TransLink OPIA API – v2.0

Overview

TRANSLink

Queensland Government

Document control sheet

Version history

Version no.	Date	Nature of amendment
0.1	2012/06/26	Created document.
0.2	2012/08/10	Added real-time service information; minor clarifications to other services
0.3	2012/10/04	Updated caching section with additional guidance Updated endpoint security details
0.4	2012/10/22	Updated many sections with REST endpoint details Removed restriction on consuming API directly from mobile devices, now a recommendation
0.5	2013/05/30	Removed references to real-time API service, customers can access this data via the GTFS Real-Time feed instead Add Data Formats section
1.0	2013/06/13	Updated REST endpoints and added information about Swagger Reorganized a few sections Removed references to Service Notices feed, which is moving between systems and would soon become obsolete Replace TransLink branding with Dept Transport & Main Roads branding
1.1	2013/07/13	Updated some endpoint URL’s
1.2	2013/08/01	Added Terms of Use Clarified how to contact us
2.0	2015/03/03	Amendments for version 2 of the API; support for regional stops and services.

Table of Contents

1. Overview

In 2010/2011, TransLink helped passengers make more than 178 million journeys.

We also helped more than 31 million customers find travel information, service updates and journey plans via our web and mobile sites.

While these are great results, we think we can do even better by exposing our data to a global network of web and app developers.

With the introduction of TransLink's Online Passenger Information Application's API ("OPIA API"), external developers are able to integrate TransLink data directly into their own web sites and applications – from a known source of truth – without resorting to the cumbersome, error-prone methods such as data scraping that have been used in the past.

This document provides an overview of the new API and explains how developers can integrate its data with their systems.

2. Target audience

This document is moderately technical and is targeted at System Architects and Software Developers who wish to consume information about services operating on the TransLink network.

3. Terms of use

By accessing, viewing, relying upon or otherwise utilizing the TransLink OPIA API you accept the terms and conditions set forth at http://translink.com.au/open-data/terms-and-conditions and agree to be bound by them.

If these terms and conditions are inconsistent with any other product or service-specific notice, the provisions of the specific notice apply to the extent of the inconsistency.

4. Getting started

4.1 Summary

This section provides an overview of the features and design of TransLink's OPIA API.

4.2 Pre-requisite skills and knowledge

Readers are expected to have an understanding of web service technologies and the challenges of integrating heterogeneous applications over a high latency network (i.e. the Internet).

4.3 Developer pre-requisites

Developers consuming TransLink’s API must have experience in the following technologies.

If consuming the SOAP 1.1 (WS-BasicProfile 1.1) service endpoints:

- SOAP 1.1 Web Services
- HTTP Basic Authentication
- XML

If consuming the SOAP 1.2 service endpoints:

- SOAP 1.2 Web Services
- WS-Security, WS-SecureConversation, WS-Policy, WS-Trust
- XML

If consuming the REST service endpoints:

- REST concepts
- JSON
- HTTP Basic Authentication

4.4 System architecture

Before reviewing more detailed information about the various services in TransLink's OPIA API, it is important to have an understanding of how the individual services and components connect together into the overall product architecture.

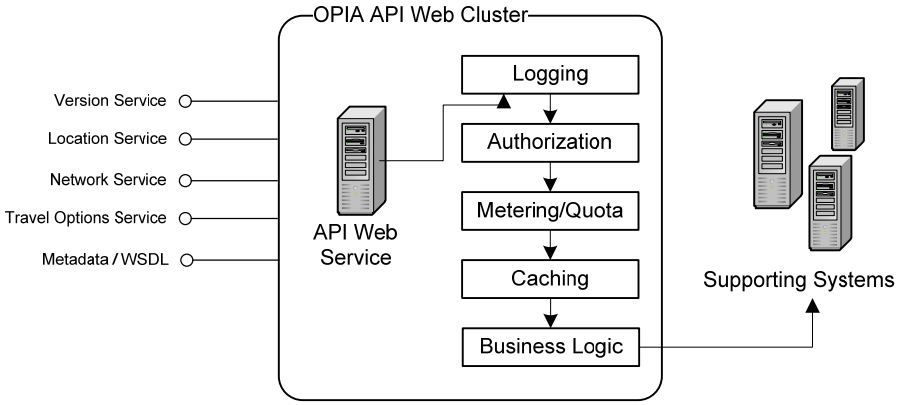
The API exposes a number of services which enable you to plan journeys, locate stops and retrieve timetable information in various ways. Metadata describing the services are exposed on each endpoint to assist in generation of client-side stubs in your language of choice.

