for each.
- 2.3.3. Test procedures shall include the following items:
  - 2.3.3.1. Function(s) to be tested and corresponding requirement(s);
  - 2.3.3.2. Set-up and conditions for testing including ambient conditions;

## Requirements Specification – Version 1.4

---

- 2.3.3.3. Step-by-step procedures to be followed;
- 2.3.3.4. Pass/Fail criterion for each requirement tested including measurement tolerances;
- 2.3.3.5. All inputs and expected results outputs for each test segment;
- 2.3.3.6. Descriptions of all simulation tools and techniques used during the test.

### 3. Transit Clean Interface Package delivery

- 3.1.1. The Agency and Business versions of the Transit Clean Interface packages along with the user guides will be packaged and delivered to the 511 web contractor. Any future changes to clean interface and the user guides shall be handled by the 511 web contractor.

### 4. Attachment 1 – Glossary

Terms	Abbreviations	Definition
Application Programming Interface	API	A set of functions or routines which <a href="#">specifies</a> how <a href="#">software components</a> should interact with each other.
General Transit Feed Specification	GTFS	Defines a common format for public transportation schedules and associated geographic information
General Transit Feed Specification – Real Time	GTFS-RT	Extension to GTFS that allows public transportation agencies to provide realtime updates about their fleet
Hyper Text Transfer Protocol	HTTP	HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various HTTP commands.

## Requirements Specification – Version 1.4

<b>Hypertext Transfer Protocol Secure</b>	<b>HTTPS</b>	A communications protocol for secure communication over a computer network
<b>Java Message Service</b>	<b>JMS</b>	An API for accessing enterprise messaging systems from Java programs
<b>JavaScript Object Notation</b>	<b>JSON</b>	A lightweight data-interchange format.
<b>Proxy</b>		A computer system or an application that acts as an intermediary for requests from clients seeking resources from other servers
<b>Representational state transfer</b>	<b>REST</b>	A simple stateless architecture useful in building APIs over HTTP.
<b>Service Interface for Real Time Information</b>	<b>SIRI</b>	An XML standard covering a wide range of types of real-time information for public transportation
<b>Secure Sockets Layer</b>	<b>SSL</b>	Cryptographic protocols designed to provide communication security over the Internet
<b>Traffic Open Messaging Service</b>	<b>TOMS</b>	The free 511 Traffic Data Feed provides incidents, speed, and travel time data for individual links on the highways, freeways, and expressways in the San Francisco nine-county Bay Area.
<b>User Interface</b>	<b>UI</b>	Space where interactions between humans and machines occur
<b>Uniform Resource Locator</b>	<b>URL</b>	Specific character string that constitutes a reference to a resource