To maximize interoperability the API exposes SOAP 1.1, SOAP 1.2 and REST endpoints. These industry standards were selected to maximize interoperability with the development languages of potential users.

All endpoint types require authentication. In the case of SOAP 1.2, endpoints are protected by WS-Security using UserId/Password credentials. REST and SOAP 1.1 endpoints utilize HTTP Basic Authentication. All endpoints require SSL.

In addition to authorization, the API also enforces Usage Quotas on a per-call basis. Quotas are determined by the security role(s) TransLink have allocated your account. Once you exceed your quota subsequent calls will result in a Quota fault/error. Refer to the Quota section for further information.

The figure below shows the flow of a typical API call:



4.4.1 Logging

TransLink may log your usage of the API to assist in diagnostics and to help us improve the service.

We may record the entire request and/or response body, IP addresses, credentials and any other information transmitted between you and the service.

4.4.2 Restrictions on use

If you are developing an on-device application (eg Android, iPhone app) rather than a website we **strongly recommend against calling the TransLink OPIA API directly from the device.**

Instead, you should build your own server-side service which wraps TransLink's API.

This recommendation exists for a number of reasons:

1) It ensures that your TransLink OPIA API credentials are never distributed or shared beyond your network boundary
2) Enables you to cache frequently accessed information to reduce load on TransLink's API servers, and minimize your quota usage
3) Improves performance by reducing mobile network traffic. A single call to your service may internally combine multiple calls to the TransLink OPIA API
4) It abstracts your application away from API changes, should there be any

4.4.3 Security and authorization

All API service methods require authorization. These credentials will be provided to you by TransLink after API account is setup. Naturally these credentials are for your exclusive use and must not be shared with other users.

Your account will be granted access to one or more security roles, which dictate the API services and functions you can access.

For SOAP 1.1

Authentication is achieved using HTTP Basic Authentication over SSL.

For SOAP 1.2

Authentication is achieved via WS-Security with UsernamePassword token authentication over SSL. The WSDL's associated with each SOAP endpoint exposes details of the supported security policies to enable client stub generation.

For REST

Authentication is achieved using HTTP Basic Authentication over SSL.

4.4.4 Quota and metering

Your API account will have one or more security roles associated with it. These roles dictate what you can access and how often. Metering is the action of monitoring usage against your quota and is performed transparently by the API upon each service call.

For accounts with multiple roles enabled, the most generous/least restrictive quota rule will be enforced

Quotas reset each hour and are recorded on a per-service, per-function basis. For example, your role may allow you up to 20,000 ResolveInput() calls per hour, 15,000 GetLocation() calls, etc.

If you exceed your quota in a given period the API will respond with a QuotaFault.

Contact TransLink on developers@translink.com.au to discuss your options if you are experiencing quota issues on a regular basis.

4.4.5 Data formats

The format of request and responses differs by endpoint (SOAP 1.1, 1.2 or REST).

For SOAP 1.1 and 1.2 endpoints the request/response types are always formatted as XML and their structure is dictated by the relevant SOAP specification in conjunction with the WSDL exposed by the services.

For REST endpoints, both XML and JSON response formats are supported. Callers can select between these two formats by sending the appropriate HTTP Accept or Content-type header – text/json (JSON) or text/xml for XML. Metadata is exposed in Swagger[1] format

The fallback if neither of these are specified is to XML format. Where both Accept and Content-type headers are specified, Accept takes precedence.

4.4.6 Caching

Caching the information returned by TransLink's API is key to building high performance, scalable solutions and ensuring that you do not exceed your quota.

---

[1] Swagger specification - https://github.com/wordnik/swagger-core/wiki

Most information returned by the API can be cached however there are certain classes of information which cannot be. For example, you should never cache the result of a Journey Plan request.

When designing your caching architecture keep in mind that Route, Trip and Timetable information is always scoped to a specific date. So whilst the recommended cache period for a Route is several days, the actual cached item's data only applies to the date it was retrieved for.

TransLink offers the following guidelines for caching returned data:

Type	Cache period	Unique constraint
Stop	7 days	Id
Line	7 days	Id
Route	7 days	RouteId
Fare	None	
Itinerary	None	
Leg	None	
TravelOptions	None	
Address	7 days	Id
Location	24 hours	Id
RoutesByLine	7 days	LineId Date
StopTimetable	7 days	Date Stop
StopTimetableTrip	7 days	TripId
Timetable	7 days	Date Route
Trip	7 days	Id

5. Developer resources

5.1 Summary

This section discusses additional resources of relevance only to developers integrating their solutions with the OPIA API.

5.2 REST documentation

The TransLink OPIA API exposes metadata, including documentation, in Swagger[2] format.

With the help of the Swagger Code-Gen project[3], this metadata can be converted directly into source code in a variety of languages including Java, Objective C, C# and PHP.

The OPIA API also contains an embedded copy of Swagger UI[4] which provides developers with simple, interactive documentation that both visualizes the API's and provides an in-built test-harness.

Note that services respond only over SSL, non HTTPS attempts will result in 404 Not Found. This includes metadata retrieval (ie Swagger definitions or WSDL)

Please refer to the section '0 -

Connecting to the API' below or visit https://opia.api.translink.com.au/v2 for more information and access to the embedded Swagger UI.

5.3 SOAP documentation

The API's SOAP endpoints expose metadata and documentation via their WSDL's. Documentation is embedded within the WSDL via xs:documentation nodes which enables code generators to automatically provide detailed method and parameter level information.

Please refer to the section '0 -

Connecting to the API' below or visit https://opia.api.translink.com.au/v2 for more information

5.4 Swagger UI test harness

To jumpstart development and provide a simple test harness for the REST API, you can utilize our embedded Swagger UI[5] instance, accessible via https://opia.api.translink.com.au/v2

Swagger UI provides a simple way to visualize methods exposed by the API, to view request/response metadata and to directly invoke calls from within your web browser:

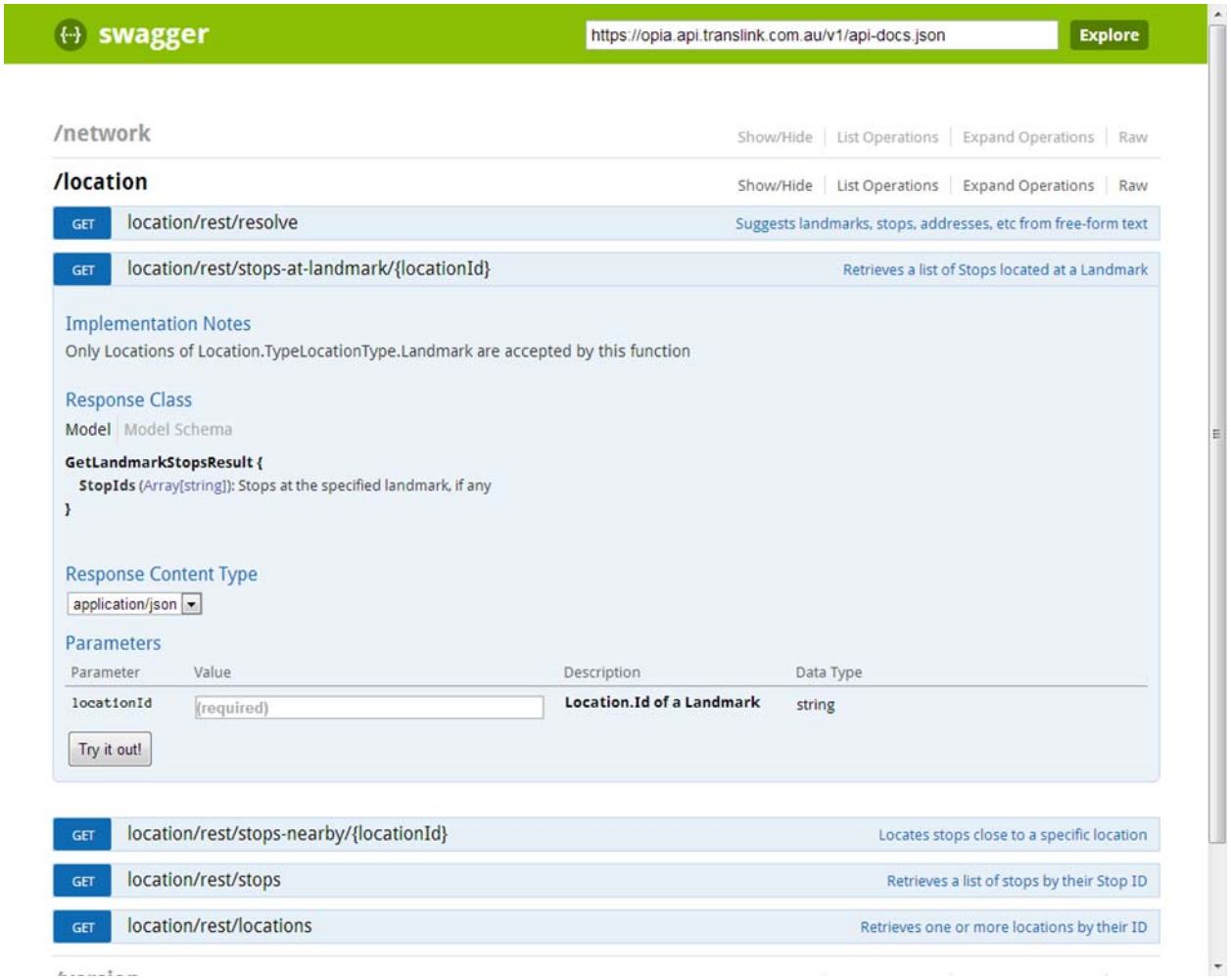
---

[2] Swagger specification - https://github.com/wordnik/swagger-core/wiki

[3] Swagger code-gen project - https://github.com/wordnik/swagger-codegen

[4] Swagger UI - https://github.com/wordnik/swagger-ui

[5] Swagger UI - https://github.com/wordnik/swagger-ui


swagger
https://opia.api.translink.com.au/v1/api-docs.json
Explore
/network
Show/Hide | List Operations | Expand Operations | Raw
/location
Show/Hide | List Operations | Expand Operations | Raw
GET location/rest/resolve
Suggests landmarks, stops, addresses, etc from free-form text
GET location/rest/stops-at-landmark/{locationId}
Retrieves a list of Stops located at a Landmark
Implementation Notes
Only Locations of Location.TypeLocationType.Landmark are accepted by this function
Response Class
Model | Model Schema
GetLandmarkStopsResult {
StopIds (Array[string]): Stops at the specified landmark, if any
}
Response Content Type
application/json
Parameters
Parameter
Value
Description
Data Type
locationId
(required)
Location.Id of a Landmark
string
Try it out!
GET location/rest/stops-nearby/{locationId}
Locates stops close to a specific location
GET location/rest/stops
Retrieves a list of stops by their Stop ID
GET location/rest/locations
Retrieves one or more locations by their ID

6. Connecting to the API

6.1 Overview

This section describes the API endpoints and how to connect to the various services they expose.

6.2 Service definitions – SOAP (WSDL)

Web service definitions in WSDL format can be retrieved from the endpoints below. Embedded xs:documentation nodes provide help at the method and parameter level.

Metadata must be retrieved over SSL.

Note the parameter ?wsdl can be replaced with ?singleWsdl to return the entire WSDL is a single file with no imports.

Service	URL
Version	https://opia.api.TransLink.com.au/v2/version**?wsdl**
Location	https://opia.api.TransLink.com.au/v2/location**?wsdl**
Timetable	https://opia.api.TransLink.com.au/v2/network**?wsdl**
Travel Options	https://opia.api.TransLink.com.au/v2/travel**?wsdl**

6.3 Service Definitions – REST (Swagger)

As per the Swagger specification, all services are linked via the root resource descriptor located at /api-docs.json

Metadata must be retrieved over SSL.

TransLink strongly advises you to utilize Swagger Code-Gen to generate stubs for interacting with the API.

Service	URL
Version	https://opia.api.TransLink.com.au/v2/api-docs.json
Location	https://opia.api.TransLink.com.au/v2/api-docs.json
Timetable	https://opia.api.TransLink.com.au/v2/api-docs.json
Travel Options	https://opia.api.TransLink.com.au/v2/api-docs.json

6.4 API versioning

Minor updates and bug fixes which do not require changes to the external service interface will continue to exist at the same endpoints dictated above.

Other changes in which involve modifications to the external service interface will be hosted at a different endpoint, and the old service interface will be left unchanged wherever possible.

You can always use the Version service to retrieve the current build version of the service. This information is automatically generated upon each new release, regardless of major or minor change.

API customers will be notified of major and minor updates using the email address TransLink has associated with their API user account.

6.5 Authenticating with the service

All calls to the OPIA API must be authorized using credentials previously provided to you by TransLink.

For SOAP 1.2

When connecting to the SOAP 1.2 endpoint, connections are authenticated using the webservice security extensions collectively known as WS-*, ie WS-Security, WS-Trust and WS-SecureSession.

All communication must be performed over SSL. Currently the service supports only username/password token authentication.

A SecurityFault will be generated if none or invalid credentials are passed to the service.

For REST and SOAP 1.1

For REST and SOAP 1.1 endpoints, connections can be made using HTTP Basic Authentication over SSL. HTTP authorization challenge occurs with HTTP 401, however you can avoid the round trip by sending authorization on your first request.

The sample below shows how to connect and authorize with the Location service in C# using the SOAP 1.2 binding after adding a Service Reference through Visual Studio 2010:

```
IApiLocationService client = new ApiLocationServiceClient();
client.ClientCredentials.UserName.UserName = “my api username”
client.ClientCredentials.UserName.Password = “my api password”
Location[] resolvedFrom = client.ResolveInput(“Queen St”, LocationType.None, 5);
```

Note that when the service reference was added in Visual Studio, the appropriate security configuration was generated by Visual Studio in the application's .config file. The key sections have been highlighted in red:

```
<wsHttpBinding>

<binding name="WSHttpBinding_IApiTimetableService" closeTimeout="01:01:00"
openTimeout="00:01:00" receiveTimeout="01:01:00" sendTimeout="01:01:00"
bypassProxyOnLocal="false" transactionFlow="false" hostNameComparisonMode="StrongWildcard"
maxBufferPoolSize="2147483647" maxReceivedMessageSize="2147483647" messageEncoding="Text"
textEncoding="utf-8" useDefaultWebProxy="true" allowCookies="false">

          <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
maxBytesPerRead="4096" maxNameTableCharCount="16384" />

          <reliableSession ordered="true" inactivityTimeout="00:10:00" enabled="false" />

          <security mode="TransportWithMessageCredential">

            <transport clientCredentialType="None" proxyCredentialType="None" realm="" />

            <message clientCredentialType="UserName" negotiateServiceCredential="false"
establishSecurityContext="false" algorithmSuite="Default" />

          </security>

        </binding>
```

7. API services

7.1 Summary

This section provides an overview of each service exposed by the API. Actual method and parameter level documentation can be retrieved from the services themselves through embedded metadata as described in the previous section (Swagger for REST, xs:documentation for SOAP).

7.2 Location service

The Location service provides details of locations within TransLink's network and provides mechanisms of retrieving information about them, and resolving them in various ways.

For example, the location service provides methods to:

- Resolve a given a piece of free-form text into an address, stop and/or landmark
- Locate stops close to an address or other location
- Find landmarks nearby a particular stop

7.3 Network service

The Network service provides information about TransLink's network including timetables for stops and routes.

It can assist you in finding information such as:

- The details of all services passing through a particular stop
- The path a particular route travels, and when
- Map co-ordinates for a route

7.4 Travel options service

The Travel Option service provides a way of determining journeys a customer could use to transit between two points.

It allows callers to retrieve a link directly to the journey plan on TransLink's website *or* the raw journey data to allow for custom presentation within an application.

7.5 Version service

The version service provides a mechanism of determining the current API build. The build details are automatically updated on each new release.

Due to internal testing the release numbers may not be sequential.

7.6 Service updates / service status

A feed of Service Updates is not currently available to the public. It is envisaged that this will be added to the API or provided via an RSS feed in the future.

8. Key terms

Term	Definition
Line	Trains and Ferries travel on a Line. Multiple Routes may travel on the same Line with variations. A route may also onto multiple Lines. For example: Airport line - routes BDBR, BDVL, BRBD, VLBD Gold Coast line – routes BDVL, BRVL, SHVL, VLBD, VLBR, VLDB Whilst Lines are a Train and Ferry concept, the API artificially exposes buses as one Line per bus Route for consistencies sake. For example, P88 bus line contains only the P88 bus.
Route	The path travelled by vehicles to pick up customers. Routes are identified by their Route Id for example. T:345, which is a combination of the Route's region identifier (T, C, N); and Route Code, for example P88, SHLP. Note that each Route has one or more Directions. For example, the P88 has both an Inbound and Outbound direction.
Service	An individual vehicle instance travelling some route. For example, the 4.10pm bus travelling on Route 111 is a service. As is the 8.29am train traveling Route BDVL on the Airport Line.
Station	A landmark which contains multiple Stops, eg 'Roma Street Busway Station' contains multiple stops – one per Platform. Stations do not have a Stop ID – but the Stops within them do.
Stop	A stop made by a vehicle on the network, eg bus stop, train stop, etc
Stop ID	Uniquely identifies a Stop on the TransLink network. A 6-digit zero left padded number, eg 000026
Vehicle	A bus, train or ferry
Region	Represents the different regions in the network. Eg. T = South East Queensland, C = Cairns, N = Non-TransLink